applied sciences

MDPI

Article

Error Detection for Arabic Text Using Neural Sequence Labeling

Nora Madi * and Hend Al-Khalifa

Information Technology Department, College of Computer and Information Sciences, King Saud University, Riyadh 12371, Saudi Arabia; hendk@ksu.edu.sa

* Correspondence: nmadi@ksu.edu.sa

Received: 24 June 2020; Accepted: 27 July 2020; Published: 30 July 2020

Abstract: The English language has, thus far, received the most attention in research concerning automatic grammar error correction and detection. However, these tasks have been less investigated for other languages. In this paper, we present the first experiments using neural network models for the task of error detection for Modern Standard Arabic (MSA) text. We investigate several neural network architectures and report the evaluation results acquired by applying cross-validation on the data. All experiments involve a corpus we created and augmented. The corpus has 494 sentences and 620 sentences after augmentation. Our models achieved a maximum precision of 78.09%, recall of 83.95%, and $F_{0.5}$ score of 79.62% in the error detection task using SimpleRNN. Using an LSTM, we achieved a maximum precision of 79.21%, recall of 93.8%, and $F_{0.5}$ score of 79.16%. Finally, the best results were achieved using a BiLSTM with a maximum precision of 80.74%, recall of 85.73%, and $F_{0.5}$ score of 81.55%. We compared the results of the three models to a baseline, which is a commercially available Arabic grammar checker (Microsoft Word 2007). LSTM, BiLSTM, and SimpleRNN all outperformed the baseline in precision and $F_{0.5}$. Our work shows preliminary results, demonstrating that neural network architectures for error detection through sequence labeling can successfully be applied to Arabic text.

Keywords: Arabic natural language processing; deep learning; error detection; neural sequence labeling; recurrent neural networks

1. Introduction

Error detection may help both natives and non-natives in producing correctly written text. Moreover, error detection has a wide range of applications, such as automatic essay scoring [1], style and grammar suggestion [2,3], and language learning systems [4]. Additionally, specific types of errors in a text may reflect the writer's age, proficiency, dialect, and native language. Therefore, the output of error detection models could be used for creating a profile of a certain writer [5]. Such profiles can be used for author identification, native language identification, or even the level of education.

Using neural network models has shown significant results for different Natural Language Processing (NLP) tasks [6]. There exist a variety of neural network architectures, ranging from Recurrent Neural Networks (RNNs) [7] to more advanced Long–Short-Term Memory Networks (LSTMs) [8] and other varieties, such as bidirectional LSTMs (BiLSTMs) [9]. RNN and its variations are very appealing for NLP as they are specialized in learning sequential data including text. Hence, they are commonly used for the sequence classification task and have also recently found success at automating grammatical error detection [1] and correction [10] through language modeling [11] and machine translation [12].

Although such approaches have been used to detect errors in English text, error detection based on neural networks has not yet been explored for Arabic. Essentially, earlier state-of-the-art

systems for Arabic depend on employing combinations of rule and statistical-based approaches [3]. However, taking into account the number and complexity of Arabic grammar rules, there is a need for more advanced approaches that can cope with these challenges.

In this paper, we treat error detection as an independent task and present the first experiments using neural network models for the task of error detection in Arabic text. We mainly investigate and report the results of comparing three neural networks architectures: SimpleRNN, LSTM, and BiLSTM. We also explore the effects of training datasets with different sizes on the overall performance by providing additional training data to each one of the models. The details will be presented for the dataset, which we created and later augmented especially for the task of error detection for Arabic. The types of errors included in the dataset are syntax errors, morphological errors, semantic errors, linguistic errors, stylistic errors, spelling errors, and the use of informal as well as borrowed words. Erroneous words that appeared within some sentences in the corpus are presented in Table 1, showing a sample for each type of error and the reason it is considered an error.

Table 1. Arabic Error Examples.

Error	Translation	Type of Error	Reason of Error
لا يفكرون سوى في الشقاء	They only think of misery	Syntax Error	Using the preposition في after the exceptive particle سوى is incorrect because سوى must be followed by a noun or pronoun
تظافر	Unite	Morphological Error	The correct word is تضافر (it is a common mistake to use ظ in place of ض and vice versa, due to their similar pronunciation)
صدفة	Coincidence	Semantic Error	The correct Arabic word to denote coincidence is مصادفة, not صدفة
حُصة	Share	Linguistic Error	The first letter ح, written as حُ using Fatha diacritic, should be written and pronounced with the Kasra diacritic حِصة
برغم	Despite	Stylistic Error	The word رغم should not be preceded with the preposition ب
خطت	Plan	Spelling Error	Using the regular ت instead of Ta-Marbuta ة
مصاري	Money	Use of Informal Word	Using the informal word مصاري instead of the formal words مال or نقود
كاريزما	Charisma	Use of Borrowed Word	Using the borrowed word كاريزما instead of the Arabic words هيئة or قبول

The rest of the paper is organized as follows: in Section 2, we give a background and overview of the related work; the details of our corpus are presented in Section 3; implementation and experiments are described in Section 4; results from our experimentation are illustrated in Section 5; Section 6 provides discussion and analysis of the results is presented; Section 7 concludes the paper along with limitations and future work.

2. Background and Related Work

In this section, an explanation of the Arabic language and its grammar is provided. Then, related works in text correction approaches for a number of languages are discussed. Lastly, previous work focusing on error correction for Arabic is presented.

