Arabisc: Context-Sensitive Neural Spelling Checker

Yasmin Moslem, Rejwanul Haque and Andy Way
ADAPT Centre
School of Computing
Dublin City University
Dublin, Ireland
firstname.lastname@adaptcentre.ie

Abstract

Traditional statistical approaches to spelling correction usually consist of two consecutive processes – error detection and correction – and they are generally computationally intensive. Current state-of-the-art neural spelling correction models usually attempt to correct spelling errors directly over an entire sentence, which, as a consequence, lacks control of the process, e.g. they are prone to overcorrection. In recent years, recurrent neural networks (RNNs), in particular long short-term memory (LSTM) hidden units, have proven increasingly popular and powerful models for many natural language processing (NLP) problems. Accordingly, we made use of a bidirectional LSTM *language model (LM)* for our context-sensitive spelling detection and correction model which is shown to have much control over the correction process. While the use of LMs for spelling checking and correction is not new to this line of NLP research, our proposed approach makes better use of the rich neighbouring context, not only from before the word to be corrected, but also after it, via a *dual-input* deep LSTM network. Although in theory our proposed approach can be applied to any language, we carried out our experiments on Arabic, which we believe adds additional value given the fact that there are limited linguistic resources readily available in Arabic in comparison to many languages. Our experimental results demonstrate that the proposed methods are effective in both improving the quality of correction suggestions and minimising overcorrection.

1 Introduction

Misspelling detection or/and correction modules are seen as critical components of many real-world NLP applications. This has also been regarded as an important research area of NLP for years. The spelling errors are broadly classified into two categories: non-word errors (NWE), and real-word errors (RWE). If the misspelled string is a valid word of a language, it is called an RWE, otherwise it is an NWE (Choudhury et al., 2007). In this context, Peterson (1986) found that the RWE rate ranges from 2% for a small lexicon to 10% for a 50,000-word lexicon and almost 16% for a 350,000-word lexicon. In this work, we investigate both error types (i.e. RWE and NWE) with our context-aware spelling error detection and correction models. We demonstrate that our approach is capable of detecting and correcting both NWEs and RWEs in a text. As an illustration, we present two sentences that contain misspelled words below, with a justification of why context-sensitive error detection and correction could be an ideal solution for this problem.

English:

- **Wrong:** Students met their *Principle* Supervisor at the University.
- **Correct:** Students met their *Principal* Supervisor at the University.

Arabic:

- **Wrong:** هذه هي الطريق المثلي التي يجب أن نتخذها
- **Correct:** هذه هي الطريق المثلى التي يجب أن نتخذها

In the first English sentence, we can see that the word *Principle* is a correct word that we can find in a dictionary; however, its use in this context is incorrect and the right word in this context is to be *Principal*. Hence, we can call this an RWE, and we can clearly see that this requires help from the neighbouring lexical contexts for error detection and correction. Similarly, in the Arabic example, we can see that the adjective المثلي (*almthly*) was incorrectly used instead of المثلى (*almthla*) to describe الطريق (*altariq*). Like the error in the English example, this error requires the same treatment.

Proceedings of the 6th Workshop on Natural Language Processing Techniques for Educational Applications, pages 11–19
Suzhou, China, December 4, 2020.

Traditional rule-based and statistical approaches to spelling correction rely on error detection first before offering correction suggestions. This minimises the chances of making unrequired corrections at least for common words. However, creating a good spelling checker using such traditional approaches involves building a large lexical database and thousands of human-generated rules for NWEs, or large phrase tables for RWEs (Verberne, 2002). This, in effect, requires a lot of linguistic resources and tools as well as massive computing resources.

