# Correcting Arabic Soft Spelling Mistakes Using Transformers

Mohammed Al-Qaraghuli
*Computer Engineering Dept.*
*The University of Jordan*
Amman, Jordan
mhm8180990@ju.edu.jo

Gheith Abandah
*Computer Engineering Dept.*
*The University of Jordan*
Amman, Jordan
abandah@ju.edu.jo

Ashraf Suyyagh
*Computer Engineering Dept.*
*The University of Jordan*
Amman, Jordan
a.suyyagh@ju.edu.jo

***Abstract*—Spelling mistakes are a common issue in the Arabic language; several techniques have been used to solve this issue such as confusion matrices, language models, and neural networks. In recent years, a neural network called the *Transformer* has been introduced as a machine translation model. Since then, the transformer and its variants has become a very popular solution for most of the Natural Language Processing (NLP) tasks. In this paper, we aim to use the transformer to correct four types of Arabic soft spelling mistakes, namely, confusion among various shapes of *hamza*, shapes of *alef* at the end of the word, insertion and omission of *alef* after *waw aljamaea*, and confusion among *teh*, *teh marbuta*, and *heh* at the end of the word. We used artificial errors for training and evaluation; these errors were generated using an approach called stochastic error injection. The best model we trained was able to correct 97.37% of the artificial errors that were injected into the test set and achieved a character error rate (CER) of 0.86% on a set that contains real soft spelling mistakes.**

***Keywords—spelling correction, transformers, Arabic, soft spelling mistakes, NLP***

## I. Introduction

Arabic is the fifth most spoken language in the world, with more than 400 million speakers [1]. Like other languages, spelling mistakes are a common issue in Arabic; these mistakes can be categorized into four types: discourse structure and pragmatic-level errors, morpho-syntactic errors, lexical and semantic errors, and soft errors [2].

### A. Discourse Structure and Pragmatic-level Errors

Discourse structure errors occur when a single or multiple words with correct form and spelling is used in a sentence but is not compatible with the sentence meaning. Some words may have various meanings depending on the context of the sentence if the word meaning is misunderstood within a certain context, then the error is considered as a pragmatic. We show this error type with the following translation example:

The sentence "*The Biology test was piece of cake. I think I will pass*" has the Arabic translation: "كان اختبار علم الأحياء عبارة عن قطعة من الكعكة ، أعتقد أنني سأنجح". The "piece of cake" phrase is literally translated to "قطعة من الكيك" the literal Arabic translation for this phrase is out of context for it has replaced the English phrase word by word to refer to an actual piece of cake, whereas the proper translation should have used the Arabic equivalent for the word easy. So, this is a pragmatic error. In the same example, the misuse of the diacritization for the word "علم" by using *fatha* instead of *kasra* changes its meaning from science to flag.

### B. Morpho-syntactic Errors

This type of errors occurs due to the confusion between the feminine and masculine forms, using the wrong tense for a verb, or wrong positioning of a word. An example for this type of errors is the sentence "اشتريتُ خمس كتب", the word "خمس" should be written in the feminine form "خمسة".

### C. Lexical and Semantic Errors

This type of errors occurs due to typographical mistakes; for example, writing the word based on its phonetics rather than its typographic form. These mistakes generate words that either do not belong to the lexicon (lexical errors), or words that are not suitable for the sentence (semantic errors). For example, the sentence "يعتبر الظفدع من البرمائيات" has a lexical error in the word "ظفدع" as it is not a real word, for the letter "ض" is phonetically confused with the letter "ظ" whereas the correct word should be "ضفدع".

### D. Soft Errors

This type of errors mainly occurs due to the confusion among various shapes of a certain letter. Letters such as *al-hamza* (ء), *alef* (ا), *teh* (ت), and *heh* (ه) have various shapes depending on the position of the letter or certain rules. For example, in the word "تفاؤل", the *hamza* is written on a *waw* "ؤ" when the word is a noun, but if it is verb, the *hamza* will be written alone "تفاءل".

The aim of this paper is to correct four types of common soft spelling mistakes using the transformer model, namely, the confusion among *hamza* various shapes, the various shapes of *alef* at the end of the word, the insertion and omission of *alef* after *waw aljamaea*, and the confusion among *teh*, *teh marbuta*, and *heh* at the end of the word.

We propose a machine learning approach of a transformer neural network that achieves excellent accuracy in correcting Arabic soft spelling mistakes. We built the transformer model from scratch to correct these mistakes at the character-level.