2.1. Arabic Language and Grammar

Arabic is a Semitic language that is rich in its morphology (internal word structure) and syntax (arrangement of words to make sentences and phrases). Although it has many variations (dialects) around the Arab World, Modern Standard Arabic (MSA) is the official language. MSA is a simple form of Classical Arabic (CA), the language of the Qur'an (Islam's Holy Book). However, MSA has much more modern vocabulary [13]. It is the main language used in media and education. Additionally, MSA is primarily written, not spoken [13].

Arabic words are generally classified into nouns, verbs, and particles [3]:

1. Noun: A noun in Arabic is a name of a person, thing, or idea. It is divided into derivatives and primitives. These could be further classified by number, gender, definition, and case.
2. Verb: The verb is any word that conveys an action. It is divided according to tense, structure, mood, and voice. These could be further classified by number, person and gender.
3. Particle: The particle is a word that can precede a noun, a verb, or both. Particles include prepositions, conjunctions, interrogative particles, exceptions, and interjections.

Moreover, an Arabic sentence can be one of two forms [3]:

1. Nominal: A nominal sentence contains two structures: inchoative (مبتدأ) and enunciative (خبر). Enunciative can be verbal/nominal sentence. Additionally, a nominal sentence can start with Inna/Kan and its sisters, which change its case ending.
2. Verbal: A verbal sentence contains two structures: verb and subject. A transitive verb needs to have one or more object. In its passive voice it includes a verb and a proagent (نائب فاعل).

Arabic grammar rules are very complex and might even be confusing to Arabic natives. The difficulty is due to several reasons—for example: (1) the sentence length and complex grammar; (2) the relatively free word order in Arabic [3]. Therefore, automatically checking Arabic grammar for errors could help improve the quality of written Arabic text.

2.2. Grammar Detection and Correction Approaches

Grammar error detection and correction has been done for various languages [14], including Chinese [15], Greek [16], and Swedish [17]. Different approaches have been exploited for automatically detecting and correcting text. These include rule-based, syntax-based, and statistical-based classification and neural networks approaches. The rule-based technique has been applied to languages such as Amharic [18] and Latvian [19]. Additionally, a syntax-based technique has been used for Danish [20] and a statistical-based technique has been used for Amharic [18], as well as Filipino [2].

Moreover, classification and Neural Networks have been employed for automatically detecting and correcting grammatical errors. The most common classifiers used in previous work include maximum entropy models, naive Bayes and support vector machines (SVM) such as in [10,21] for English. On the other hand, neural networks have been used in [12,22] by applying Recurrent Neural Networks (RNNs) on various English corpora to perform Neural Machine Translation (NMT) for error correction.

Other works, such as in [1,23–25], focused solely on identifying grammar errors in English without correcting them by employing sequence labeling to generate the probability of each word in the sentence being correct or not.

2.3. Grammar Detection and Correction in Arabic

In the context of Arabic, there have not been any efforts we know of which concentrate exclusively on grammar error detection. However, previous work has been done targeting error correction. One of the earliest works was Arabic GramCheck by Shaalan [3], a syntax-based grammar checker for identifying Arabic grammatical errors while providing feedback. The grammar checker included two

parts: an Arabic morphological analyzer and a syntactic parser. A total of 100 Arabic sentences were used to evaluate the system in comparison to a commercial system. The results showed that Arabic GramCheck outperformed the commercial Arabic grammar checker. However, Arabic GramCheck was evaluated on a small number of manually generated Arabic sentences.

In [26], Tomeh et al. presented a pipeline approach which included error detection. They approached the error detection task as a sequence-labeling problem by training an SVM classifier using morphological and lexical features including Part-of-Speech tags. For their experiments, they used QALB corpus [27]. The precision of the error detection model was very low (0.345), which leaves much room for improvement.

In terms of the use of neural networks, we are aware of only three attempts that have employed neural models for end-to-end grammar error correction. The first was by Ahmadi et al. [28] who treated error correction as monolingual NMT using sequence-to-sequence and attention-based models. Using QALB, the results showed that their models could be effectively applied to the task of error correction for the Arabic language. However, their work was outperformed by the work of Watson et al. [29]. They investigated both character embeddings and pre-trained word embeddings, employing a number of embedding models. They achieved state-of-the-art results in the QALB corpus. On the other hand, Solyman et al. [30] utilized Convolutional Neural Networks with an attention mechanism using QALB. However, the results were preliminary, and the proposed model was not fully applied.

This paper is a continuation of the work in [31] as an effort to produce a model to be used within a tool for Arabic grammar error detection.

3. Corpus and Annotation

In this section, we present the corpora used for training and testing, the process of preparing them, and their statistics. We first extracted sentences from the "Linguistic Error Detector-Saudi Press as a Sample" book [32] to produce a parallel corpus as described in [33]. It is a monolingual parallel corpus of Arabic text, called A7'ta. It has 470 incorrect sentences and their 470 correctly written counterparts. This parallel corpus was used to create a token-level labeled corpus suitable for the task of error detection. Then, we augmented this corpus as an attempt to increase the number of sentences and see how this would affect the models' performances. All text in the corpora is not diacritized. All datasets have been made public [33] and can be found at https://github.com/iwan-rg/.