Many neural approaches (Weiss, 2016) to spelling checking normally correct errors directly over an entire input sentence. Presenting an entire sentence to the network or decoder for correction involves the risk of modifying words that are correct in the context and should not be changed. For instance, the experiments carried out by Weiss (2016) demonstrate how neural spelling checking models can make overcorrection mistakes with examples. They categorise such errors as follows (Q: input; A: ground truth; S: system output):

1- Correcting words that are not really misspellings:

- Q. In addition to personal-injury and
- A. In addition to personal-injury and
- S. In addition to personal injury and

2- Changing the original meaning:

- Q. had learned of Ca secret plan y Iran
- A. had learned of a secret plan by Iran
- S. had learned of a secret plan I ran

3- Even introducing new misspellings:

- Q. post-Thanksgiving performances, but
- A. post-Thanksgiving performances, but
- S. post-thanks gving performances, but

As can be seen from these examples, the neural model corrects some words that should not be corrected. We conjecture that this happened because the model tries to make correction directly on the entire sentence while bypassing the error detection process. In this context, Hertel (2019) found that neural *many-to-many* encoder-decoder models for spelling correction perform worse than neural *many-to-one* LM-based approaches. What if we rather ask the neural network to first "detect" the error and then "correct" it, with the help of language modelling while still taking the context into consideration? This is the research question we explore in this paper.

In this work, we propose a context-sensitive neural model, *Arabisc*,[1] which adds more control to the spelling correction process using language modelling, i.e. a many-to-one LSTM network, and it consists of two processes: (i) identifying spelling errors, and (ii) offering correction suggestions. The idea is that we have to only correct the mistakes not the whole sentence. In other words, we combine the best of two worlds (statistical[2] and neural) i.e. we detect potential spelling mistakes and then offer diverse correction suggestions for the user to choose from in one go.[3] Although we tested our method on standard Arabic (*Fosha*), it can theoretically be applied to any other language.

The rest of this paper is organised as follows. Section 2 elaborates on our methodology including the architecture of our proposed model. Section 3 describes the experimental results and findings with some discussions, while Section 4 concludes and suggests some avenues for future work.

2 Methodology

The backbone of our approach involves building and using language modelling, i.e. a *many-to-one* LM for text generation. The task is to check a given input sentence *word by word*, predict the next word, and to find out whether the current word cw of the input sentence is in the list of high-scoring candidates B generated by the LM given the context of cw (previous and following words of cw). If cw is not in the list, correction suggestions are offered based on the edit distance (Levenshtein, 1966) between cw and the candidates in B. In our work, we compare two different models, namely: (i) a single-input model that uses only the preceding words of cw as context, and (ii) a dual-input model that uses the preceding and following words of cw as context. We describe our models in detail in the following section.

2.1 Experimental Setups

2.1.1 Training Data

In order to build an LM to be used in the spelling correction task, it is important to make sure that

[1] *Arabisc* is a common misspelling of the word *Arabesque*, which refers to a form of artistic decoration. Surprisingly, *Arabisc* (or *Arabiśc*) is a real word from old English and it means Arabic or an Arab. Wikipedia: https://en.wiktionary.org/wiki/Arabisc

[2] Neural networks are statistical models. In this paper, we use "statistical" to refer to those models that do not have neural components.

[3] In our implementation, suggestions are generated as a JSON object, which can be used to display correction options to users, i.e. via a GUI.

sentences in our training set are linguistically correct and do not have many spelling mistakes. We selected the News Commentary Corpus v11[4] from OPUS (Tiedemann, 2012) as it is a reasonably clean corpus. We applied the standard filtering and pre-processing steps to the corpus. We are left with 213,036 Arabic sentences after cleaning and pre-processing. We also added a portion of the MultiUN[5] corpus from OPUS to the News Commentary corpus. Our final training data contains 554,622 Arabic sentences. The MultiUN corpus is of a better linguistic quality and the News Commentary corpus is more generic in nature. Therefore, we believe that adding the MultiUN corpus to the News Commentary corpus enriches our training data vocabulary. In order to pre-process the training sentences, we applied the following steps:

- Split those lines that consist of multiple segments based on newline, period followed by a space or a newline, Arabic question mark "؟", and exclamation mark;
- Remove duplicate segments;
- Remove Arabic diacritics, mainly *Tashkil* (marks used as phonetic guides);
- Remove punctuation marks and numbers. Some spelling checkers would keep punctuation marks and even correct them; but for the purpose of our experiments, we chose to remove them;
- Remove Latin characters;
- Append a start token <s> at the beginning;
- In order to avoid repetitions after applying the next step, truncate the sentences up to the maximum sequence length; and
- For our single-input encoder (cf. Section 2.2.1), generate *n*-gram sequences, using all preceding tokens as the context except the current token (cw) which is used as the label. Tables 1 and 2 illustrate the *n*-gram generation process. As for our dual-input encoder (cf. Section 2.2.2), in addition to the preceding tokens, include the remaining tokens after the label (cw) as the context, in *reverse* order, as the second contextual input. The *n*-gram generation process of the latter setup is illustrated in Tables 3 and 4.

[4] http://opus.nlpl.eu/News-Commentary.php

[5] http://opus.nlpl.eu/MultiUN.php

Input Sentence	
<s> students met their principle supervisor at the university	
Initial Sequence	**Current Word**
<s>	students
<s> students	met
<s> students met	their
<s> students met their	principle
<s> students met their principle	supervisor
<s> students met their principle supervisor	at
<s> students met their principle supervisor at	the
<s> students met their principle supervisor at the	university

Table 1: Single-input *n*-gram splitting of an English sentence.

Input Sentence	
<s> هذه هي الطريق المثلي التي يجب أن تتخذها	
Initial Sequence	**Current Word**
<s>	هذه
<s> هذه	هي
<s> هذه هي	الطريق
<s> هذه هي الطريق	المثلي
<s> هذه هي الطريق المثلي	التي
<s> هذه هي الطريق المثلي التي	يجب
<s> هذه هي الطريق المثلي التي يجب	أن
<s> هذه هي الطريق المثلي التي يجب أن	تتخذها

Table 2: Single-input *n*-gram splitting of an Arabic sentence.

2.1.2 Evaluation Test Set

In order to evaluate *Arabisc*, our spelling correction model, we randomly extracted 20 Arabic unseen sentences from the UN corpus.[6] From now on, we refer to this set of sentences as the evaluation test set. We introduced two types of errors in our evaluation test set: (i) the first set contains RWEs based on the confusion lists provided by Al-Jefri and Mahmoud (2013), and (ii) the second set contains NWEs based on deletion, insertion, substitution and transposition of adjacent alphabets in a word, being the causes of most spelling errors (Damerau, 1964). In our experiments, we used a development set which has helped us explore potential issues in relation to Arabic spelling checking and correction and fine-tune hyper-parameters.

Each sentence of the test set was pre-processed the way we prepared the training corpus (cf. Section 2.1.1). We split each test set sentence into a list of initial *n*-gram sequences and use the last word as the current word (cw) that we want to compare with the high-scoring next-word candidates *B* generated by the LM. Tables 1 and 2 demonstrate the *n*-gram generation process for the single-input decoder. As for the multiple input decoder, we provide the model with two sets of input tokens, i.e. tokens before and after the current word (cw) to be checked, and the feature generation process is shown in Tables 3 and 4.

[6] http://opus.nlpl.eu/UN.php

2.2 Arabisc

2.2.1 Single-Input Encoder

Our *many-to-one* spelling correction model is an RNN (Rumelhart et al., 1986; Werbos, 1990) with LSTM units (Hochreiter and Schmidhuber, 1997). The total number of layers in the network is 4. We use an embedding layer with an input dimension 256 and then add two hidden layers, one bidirectional LSTM with 512 units followed by an LSTM with 128 units. The output layer is a *Dense* layer with the softmax activation function, and the number of units in this layer is equal to the vocabulary size. The model is trained with the *Adam* optimizer (Kingma and Ba, 2015), with the learning-rate set to 0.001. The sparse categorical cross-entropy is used as the loss function. As mentioned earlier, we limit the maximum sequence length to 15 tokens.[7] The vocabulary size is set to 100,000 of the most frequently occurring tokens in the corpus. The encoder takes an input in the form of *n*-gram sequences generated by the training example creation module described in Section 2.1.1. For building our network, we used Keras Sequential API of TensorFlow 2.[8] The model was trained on 2 GeForce RTX 2080 TI GPUs for 8 epochs. Early stopping was used on the validation accuracy. In this setup, we found that the training loss was 4.88 and training accuracy was 0.26.