The rest of the paper is structured as follows: in Section II, we review the previous works that deal with spelling correction for both Arabic and other languages. In Section III, we describe the model structure and the datasets that we used. In Section IV, we present the experiments that we conducted and show our results, and lastly, in Section V, we provide the conclusions and ideas for future work.

## II. Related Work

Spelling correction techniques have been evolving through the years; various methods have been used such as confusion matrices, language models, neural networks, and long short-term memory (LSTM) recurrent neural networks. These techniques were used to improve the performance of systems such as optical character recognition (OCR) and automatic speech recognition (ASR).

### A. Spelling Correction for the Arabic Language

Most Arabic spelling correction works use non-machine learning techniques and few of them use machine learning,

specifically BiLSTM. Abandah *et al.* introduced two character-level BiLSTM models to correct Arabic soft spelling mistakes identical to the ones we target in this paper [2]. They trained the models on Tashkeela and ATB3 sets; the authors implemented two training approaches: transformed input and stochastic error injection. Transformed input was designed to map a set of targeted letters into a one-letter form. Stochastic error injection was used to replace a letter from a certain set with another letter from the same set based on a parameter called error injection rate *p*. The two models were trained using both approaches and a third set called *test200* that contains a real soft spelling mistakes was used for evaluation. The best model was the one that was trained using the Tashkeela set with stochastic error injection approach and 40% error injection rate; this model obtained a character error rate (CER) of 1.28%.

Alkhatib *et al.* introduced word-level model with BiLSTM and polynomial network classifier for Arabic spelling and grammatical mistakes detection and correction [3]. The authors trained the model using artificial errors generated using linguistic knowledge algorithms. For testing, they used a set that contains 15 million words with errors. These errors were manually inserted and corrected by experts. The authors report an F-measure of 93.89%.

Azmi *et al.* introduced a system to detect and correct Arabic semantic errors [4]. The system was trained by inserting one artificial error in the sentence then a support vector machine classifier (SVM) detects if the word contains an error or not, if it has an error the N-gram language model generates a candidate word to replace the word with error. The system achieved an $F_1$ score of 90.7%.

Several non-machine learning techniques have been used in many previous works such as N-gram language model in [5], confusion matrix and noisy channel spelling correction model in [6], Levenshtein distance algorithm in [7], and a hybrid system that combine more than one technique in [8].

### B. *Spelling Correction for Other Languages*

Most recent works in other languages started using transformers for spelling correction. Tran *et al.* introduced a transformer model to detect and correct Vietnamese spelling mistakes [9]. The authors built their model with character-level transformer encoder with four self-attention layers, word-level transformer encoder with 12 self-attention layers, and a detector and corrector classifier. The model achieved an $F_1$ score of 68.88% for detection, and a correction accuracy of 96.01%.

Kuznetsov and Urdiales introduced a transformer model for spelling correction [10]. The model was trained with a method that generates artificial spelling errors for any input language. The authors built this method by collecting data from search engines and applying a set of rules to generate errors. Their model was tested on four languages and has an accuracy of 83.33% for Arabic, 93.97% for Greek, 91.83% for Russian, and 94.48% for Setswana.

Zhao *et al.* introduced a transformer model for semantic errors correction in Mandarin automatic speech recognition system [11]. The authors used bidirectional and auto-regressive transformer (BART) model to correct the output of the ASR system and improve its performance. The ASR system that integrated with their model achieved a CER of 6.08%.

Zhang *et al.* introduced a new method called soft-masked BERT to detect and correct Chinese spelling mistakes [12]. The detection stage works by implementing bidirectional gated recurrent unit network (BiGRU) to detect the errors in the input. The correction stage uses bidirectional encoder representations from transformers (BERT); BERT is a transformer model that consists of 12 bidirectional encoders. The model achieved an $F_1$ score of 73.5% for detection and 66.4% for correction.

## III. Proposed Models and Experimental Setup

We built our model based on the architecture of the original transformer [13] shown in Fig. 1.