3.1. Token-Level Labeled Corpus

Most of the work that focuses on grammatical error detection using Neural Networks treat this task as a sequence labeling problem where the model receives a sentence and takes context into consideration to produce the probability of each word in the sentence being correct or incorrect [1]. This kind of task requires training on a token-level labeled corpus such that every word in a sentence is labeled as correct or incorrect. However, to the best of our knowledge, there is no such corpus that has been created for Arabic sentences.

Therefore, for the purpose of this work, we manually transformed the parallel corpus in [33] to accommodate a token-level error detection task similar to the work in [1]. We did not consider punctuation marks or punctuation errors in the experiments presented here. We extracted all the incorrect sentences and labeled each word as being correct (c) or incorrect (i). We used the corrected versions of the sentences and compared them to their erroneous counterparts to indicate which word(s) are those that should be labeled as incorrect.

In case there was an extra word in the incorrect sentence, it was labeled as "i". Furthermore, in case there was a missing word, the word after the incorrect gap was labeled as "i". Table 2 shows statistics related to the original corpus.

Table 2. Corpus Statistics.

Feature	Original Corpus	Augmented Corpus
Number of Sentences	494	620
Number of Words	1216	1380
Number of Tokens	2027	2705
Number of Correct Words	1438	2122
Number of Incorrect Words	576	592

3.2. Augmented Token-Level Labeled Corpus

In order to increase the size of the corpus, we manually augmented it by adding more sentences. We applied four types of augmentation, as explained in Table 3. In the table, "Type" is the name of the type of augmentation inspired by the "Case", "Case" is the situation where this type of augmentation was applied, and "Explanation" is a description of how the augmentation was done. The book used to extract the corpus sentences included explanations of linguistic rules. These explanations included example sentences, which we collected for augmenting the corpus. We called this type of augmentation "From Rules". These sentences occupied 57.14% of the total augmented sentences. Table 2 shows statistics related to the augmented corpus. Augmentation has caused the number of sentences to increase by 25.5% in comparison to the original corpus, which had 494 sentences.

Table 3. Types of Augmentation.

Type	Case	Explanation
Deletion	Incorrect sentence has a missing word	Augment by duplicating the sentence and adding the missing word
Syntax	Words in the incorrect sentence are incorrectly ordered	Augment by duplicating the sentence and reordering the words correctly
From Rules	There were some example sentences in "The Rules" of the book	Extracting sentences from rules
Multi-error	Incorrect sentence has multiple errors	Augment by duplicating the sentence as many times as there are errors, fix only one error per sentence

4. Implementation and Experiments

In this section, we present the details of our implementation and experiments. First, we present the toolkit we used as well as the corpus preparation process. Then, we describe the neural network architectures, which we used for the experiments. Next, we describe which hyperparameters we tuned, and then the measure we used for evaluation—lastly, we present our baseline.

4.1. Toolkit and Corpus Preparation

Keras with a TensorFlow backend is used for all the experiments. Particularly, Keras version 2.2.4 and Tensorflow version 1.12.0 are used. For the use of neural networks in Keras, input sequences need to be equally long for modeling. Therefore, we pad our sentences to a uniform maximum length, which is set to 17, since this is the length of the longest sentence in the corpus. Additionally, label sequences were padded with zeros to the maximum length. The model learns that padding values carry no information.

4.2. Neural Network Architectures

Neural networks have been used for various sequence-labeling NLP tasks, such as Part of Speech (POS) tagging and Named-Entity Recognition (NER). Similarly, we investigated treating error detection for Arabic text as a sequence-labeling task by utilizing token-level labeled Arabic sentences. For the sequence-labeling model, we built a neural network, which takes a sequence of tokens (words) as

input, and outputs the probability of each word in the sentence. We explored a few types of neural networks, including a SimpleRNN, LSTM, and BiLSTMs.

We used a similar sequence of layers for every experiment while varying the hyperparameters and RNN types (i.e., SimpleRNN, LSTM, or BiLSTM) for comparison. A SimpleRNN layer is just plain vanilla RNN in Keras where the output is to be fed back to input.

Models are basically composed of an input layer, a single embedding layer, a dropout layer, a SimpleRNN layer (or 1–2 LSTM layers or 1–2 BiLSTM layers) with recurrent unit dropout, and TimeDistributed wrapped Dense output layer with sigmoid activation. The sigmoid function outputs a probability for each class as a value between 0 and 1.

For our task, we trained word embeddings using the Embedding layer in Keras [34]. The Embedding layer receives the input sequence and encodes it into a sequence of vectors, thereby learning an embedding for all the words in the training dataset. The advantage of this approach in comparison to using pre-trained embeddings is that learned vectors are targeted to the specific corpus as well as the error detection task.

4.3. Hyperparameters

Empirical parameters are used as candidate values for the model. A manual random search method is used to find the best hyperparameter combinations, taking into consideration the small corpus size and potential overfitting. We observed the effects of altering these values over the performance in order to develop the best possible model given the available corpus size. For example, parameters such as embedding size, batch size, and number of units are varied. Moreover, we explored multi-layer variants of each of the neural network models investigated. Table 4 provides a summary of the used hyperparameters and associating values.

Table 4. Hyperparameter Summary.

Hyperparameter	Values
Loss	Weighted binary cross entropy
Number of Layers	1 and 2
Hidden Layer Size	25, 50, 75, 100, and 125
Embedding Size	32, 50, 75, and 100
Batch Size	1, 8, 16, 32, and 64
Learning Algorithm	Nadam with default values
Early Stopping	monitor = 'val_loss', patience = 2
Dropout	None, 0.1, 0.2, and 0.3

The models are optimized using Nadam, which is an Adam optimizer [35] variant with Nesterov momentum [36]. We left the parameters of this optimizer at their default values as recommended by Keras documentation [37]. Training is stopped when the validation loss stops decreasing over two epochs.