2.2.2 Dual-Input Encoder

We start this section by revisiting the example sentence *"Students met their Principal Supervisor at the University,"* and the list of conditional contexts (i.e. *n*-grams) shown in Table 3. We can see from the table that the word *"Principal"* is affected by words before it (e.g. *"Students"*) and words after it (e.g. *"Supervisor"* and *"University"*). Therefore, using a dual-input encoder that takes both the preceding and following contexts into account can be more appropriate as far as the spelling error detection and correction are concerned. Note that our dual-input encoder is similar in terms of its architecture to the single-input encoder. The only difference is that the conditional context of the word (cw) to be predicted comprises two inputs: the tokens ($[w_1, w_2 ... w_{n-1}]$) that come before the current word cw, and the tokens ($[w_{n+1}, w_{n+2} ...]$) that come after the current word cw in *reverse* order. To exemplify, for the aforementioned sentence, we will have:

Left-Branch Input: <s> → students → met → their

Current Word: → principal ←

Right-Branch Input: supervisor ← at ← the ← university

As we can see above, both the preceding and following parts of the input sequence are used as the conditional context by the neural network for the prediction of the token in between. We apply the same step to all tokens to be predicted. We conducted experiments by both keeping and reversing the order of tokens in the right-branch input, and found that the model with reversing the tokens that follow the current word cw beforehand works best in terms of the validation and test set accuracy. Note that Section 2.1.1 describes the details of pre-processing the input data.

In this setup, we used Keras Functional API of TensorFlow 2, that allows multiple inputs. The identical four layers described in Section 2.2.1 are used for each of the two inputs. Finally, the two output layers are merged together using a *Concatenate* layer to generate the final (single) output using a *Dense* layer. Figure 1 illustrates the right and left branches of our dual-input neural network. Like the single-input encoder, the dual-input model was trained on 2 GeForce RTX 2080 TI GPUs for 11 epochs. Early stopping was used on the validation accuracy. In this setup, we found that the training loss was 3.15 and training accuracy was 0.46.

2.2.3 Bidirectional LSTM versus Dual-Input Encoder

In Section 2.2.1, we pointed out that the Single-Input Encoder uses a *Bidirectional LSTM* layer. If we express this in a different way, the bidirectional effect is applied only up to the word that is currently being generated, i.e. the "Left-Branch Input" in the aforementioned example. As for the Dual-Input Encoder described in Section 2.2.2, in addition to the "Left-Branch Input", it uses the "Right-Branch Input" which plays a pivotal role in observing the wider context and improving the quality of spelling corrections. This subtlety differentiates fundamental single-input language modelling from the encoder-decoder architecture, as the former takes only words before the current word to be generated while the latter deals with the sentence as a whole. As pointed out earlier, using *many-to-one* text generation with LMs for spelling correction tasks brings about better quality over using *many-to-many* encoder-decoder architectures (Hertel, 2019). Hence, we chose to use language modelling to have more control over the correction process, word by word, while we propose to use the Dual-Input Encoder to solve this limitation. While

[7] We restricted the length to 15 as processing longer sentences is found to be computationally expensive.

[8] https://github.com/tensorflow/tensorflow

Input Sentence		
<s> students met their principle supervisor at the university		
1st Input Sequence	**Current Word**	**2nd Input Sequence (in reverse order)**
<s>	students	university the at supervisor principle their met
<s> students	met	university the at supervisor principle their
<s> students met	their	university the at supervisor principle
<s> students met their	principle	university the at supervisor
<s> students met their principle	supervisor	university the at
<s> students met their principle supervisor	at	university the
<s> students met their principle supervisor at	the	university
<s> students met their principle supervisor at the	university	

Table 3: Dual-input *n*-gram splitting of an English sentence.