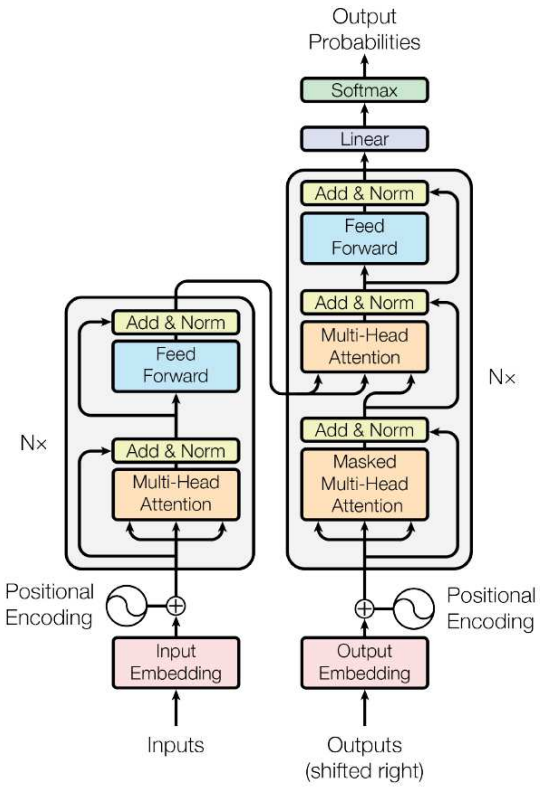


Fig. 1. Transformer architecture [13]

The transformer model uses self-attention to capture the relations among the input tokens; it also helps the model to find the best representation for a certain token. We illustrate the concept of self-attention in Fig. 2. One can observe that "it" in the left-side sentence is referring to the word "animal" due to the presence of the word "tired". The right-side sentence contains the word "wide" instead of the word "tired", thus, the best representation for "it" is the word "street".

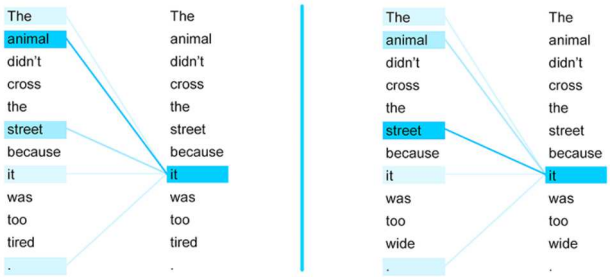


Fig. 2. Example of self-attention [14]

### A. Model Configurations

The default transformer has encoder-decoder stacks with number of layers $N = 6$, self-attention layers or heads $h = 8$, pointwise feed-forward network with dimension $d_{ff} = 2{,}048$, dropout rate $p_{drop} = 0.1$, and model dimension $d_{model} = 512$.

We used the default configuration alongside two configurations as shown in Table I. However, for all three configurations, we used sparse categorical crossentropy as a loss function and *adam* optimizer with the same custom learning rate scheduler in [13].

TABLE I. MODEL CONFIGURATIONS

| Parameter | Conf. 1 | Conf. 2 | Conf. 3 (base) |
|---|---|---|---|
| $N$ | 2 | 4 | 6 |
| $d_{model}$ | 64 | 128 | 512 |
| $d_{ff}$ | 128 | 512 | 2,048 |
| $h$ | 8 | 8 | 8 |
| $p_{drop}$ | 0.1 | 0.1 | 0.1 |
| Batch Size | 128 | 128 | 128 |

We trained the model on the Tashkeela and Wiki-40B sets (described below). In result, we produced two models, the Tashkeela model trained on Tashkeela set and Wiki model trained on Wiki-40B set. For training, we used TPU accelerator on Kaggle to reduce the training time; training with a GPU is slower (22 *vs.* 0.8 hrs. per epoch for the Wiki model).

### B. Training Approach

In this paper, we base the training approach of stochastic error injection [2]; an approach that is built entirely on the concept of artificial errors. The errors are injected onto the set randomly using a parameter called the *error injection rate p*. The authors in [2] reported best model results using an error injection rate of 40%. Thus, we used it as a baseline when evaluating the error injection rate. We also adopted the one-to-one mapping approach in [2], where certain two-letter combinations and the letters that appear at the end of the word are represented by one special token each. Additionally, our target characters are split into four sets based on which characters are often confused in place of others. The first set contains the various shapes of *al-hamzat* (ا, أ, إ, آ, ؤ, ئ, ء, اء), the second set contains the two shapes of *alef* at the end of the word (ا, ى), the third set contains the two shapes of *waw* at the end of the word (و, ؤ), and the fourth set is for the *teh*, *teh marbuta*, and *heh* at the end of the word (ت, ة, ه). Each character in these sets is randomly replaced with another character from its own set in the stochastic error injection approach.

### C. Datasets

We used two sets: the Wiki-40B and Tashkeela.Wiki-40B is a set that contains cleaned articles from Wikipedia for over 40 languages. The Arabic version contains 220,885 articles for training, 12,271 articles for testing, and 12,198 articles for validation written in modern standard Arabic (MSA). The set was released as part of multilingual language modeling research that introduced a language model for over 40 languages [15]. We used the training and validation subsets as are and for testing, we randomly selected 2000 sequences from the test subset. The reason for that is to reduce the testing time since it takes approximately 100 hours for the entire test subset and Kaggle only offers 30 hours per week to use its accelerators.