Moreover, we optimized the model by minimizing cross-entropy between the predicted label distributions and the annotated labels. However, our dataset is imbalanced because there are far more correct words than incorrect words. Therefore, we defined and used a weighted binary cross entropy function [38]. This function allowed trading off between recall and precision by adjusting the weight of positive errors in comparison to negative errors.

4.4. Evaluation

To train a model, data could be split into training and testing sets. However, doing so on small datasets would cause testing scores to vary depending on which data parts were used for training and which were used for testing. Therefore, using this approach on small data in particular would not provide reliable results [39].

So, to make better use of our corpus, we used 10-fold cross validation and the averages of precision, and the recall for all iterations of a single model were calculated and collected for analysis. Various research papers apply k-fold cross validation when dealing with smaller datasets [40–46].

Additionally, we calculated and observed $F_{0.5}$ because, for the task of error detection, feedback accuracy (i.e., precision) has a higher priority than coverage (i.e., recall). Other evaluation measures used in the literature, such as the M^2-scorer, were not directly valid for error detection because they require that a system suggests a correction.

Additional code was written to be able to evaluate precision, recall, and $F_{0.5}$ by comparing true labels to predicted labels and computing true negatives, true positives, false negatives, and false positives.

4.5. Baseline

For our baseline, we tested the corpus sentences using a commercially available rule-based Arabic grammar checker (Microsoft Word 2007). We compared the errors detected by the grammar checker with the golden labels of the corpus to obtain the results. Consequently, precision for the baseline was 75.6%, while recall was 91.96% and $F_{0.5}$ was equal to 78.39%.

5. Results

In this section, we present our experimental results using the corpora we created. The first part describes the results received from training and testing the models using 494 sentences. The second part shows the results when using the best performing model configurations to train and test the augmented corpus of 620 sentences. Lastly, the third part reports model performances when tested on 50 sentences from QALB.

5.1. Results of the Models

Table 5 presents the best results from each type of neural network we investigated using our un-augmented corpus and compares them with other works. The BiLSTM model performed the best of all systems that we developed in terms of precision and $F_{0.5}$. However, SimpleRNN performed better than LSTM in terms of precision and $F_{0.5}$. LSTM outperformed the other models in terms of recall. Unlike LSTM, SimpleRNN is not suitable for processing long sequences, such as text [39]. Therefore, we expected that LSTM would perform better in our experiments. However, one reason that SimpleRNN actually performed better than LSTM might be because most of the sentences in the training set are short, where the longest was only 17 words long. Additionally, because the task was considered simple (i.e., sequence labeling) and the training set was small, a simpler network was more suitable. LSTM could most probably perform better on more complex tasks and datasets [39].

Table 5. Evaluation Results.

Feature	Precision	Recall	$F_{0.5}$
Baseline	75.6	91.96	78.39
SimpleRNN	78.09	83.95	79.62
LSTM	76.21	**93.80**	79.16
BiLSTM	**80.74**	85.73	**81.55**
[26]	34.5	80.0	-
[1]	46.1	28.5	41.1
[47]	79.10	21.19	51.14
[48]	54.41	91.79	-
[49]	89.71	62.48	82.52
[50]	66.47	54.11	63.57

Additionally, the fact that LSTM outperformed the others in recall is because LSTM was better than other models with regard to not considering correct words as mistakes. So, it was better at recognizing that correct words were in fact correct.

LSTM, BiLSTM, and SimpleRNN all outperformed the baseline in precision and $F_{0.5.}$ However, the baseline had higher recall than SimpleRNN and BiLSTM.

Table 6 shows the hyperparameter settings for each of the models in Table 5. We noticed that BiLSTM only needed one layer of size 25, the smallest sized model of the three, to perform the best. Implying that BiLSTM was more powerful in being able to select the most important structure in the input data to model while being small in size.

Table 6. Best Models' Hyperparameters.

Hyperparameter/Model	SimpleRNN	LSTM	BiLSTM
Layers	2	2	1
Layer Size	50	125	25
Embedding Size	50	32	100
Batches	1	1	32
Dropout	0.2	0.1	none
Optimizer	Nadam	Nadam	Nadam

5.2. Increasing Corpus Size

Since we were dealing with quite a small training corpus, it seemed that the size of the corpus would be a limiting factor. Therefore, we decided to manually augment our corpus (explained in Section 3.2) and use that to see the effects of increasing data. For training, we used the best configuration of each model (as in Table 6) in order to see the impact on performance when more data are used.

Table 7 presents the results of this experiment. In comparison to Table 5, all three models actually performed significantly better when trained on more data, even though the augmented corpus was only 25.5% larger than the original. Additionally, with more data, LSTM outperformed SimpleRNN over all measures. This indicates that LSTM would handle larger data better. While BiLSTM achieved better recall when trained on the augmented corpus, it had the least recall in comparison to the other two, meaning it had given more false negatives. Moreover, the three neural networks outperformed the baseline in recall and $F_{0.5}$.

Table 7. Results of Best Configuration When Trained on Augmented Dataset.