Input Sentence		
<s> هذه هي الطريق المثلي التي يجب أن نتخذها		
Initial Sequence	**Current Word**	**2nd Input Sequence (in reverse order)**
<s>	هذه	نتخذها أن يجب التي المثلي الطريق هي
<s> هذه	هي	نتخذها أن يجب التي المثلي الطريق
<s> هذه هي	الطريق	نتخذها أن يجب التي المثلي
<s> هذه هي الطريق	المثلي	نتخذها أن يجب التي
<s> هذه هي الطريق المثلي	التي	نتخذها أن يجب
<s> هذه هي الطريق المثلي التي	يجب	نتخذها أن
<s> هذه هي الطريق المثلي التي يجب	أن	نتخذها
<s> هذه هي الطريق المثلي التي يجب أن	نتخذها	

Table 4: Dual-input *n*-gram splitting of an Arabic input sentence.

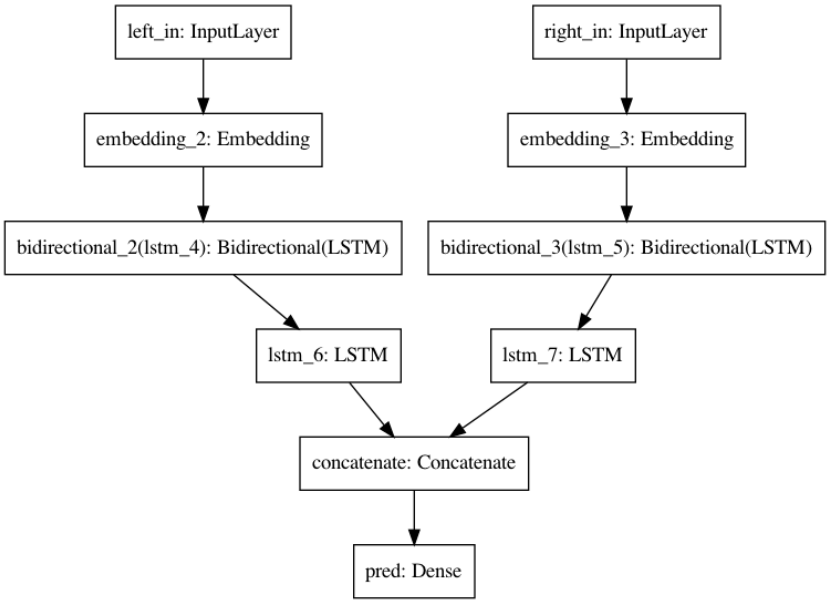

Figure 1: Dual-Input Encoder

our solution is simple, we believe its novelty lies in adding more context to the regular *many-to-one* language modelling process, which is also reflected in our results (cf. Section 3).

2.3 Inference

In the decoding process, the *many-to-one* LSTM network takes each item from the list of *n*-gram sequences generated from the input sentence (cf. Section 2.1.2) as input and predicts the next (or current) word cw. For our dual-input model, this means that we use two inputs, words before the current word $LeftW$ and words after it $RightW$. LMs are normally utilised for text generation to predict the next token or next few tokens in a sequence given the preceding tokens as context (Santhanam, 2020). Similarly, our neural network greedily decodes to search for the most likely sequences. However, in our case, instead of keeping only the 1-best candidate, we keep the *n*-best candidates B and then calculate the edit distance ed between each candidate b and the current word cw. We observed that *n* for the *n*-best list is a sensitive hyper-parameter, i.e. when we increase the size of this hyper-parameter, we obtain a better vocabulary coverage and more suggestions at the expense of many less probable candidates. In this case, the decoder may choose an incorrect word as a possible suggestion. Therefore, the value of *n* of the *n*-best list (B) is a kind of trade-off. There are three possible cases:

1. $ed = 0$: this indicates that the current word cw is found in the *n*-best candidate list B and it is likely that cw is a correct word;

2. $ed > 0$ and $ed <= 2$: this indicates that there are other suggestions for the current position in B. If the current word cw is not found at all in B or found but after several suggestions (e.g. 10), there are chances that for the current context one of these suggestions is better than cw. We also take the length of the current word cw into consideration. If the length of $cw <= 3$, we stick to $ed = 1$, and if the length of $cw > 3$, we allow $ed <= 2$. Since we have a large pool of suggestions, our current decoder uses greedy search in order to find the item in B and calculate the edit distance measure. We empirically found that this setup worked best in our case. However, in order to obtain a list of better suggestions, beam search or bidirectional beam search (Sun et al., 2017) can be applied, which has been kept for our future work;

3. if neither the current word cw nor any similar candidates are found in the *n*-best candidate list B, no output is offered.