The second set that we used is a cleaned version of the Tashkeela set that contains text written in classical Arabic and text from the Holy Quran. The set is split into 50k lines for training, 2.5k lines for validation, and 2.5k lines for testing. This set was released as part of Arabic text diacritization research [16]. We used the test subset as is without changes. The training and validation subsets were merged then redistributed and split into 80% training and 20% validation.

We selected a maximum sequence length of 300 for training, and sequences longer than 300 were wrapped to get the maximum benefit from all the text. Table II shows the characteristics for both sets.

TABLE II. WIKI-40B AND TASHKEELA SETS CHARACTERISTICS

| Criterion | Wiki-40B Set | Tashkeela Set |
|---|---|---|
| Sequence count | 4,015k | 55k |
| Word count | 73,525k | 2,312k |
| Arabic letter count | 354,047k | 9,189k |
| Words per sequence | 18.31 | 42.05 |
| Letters per word | 4.82 | 3.97 |
| Sequences ≤ 300 chars | 95.78% | 78.06% |

Lastly, we used the Test200 set for the final test. This set contains 200 sequences with real soft spelling mistakes and an average of 6.5 mistakes per sequence [2].

### D. Evaluation Metrics

We evaluate our models with the following metrics: Accuracy, BLEU score, CER, WER, and FP/Changes. Accuracy is a general metric that measures the overall performance of a model. CER is the ratio of the incorrect characters in the text. WER is the ratio of the incorrect words in the text. BLEU is often used in machine translation tasks to measure how close the translated text generated by a model to the reference translation, we used SACREBLEU that was introduced in [17]. FP/Changes is a metric introduced in [2] to obtain the error rate in the predicted text. It is calculated by dividing the false positive cases in the predicted text over the number of changes (artificial errors) in the input text.

## IV. EXPERIMENTS AND RESULTS

In this section, we will discuss the experiments we conducted to achieve the optimal results. We used the accuracy metric and BLEU score to show the models' ability to produce a well-structured and understandable text. As for the ability to correct; we used the remaining metrics CER, WER, and FP/Changes. We compare our results with the ones reported in [2].

### A. Model Size Results

In Fig. 3, we observe that Tashkeela model performance using four layers configuration is nearly identical to six layers configuration with an accuracy of 99.65% for the four layers and 99.62% for the six layers. With similar behavior Wiki model performance using four layers configuration is nearly identical to six layers configuration with an accuracy of 99.75% for four layers and 99.77% for six layers.

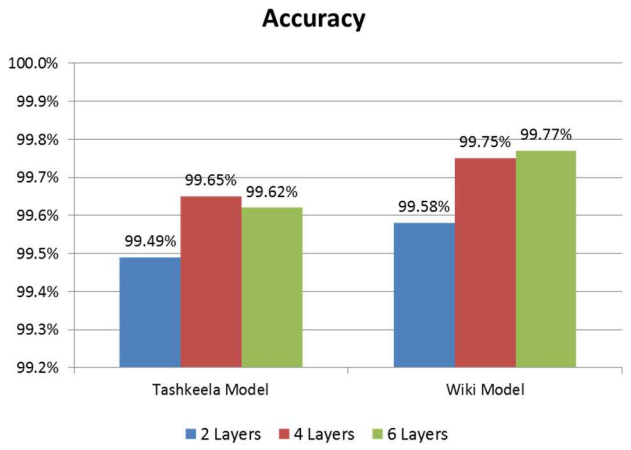


Fig. 3. Tashkeela and Wiki models accuracy results with various numbers of layers

In Fig. 4, the Tashkeela model with four layers performs better in term of BLEU with a score of 95.79% for four layers and 95.35% for six layers. Wiki model scores 96.58% for six layers and 96.33% for four layers. Both models achieved high results in accuracy and BLEU, which indicates that the model can produce a well-structured text like the one provided in the target sequences.