Figure 0.	Precision	Recall	$F_{0.5}$
Baseline	75.6	91.96	78.39
SimpleRNN	81.22	91.75	83.09
LSTM	81.42	94.57	83.71
BiLSTM	81.68	91.54	83.45

5.3. Testing on QALB

In order to further validate our model, we used a portion of the Qatar Arabic Language Bank (QALB) corpus [51] for testing. We did not initially use it to train our models mainly because the type of annotation in QALB was targeted at error correction, which was not suitable for our task (error detection). Additionally, we wanted to support efforts produced by language experts, add to the existing number of Arabic corpora, and investigate sequence labeling for detecting errors in Arabic on a smaller scale and simpler model.

QALB is, to the best of our knowledge, the only corpus for Arabic grammar checking. We took 50 MSA sentences from testing data of QALB 2014, and only used the first 10–17 words from each sentence to mimic the lengths of sentences in the training corpus and to speed up the annotation process. This resulted in having 516 words and 781 tokens for the QALB test corpus. Punctuation

marks were eliminated, and punctuation edits were not taken into consideration. Furthermore, labeling was manually done, similar to our training corpus. In order to label this corpus, we depended on the original annotation in QALB to lead the labeling process. For example, an edited word in the QALB sentence was labeled as "i" in the token-level labeled test corpus.

We ran QALB using the best configuration of each model (as in Table 6) in order to predict the labels of the words in our QALB test set. Table 8 shows the results of this experiment. While LSTM had the lowest performance in terms of precision and $F_{0.5}$ on our training set, it performed the best when tested on QALB. On the contrary, BiLSTM had the lowest performance on QALB over all measures. This might be a sign that BiLSTM was lacking regularization or further tuning.

Table 8. QALB Test Results.

Feature	Precision	Recall	$F_{0.5}$
Baseline	75.6	91.96	78.39
SimpleRNN	84.25	67.72	80.33
LSTM	84.81	68.04	80.83
BiLSTM	82.97	41.61	69.21

6. Discussion and Analysis

For error detection in Arabic, Tomeh et al. [26] approached the task as a sequence labeling problem by training an SVM classifier using morphological and lexical features, such as lemmas and POS tags. The model labels the incorrect words in the text with a tag before passing them on for correction. Recall of the error detection model was 80%, whereas precision was 34.5%. Both scores are considered low in comparison to the results of the neural network models in Table 5.

On the other hand, the first notable application of neural sequence labeling to the task of error detection was done by Rei and Yannakoudakis [1]. They performed neural sequence labeling on English corpora and reported the results using precision, recall, and $F_{0.5}$. They achieved the best result on BiLSTM with $F_{0.5}$ = 41.1%, which was improved to $F_{0.5}$ = 41.88% when integrating the character-level component in [24]. Similarly, Yannakoudakis et al. [47] treated error finding as a neural sequence labeling task with the character-level component, while using a small set of features that were easily computed and needed no linguistic input. Their detection model achieved an $F_{0.5}$ score of 51.14%. On the other hand, Liu et al. [48] utilized Conditional Random Field for sequence labeling to automate Chinese grammar error detection without the use of any external features. Yuan et al. [49], however, did incorporate some external features to their sequence labeling model for English language error detection and achieved 82.52% in $F_{0.5}$. Additionally, Bell et al. [50] used BiLSTM sequence labeling for detecting errors but the model used a forward Language Model to predict the following token and a backward Language Model to predict the preceding token achieving $F_{0.5}$ = 63.57%.

All of these works utilized multiple large corpora and more complex models for training and testing. However, we believe that this is the first application of neural sequence labeling on Arabic text. Considering the small size of training corpus, we had expected that precision would be easier to achieve than recall. On the contrary, the results revealed a noticeable trend throughout all experiments: recall was mostly easier to achieve than precision. This is most probably due to the shortness of the sentences in the training set.

In addition, we found that smaller batches performed better than larger batches on our small dataset, which agrees with the work in [52]. Additionally, configurations using smaller batches were able to generalize better on new data than larger batches. This was apparent in our results for SimpleRNN and LSTM.

Table 9 presents four examples and their labels in five cases: gold standard and the baseline commercial Arabic grammar checker, as well as the testing results of SimpleRNN, LSTM, and BiLSTM models. Additionally, the specific type of error for each example is shown, as was mentioned in [32].

Table 9. Sentence Examples and Labels.

Sentence	Translation	Gold Label	Type of Error	Baseline	SimpleRNN	LSTM	BiLSTM
و	And	c		c	c	c	c
مذاق	taste of	c		c	c	c	c
الكراث	leeks	c		c	i	i	c
قريبا	is close	i	Accusative predicate	c	i	i	c
من	to	c		c	c	c	c
مذاق	taste of	c		c	c	i	c
البصل	onions	c		c	i	i	c
مشيرا	pointing	c		c	c	c	c
إلى	out	c		c	c	i	c
أن	that	c		c	c	c	c
ذلك	that is	c		c	c	c	c
خير	a good	c		c	i	c	c
موقف	attitude	c		c	i	i	c
يحتذى	to be looked	c		c	c	c	c
به	up to	c		c	c	c	c
مثالا	and honest	i	Accusative word related to the predicate of inna "إن" through conjunction	c	c	c	c
صادقا	example	i		c	i	c	c
أشار	Pointed	c		c	c	c	c
إلى	out	c		c	c	c	c
إن	that	i	Using "إن" instead of "أن"	i	c	i	i
ورشة	the	c		c	c	i	c
العمل	workshop	c		c	c	c	c
موجهة	is directed	c		c	c	i	c
اتخذها	taken by	c		c	c	c	c
الباعة	vendors	c		c	c	c	c
الجائلين	in streets	i	Accusative adjective of a nominative	c	i	c	i

Overall, when observing the predictions on test sets, BiLSTM had the best performance; it was better at learning sentence structure in general since LSTMs are capable of handling long-term dependencies by applying a linear update to the internal cell representation, unlike SimpleRNN. More importantly, it is because BiLSTMs are able to incorporate context on both sides of every token.