2.3.1 Out-of-Vocabulary Tokens

There is a known limitation of neural networks, i.e. they typically operate with a fixed vocabulary. As for a more complex task such as neural machine translation (Vaswani et al., 2017), sub-word segmentation techniques such as Byte Pair Encoding (Sennrich et al., 2016) or using a unigram language model (Kudo, 2018) are usually utilised in order to solve this problem. Since we calculate the edit distance measure on tokens, it is difficult to apply sub-word segmentation or similar techniques to this problem. As far as the spelling checking is concerned, the presence of out-of-vocabulary tokens in the input sentence may cause overcorrection at decoding because they will not come as suggestions in the *n*-best list (B). In order to solve this problem, we adopted two strategies:

- handling out-of-vocabulary tokens *on-the-fly*: lemmatising long words (consisting of more than 7 characters) and comparing different lemmas to probable suggestions at the decoding time; and

- fixing the previous misspelled word before predicting cw.

We present the pseudocode of the decoding process in Algorithm 1.

Algorithm 1: Spelling Checker Algorithm

```
   // For each current word
1  for cw=1 ... CW do
       // Predict the most likely
          sequences based on the left and
          right sequences
2      B = Predict([LeftW, RightW])
3      S = []          // List of suggestions.

       // For each candidate in the n-best
          candidates B generated by the
          LM for the current word cw
4      for b=1 ... B do
           // Calculate Edit Distance ed
              between cw and b
5          if EditDistance(cw, b) = 0 then
6              break        // Word is correct
7          else if Length(cw) <= 3 AND
             EditDistance(cw, b) = 1 then
8              Add b to S
9          else if Length(cw) > 3 AND
             EditDistance(cw, b) <= 2 then
10             Add b to S
11         else
12             continue
13     Return S
```

3 Results and Discussions

This section presents the results obtained along with our findings and some discussion.

3.1 Real Word Errors (RWE)

To the best of our knowledge, there is no freely available tool that supports context-sensitive spell checking for Arabic as far as RWEs are concerned. Hence, we could not compare our proposed models with other existing spelling correction models. We obtained results to evaluate our both single-input and dual-input models on the evaluation test set, and they are reported in Table 5. Note that existing *many-to-one* spelling correction models that use LMs to detect misspelled words are in fact based on a single-input architecture, i.e. tokens before the word to be corrected used as a conditional context for correction. As mentioned earlier, our dual-input encoder takes both the preceding and following tokens in *reverse* order as the conditional context for spelling checking and correction. We see from Table 5 that both our models correctly detect the same number of RWEs. However, we can clearly see from the table that the dual-input model outperforms the single-input model in terms of the quality of suggestions and minimisation of overcorrection. We also see from Table 5 that the two strategies explained in Section 2.3.1 (i.e. comparing lemmatised variants of tokens and correcting previous words before predicting the next word) were effective in handling out-of-vocabulary words and helped minimise overcorrection.

We believe that the success of our context-sensitive approach, especially our dual-input encoding model, lies in its ability to detect RWEs regardless of the location of the word in the sentence because it takes both sides of the sentence into account for correction.

3.2 Non-Word Errors (NWE)

This section presents our results for NWEs. In this case, in addition to our single-input and dual-input models, we considered two popular Arabic spelling checkers: *LanguageTool*[9] and *Sakhr Tadqeek*.[10] We report the results obtained in Table 6. We see from the table that our dual-input model outperforms all other models in terms of quality of suggestions and minimisation of overcorrection. We also see that our models outperform *LanguageTool* and *Sakhr Tadqeek* even in terms of detecting wrong words. Additionally, we observed that while *LanguageTool* and *Sakhr Tadqeek* consider some barely-used outdated words as correct, our model detects them as potential spelling mistakes and suggests good corrections.