While SimpleRNN and LSTM also performed well, they were able to correctly learn and identify that correct words were correct. This is due to the fact that there were far more correct words in the training corpus than incorrect words. This might be an indication that a more balanced corpus would have enabled models to learn both correct and incorrect words equally. For the same reason, BiLSTM predicted most words as correct as can be seen in the first two examples in Table 9. However, the difference here is that BiLSTMs learn the sentences forward as well as backwards, which enabled the model to learn more correct word usage. At the same time, it was more accurate

than SimpleRNN and LSTM when predicting that incorrect words were incorrect, as can be seen in the third example in Table 9. Although the examples in the table show similar results in the baseline and BiLSTM, the commercial checker had actually missed many erroneous words considering them as correct as shown in the fourth example. This indicates that a rule-based checker may not be sufficient enough in detecting particular mistakes. This is because it is difficult to conduct fully exhaustive rules, especially when writing mistakes in a language are not constant.

7. Conclusions, Limitations and Future Work

In this paper we focused on the detection of errors in MSA text using neural sequence labeling. We presented a background and related work on Arabic language and approaches to Automated Grammatical Error Detection and Correction for Arabic and other languages. We also described the corpora we created for this task and presented our experimental setting. Finally, we showed our results and discussed them.

One of the main limitations of this project was the lack of data, which in turn called for using cross-validation. Although we were satisfied with the results, models were simplified in order to be able to accommodate the small dataset. However, we believe that larger data and larger networks would be able to generalize more when presented with new data. Additionally, our corpora sentences were short (maximum 17 words); however, we believe that the model would have benefited from longer sentences in order to learn from longer sequences of words.

Furthermore, the fact that neural networks are stochastic required the presentation of fixed seeds into random number generators. It has been shown that results may change when different seed numbers are used. Therefore, every set of hyperparameter configurations needs to be evaluated a number of times for every seed number, such that results are averaged and compared. A number of 5 up to 30 seeds could be used randomly. This process would enable drawing conclusions and making fair comparisons between different models [52]. In our case, we fixed a single seed value for all experiments (to 42), because, for the lack of resources, we could not run every configuration multiple times for different seeds. Nevertheless, this does not hinder the fact that our experiments are reproducible.

The results shown in this paper can be extended and improved in a number of ways. We suggest using a larger training set with longer sentences, conducting more experimentation with more hyperparameters, using deeper networks with larger embedding size, and running configurations multiple times using a variety of seed values and reporting the averaged results. Additionally, evaluating the performance of the different proposed neural network architectures per error type could be useful and may facilitate a vehicle for comparing systems. Moreover, incorporating the POS tags in the neural network could further improve the error detection.

As far as we know, we are presenting the first results of applying neural sequence labeling on Arabic text. Additionally, these are the first results of using the A7'ta corpus [33] in an experiment. We believe that the configurations we reported would show good performance on any other comparable Arabic corpus. The code for the experiments is available at https://gist.github.com/iwan-rg/4e7f522a53e664607c2a3e664f4c076a.

Author Contributions: Conceptualization, N.M. and H.A.-K.; methodology, N.M. and H.A.-K.; software, N.M.; validation, N.M.; formal analysis, N.M. and H.A.-K.; investigation, N.M.; resources, N.M.; data curation, N.M.; writing—original draft preparation, N.M.; writing—review and editing, H.A.-K.; visualization, N.M. and H.A.-K.; supervision, H.A.-K.; project administration, H.A.-K. All authors have read and agreed to the published version of the manuscript.

Funding: Research was funded by the "Research Center of the Female Scientific and Medical Colleges", Deanship of Scientific Research, King Saud University.

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

References

1. Rei, M.; Yannakoudakis, H.; Erk, K.; Smith, N.A. Compositional sequence labeling models for error detection in learner writing. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, Berlin, Germany, 7–12 August 2016; pp. 1181–1191.
2. Go, M.P.; Nocon, N.; Borra, A. Gramatika: A grammar checker for the low-resourced Filipino language. In Proceedings of the TENCON 2017–2017 IEEE Region 10 Conference, Institute of Electrical and Electronics Engineers (IEEE), Penang, Malaysia, 5–8 November 2017; pp. 471–475.
3. Shaalan, K. Arabic GramCheck: A grammar checker for Arabic. *Software: Pr. Exp.* **2005**, *35*, 643–665. [CrossRef]
4. Chodorow, M.; Gamon, M.; Tetreault, J.R. The utility of article and preposition error correction systems for English language learners: Feedback and assessment. *Lang. Test.* **2010**, *27*, 419–436. [CrossRef]
5. Modaresi, P.; Liebeck, M.; Conrad, S. Exploring the effects of cross-genre machine learning for author profiling in PAN 2016. In *Notebook for PAN at CLEF 2016*; CLEF: Évora, Portugal, 2016; pp. 970–977.
6. Goldberg, Y. Neural network methods for natural language processing. *Synth. Lect. Hum. Lang. Technol.* **2017**, *10*, 1–309. [CrossRef]
7. Elman, J.L. Finding structure in time. *Cogn. Sci.* **1990**, *14*, 179–211. [CrossRef]
8. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef]
9. Schuster, M.; Paliwal, K. Bidirectional recurrent neural networks. *IEEE Trans. Signal Process.* **1997**, *45*, 2673–2681. [CrossRef]
10. Rozovskaya, A.; Chang, K.-W.; Sammons, M.; Roth, D.; Habash, N. The illinois-columbia system in the CoNLL-2014 shared task. In Proceedings of the 18th Conference on Computational Natural Language Learning: Shared Task, Baltimore, Maryland, 26–27 July 2014; pp. 34–42.
11. Hdez, S.D.; Calvo, H. CoNLL 2014 Shared Task: Grammatical error correction with a syntactic N-gram language model from a big corpora. In Proceedings of the 18th Conference on Computational Natural Language Learning: Shared Task, Baltimore, Maryland, 26–27 July 2014; pp. 53–59.
12. Yuan, Z.; Briscoe, T.; Knight, K.; Nenkova, A.; Rambow, O. Grammatical error correction using neural machine translation. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego, CA, USA, 12–17 June 2016; pp. 380–386.
13. Habash, N.Y. Introduction to arabic natural language processing. *Synth. Lect. Hum. Lang. Technol.* **2010**, *3*, 1–187. [CrossRef]
14. Madi, N.; Al-Khalifa, H.S. Grammatical error checking systems: A review of approaches and emerging directions. In Proceedings of the 2018 13th International Conference on Digital Information Management (ICDIM), Berlin, Germany, 24–26 September 2018; pp. 142–147.
15. Fu, K.; Huang, J.; Duan, Y. Youdao's winning solution to the NLPCC-2018 task 2 challenge: A neural machine translation approach to Chinese grammatical error correction. In Proceedings of the CCF International Conference on Natural Language Processing and Chinese Computing, Hohhot, China, 26–30 August 2018; pp. 341–350.
16. Gakis, P.; Panagiotakopoulos, C.; Sgarbas, K.; Tsalidis, C.; Verykios, V. Design and construction of the Greek grammar checker. *Digit. Sch. Humanit.* **2016**, *32*, 554–576. [CrossRef]
17. Gudmundsson, J.; Menkes, F.; Hagelbäck, J. Swedish Natural Language Processing with Long Short-Term Memory Neural Networks-A Machine Learning-Powered Grammar and Spell-Checker for the Swedish Language. Bachelor's Thesis, Linnaeus University, Småland, Sweden, 2018.
18. Gebru, A.T. Design and Development of Amharic Grammar Checker. Master's Thesis, ADDIS ABABA University, Addis Ababa, Ethiopia, 2013.
19. Deksne, D. A new phase in the development of a grammar checker for Latvian. In Proceedings of the 7th International Conference Baltik HLT 2016, Riga, Latvia, 6–7 October 2016; Volume 289, pp. 147–152.
20. Bick, E. DanProof: Pedagogical spell and grammar checking for Danish. In Proceedings of the Recent Advances in Natural Language Processing, Hissar, Bulgaria, 7–9 September 2015; pp. 55–62.
21. Wang, P.; Jia, Z.; Zhao, H. Grammatical error detection and correction using a single maximum entropy model. In Proceedings of the 18th Conference on Computational Natural Language Learning: Shared Task, Baltimore, Maryland, 26–27 June 2014; pp. 74–82.