3.3 Prediction Examples

As mentioned in Section 2.3.1, we lemmatised those words which contain more than seven characters in order to minimise the data sparsity problem. For example, with this approach, we avoided the detection of الفعالية (*alfaaleya*) as a mistake by comparing it to other words of the same lemma such as والفعالية (*walfaaleya*) and بفعالية (*befaaleya*). Similarly, the word المصرفية (*almasrefeya*) was compared to المصرفي (*almasrefey*). We show the results obtained by applying this lemmatisation strategy to our dual-input model in Tables 5 and 6 (cf. row "*Dual-Input+Lemma*").

One of the possible ways to improve correction suggestions and avoid overcorrection is to correct the previous word (if it is a misspelled item) before predicting the next word. We observed that the collaboration of the two strategies (i.e. lemmatisation and correcting the preceding misspelled word) leads us to the best spelling detection and correction model; the evaluation scores of the best model on the test set are shown in the last row of Table 5. Note that we refer to the system that applies the first approach (correcting the previous word) as "*Dual-Input+Prev*" and the collaborative method as "*Dual-Input+Lemma+Prev*".

The last two rows of Table 5 represent the results obtained using *LanguageTool* and *Sakhr Tadqeek*. Although both tools were able to detect most NWEs, they failed to detect مواصله (*muwaseleh*) as a mistake for مواصلة (*muwaselet*).[11] This example clearly shows how such spell-checking tools may consider barely-used outdated words as real words, which are in fact spelling mistakes in the context. When it comes to correction suggestions, both *LanguageTool* and *Sakhr Tadqeek* failed to offer exact suggestions or similar alternatives for some NWEs. For example, both tools could not correct the word معدلت (*mueddelt*) as معدلات (*mueddelat*); instead, they offered words like معدلة (*mueddelet*) or معدات (*mueddat*), and *LanguageTool* offered a similar alternative معدل (*mueddel*) which is the singular form of the original word.

[9] https://languagetoolplus.com/

[10] https://tadqeek.alsharekh.org/

[11] The error comes from the letter ه which should be ة.

Model	Detected	Exact Suggestion	Similar Suggestion	Over-Correction
Single-Input	20	17	0	9
Dual-Input	20	20	N/A	5
Dual-Input+Lemma	20	20	N/A	3
Dual-Input+Prev	20	20	N/A	3
Dual-Input+Lemma+Prev	**20**	**20**	**N/A**	**1**

Table 5: Results for RWEs. The first column "Detected" represents the percentage of wrong words marked as wrong. The second column refers to the percentage of those words that are exactly found in the suggestions. The third column shows the percentage of those words whose suggestions do not include the original word but acceptable alternatives. The last column is for words marked as incorrect as they are not among the *n*-best tokens. Rows 3 and 4 represent the use of lemmatisation and previous word correction individually while row 5 shows the results of applying both methods to the dual-input model.

Model	Detected	Exact Suggestion	Similar Suggestion	Over-Correction
Single-Input	20	19	0	9
Dual-Input	20	18	1	3
Dual-Input+Lemma	20	18	1	1
Dual-Input+Prev	**20**	**19**	**0**	**0**
LanguageTool	19	11	1	3
Sakhr Tadqeek	18	14	0	0

Table 6: Results for NWEs. The first column "Detected" represents the percentage of wrong words marked as wrong. The second column refers to the percentage of those words that got the exact original word among the suggestions. The third column shows the percentage of those words whose suggestions did not include the original word but the acceptable alternatives. The last column is for words marked as incorrect as they are not among the *n*-best tokens. Rows 3 and 4 represent the use of lemmatisation and previous word correction, respectively. The last two rows show results from LanguageTool and Sakhr Tadqeek.