22. Xie, Z.; Avati, A.; Arivazhagan, N.; Jurafsky, D.; Ng, A.Y. Neural language correction with character-based attention. *arXiv* **2016**, arXiv:1603.09727.
23. Liu, Z.-R.; Liu, Y. Exploiting unlabeled data for neural grammatical error detection. *J. Comput. Sci. Technol.* **2017**, *32*, 758–767. [CrossRef]
24. Rei, M.; Crichton, G.K.O.; Pyysalo, S. Attending to characters in neural sequence labeling models. In Proceedings of the COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, Osaka, Japan, 11–16 December 2016; pp. 309–318.
25. Kaneko, M.; Sakaizawa, Y.; Komachi, M. Grammatical error detection using error- and grammaticality-specific word embeddings. *J. Nat. Lang. Process.* **2018**, *25*, 421–439. [CrossRef]
26. Tomeh, N.; Habash, N.; Eskander, R.; Le Roux, J. A pipeline approach to supervised error correction for the QALB-2014 shared task. In Proceedings of the EMNLP 2014 Workshop on Arabic Natural Language Processing (ANLP), Doha, Qatar, 25 October 2014; pp. 114–120.
27. Zaghouani, W.; Mohit, B.; Habash, N.; Obeid, O.; Tomeh, N.; Rozovskaya, A.; Farra, N.; Alkuhlani, S.; Oflazer, K. Large scale arabic error annotation: Guidelines and framework. In Proceedings of the 9th International Conference on Language Resources and Evaluation (LREC'14), Reykjavik, Iceland, 26–31 May 2014; pp. 2362–2369.
28. Ahmadi, S.; Le Roux, J.; Tomeh, N. Attention-Based Encoder-Decoder Networks for Spelling and Grammatical Error Correction. Master's Thesis, Paris Descartes University, Paris, France, 2017.
29. Watson, D.; Zalmout, N.; Habash, N. Utilizing character and word embeddings for text normalization with sequence-to-sequence models. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, 31 October–4 November 2018; pp. 837–843.
30. Solyman, A.; Wang, Z.; Tao, Q. Proposed model for arabic grammar error correction based on convolutional neural network. In Proceedings of the 2019 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE), Khartoum, Sudan, 21–23 September 2019; pp. 1–6.
31. Madi, N.; Al-Khalifa, H. A Proposed arabic grammatical error detection tool based on deep learning. *Procedia Comput. Sci.* **2018**, *142*, 352–355. [CrossRef]
32. Aljindi, A.; Sakhawy, D.; AlSaleh, N.; AlAndas, F.; AlRuhaily, A.; AlSaraa, H.; AlHarbi, N. *Linguistic Error Detector-Saudi Press as a Sample*; Princess Noura Bint Abdul Rahman University, Al-Jazirah Publishing: Riyadh, Saudi Arabia, 2015.
33. Madi, N.; Al-Khalifa, H. A7'ta: Data on a monolingual Arabic parallel corpus for grammar checking. *Data Brief* **2019**, *22*, 237–240. [CrossRef] [PubMed]
34. Keras. Embedding Layers–Keras Documentation. Available online: https://keras.io/layers/embeddings/ (accessed on 6 December 2018).
35. Kingma, D.P.; Lei Ba, J. ADAM: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
36. Nesterov, Y.E. A method for solving the convex programming problem with convergence rate O(1/k^2). *Dokl. Akad. Nauk SSSR* **1983**, *269*, 543–547.
37. Keras. Optimizers–Keras Documentation. Available online: https://keras.io/optimizers/ (accessed on 8 December 2018).
38. tf.nn.weighted_cross_entropy_with_logits|TensorFlow. Available online: https://www.tensorflow.org/api_docs/python/tf/nn/weighted_cross_entropy_with_logits (accessed on 6 December 2018).
39. Chollet, F. *Deep Learning with Python*; Manning Publications Co.: Shelter Island, NY, USA, 2018.
40. Azmi, A.M.; Almutery, M.N.; Aboalsamh, H.A. Real-word errors in arabic texts: A better algorithm for detection and correction. *IEEE/ACM Trans. Audio Speech Lang. Process.* **2019**, *27*, 1308–1320. [CrossRef]
41. Yang, X.; Huang, K.; Zhang, R.; Goulermas, J.Y.; Hussain, A. A new two-layer mixture of factor analyzers with joint factor loading model for the classification of small dataset problems. *Neurocomputing* **2018**, *312*, 352–363. [CrossRef]
42. Behroozi-Khazaei, N.; Nasirahmadi, A. A neural network based model to analyze rice parboiling process with small dataset. *J. Food Sci. Technol.* **2017**, *54*, 2562–2569. [CrossRef]
43. Bertolaccini, L.; Solli, P.; Pardolesi, A.; Pasini, A. An overview of the use of artificial neural networks in lung cancer research. *J. Thorac. Dis.* **2017**, *9*, 924–931. [CrossRef]
44. Jiang, P.; Chen, J. Displacement prediction of landslide based on generalized regression neural networks with K-fold cross-validation. *Neurocomputing* **2016**, *198*, 40–47. [CrossRef]

45. Gambäck, B.; Sikdar, U.K.; Waseem, Z.; Chung, W.H.K.; Hovy, D.; Tetreault, J. Using convolutional neural networks to classify hate-speech. In Proceedings of the 1st Workshop on Abusive Language Online, Vancouver, BC, Canada, 4 August 2017; pp. 85–90. [CrossRef]
46. Ren, X.; Zhang, L.; Wei, D.; Shen, D.; Wang, Q. Brain MR image segmentation in small dataset with adversarial defense and task reorganization. *Intel. Tutoring Syst.* **2019**, 1–8. [CrossRef]
47. Yannakoudakis, H.; Rei, M.; Andersen, Ø.E.; Yuan, Z. Neural sequence-labelling models for grammatical error correction. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, Copenhagen, Denmark, 7–11 September 2017; pp. 2795–2806.
48. Liu, Y.; Zan, H.; Zhong, M.; Ma, H. Detecting simultaneously Chinese grammar errors based on a BiLSTM-CRF model. In Proceedings of the 5th Workshop on Natural Language Processing Techniques for Educational Applications, Association for Computational Linguistics (ACL), Melbourne, Australia, 19 July 2018; pp. 188–193.
49. Yuan, Z.; Stahlberg, F.; Rei, M.; Byrne, B.; Yannakoudakis, H. Neural and FST-based approaches to grammatical error correction. In Proceedings of the 14th Workshop on Innovative Use of NLP for Building Educational Applications, Florence, Italy, 2 August 2019; pp. 228–239.
50. Bell, S.; Yannakoudakis, H.; Rei, M. Context is key: Grammatical error detection with contextual word representations. In Proceedings of the 14th Workshop on Innovative Use of NLP for Building Educational Applications, Florence, Italy, 2 August 2019; pp. 103–115.
51. Mohit, B. QALB: Qatar Arabic language bank. *Qatar Found. Annu. Res. Forum Proc.* **2013**. [CrossRef]
52. Reimers, N.; Gurevych, I. Optimal hyperparameters for deep LSTM-networks for sequence labeling tasks. *arXiv* **2017**, arXiv:1707.06799.