4 Conclusion

In this paper, we presented a deep *many-to-one* neural network-based context-sensitive spelling checking and correction model. In short, we modelled words that come both before and after the word to be corrected as the conditional context in language model predictions. The experimental results suggest that our approach has achieved considerable success in terms of both offering better correction suggestions and minimising overcorrection. Our project, *Arabisc*, code, spelling correction models and data sets are now available as an open-source project via an open repository.[12]

In the future, we plan to increase the training data size to see how our models will perform on a large-scale data set and more languages other than Arabic. The state-of-the-art bidirectional encoder representation from transformers (BERT) architecture (Devlin et al., 2018) makes use of Transformer (Vaswani et al., 2017), an attention mechanism that learns contextual relations between words in a text and can offer powerful masked language modelling. As an alternative to our LSTM LMs, we plan to investigate using BERT masked language models in *Arabisc*. We evaluated our models on a test set that contains a small number of examples. In the future, we plan to increase the size of test set examples. Currently, our models operate at word level for spell-checking and correction. This could be an issue while encountering the out-of-vocabulary items. In the future, we aim to investigate applying byte-pair encoding (Sennrich et al., 2016) or similar word-segmentation technique in our model.

[12] https://github.com/ymoslem/Arabisc

Acknowledgments

The ADAPT Centre for Digital Content Technology is funded under the Science Foundation Ireland (SFI) Research Centres Programme (Grant No. 13/RC/2106) and is co-funded under the European Regional Development Fund. The publication has emanated from research supported in part by research grants from SFI and Microsoft under Grant Numbers 13/RC/2077 and 18/CRT/6224.

References

Majed Al-Jefri and Sabri Mahmoud. 2013. Context-Sensitive Arabic Spell Checker Using Context Words and N-Gram Language Models. In *2013 Taibah University International Conference on Advances in Information Technology for the Holy Quran and Its Sciences, Madinah, Saudi Arabia*, pages 258–263.

Monojit Choudhury, Markose Thomas, Animesh Mukherjee, Anupam Basu, and Niloy Ganguly. 2007. How Difficult is it to Develop a Perfect Spell-checker? A Cross-Linguistic Analysis through Com-

plex Network Approach. In *Proceedings of the Second Workshop on TextGraphs: Graph-Based Algorithms for Natural Language Processing*, pages 81–88, Rochester, NY, USA. Association for Computational Linguistics.

Fred J. Damerau. 1964. A Technique for Computer Detection and Correction of Spelling Errors. *Commun. ACM*, 7(3):171–176.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR*, abs/1810.04805.

Matthias Hertel. 2019. *Neural Language Models for Spelling Correction*. Master's thesis, Albert-Ludwigs-Universität Freiburg im Breisgau Technische Fakultät, Freiburg, Germany.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Taku Kudo. 2018. Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates. *CoRR*, abs/1804.10959.

Vladimir I. Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics-Doklady, Vol. 10*, pages 707–710.

James L. Peterson. 1986. A Note on Undetected Typing Errors. *Commun. ACM*, 29(7):633–637.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1986. Learning representations by back-propagating errors. *Nature*, 323(6088):533.

Sivasurya Santhanam. 2020. Context based Text-generation using LSTM networks.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Qing Sun, Stefan Lee, and Dhruv Batra. 2017. Bidirectional Beam Search: Forward-Backward Inference in Neural Sequence Models for Fill-in-the-Blank Image Captioning. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, Hawaii, USA*, pages 1339–1348.

Jörg Tiedemann. 2012. Parallel Data, Tools and Interfaces in OPUS. In *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC'2012)*, pages 2214–2218, Istanbul, Turkey.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 6000–6010.

Suzan Verberne. 2002. *Context-sensitive Spell Checking Based on Word Trigram Probabilities*. Master thesis Taal, Spraak & Informatica, University of Nijmegen, Nijmegen, the Netherlands.

Tal Weiss. 2016. Deep Spelling, Rethinking Spelling Correction in the 21st Century. machinelearnings.co, accessed September 10, 2020.

Paul J Werbos. 1990. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560.