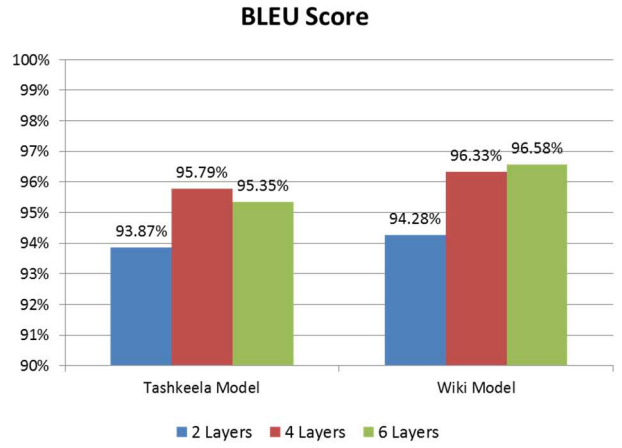


Fig. 4. Tashkeela and Wiki models BLEU results with various numbers of layers

The two models behave differently with the various numbers of layers due to the different size of each set; thus, we cannot decide which size is best based on these results only. Therefore, we used CER on Test200 to determine the best model size. In Fig. 5, we observe that the four-layer configuration achieves the lowest CER for both models with a score of 2.05%, and 3.11% for Tashkeela model and Wiki model, respectively. Therefore, we adopted four-layer model size.

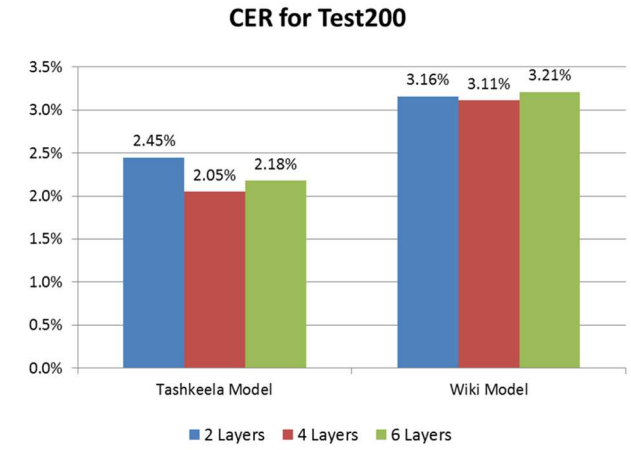


Fig. 5. Tashkeela and Wiki models CER results on Test200 set with various numbers of layers

## B. *Error Injection Rate Results*

In Fig. 6 and Fig. 7, we show the accuracy and BLEU score for both models with various error injection rates. The models at 30% error injection rate achieved best results. Both accuracy and BLEU score decrease with an increase in the error injection rate. This behavior extends to the CER and WER metrics as shown in Fig. 8 and Fig. 9. It also like the behavior of the BiLSTM models reported in [2]. The reason behind this behavior is the fact that these metrics are calculated for the entire set of characters. Therefore, we used FP/Changes metric to show that increasing the error injection rate helps the model to correct better. The CER results on Test200 also support this assumption since the models achieved the best results at 90% error injection rate.

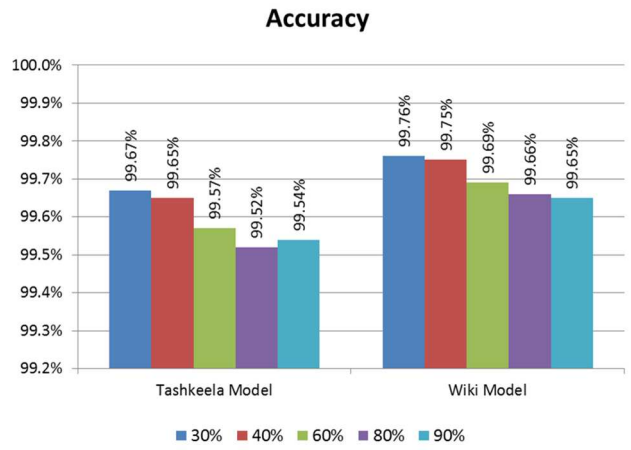


Fig. 6. Tashkeela and Wiki models accuracy results with five error rates

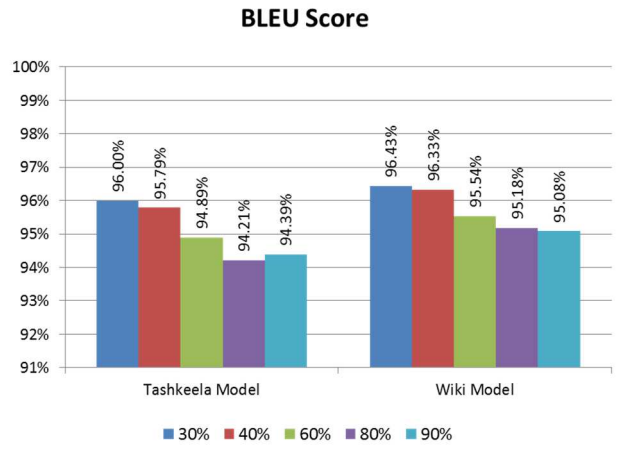


Fig. 7. Tashkeela and Wiki models BLEU results with five error rates

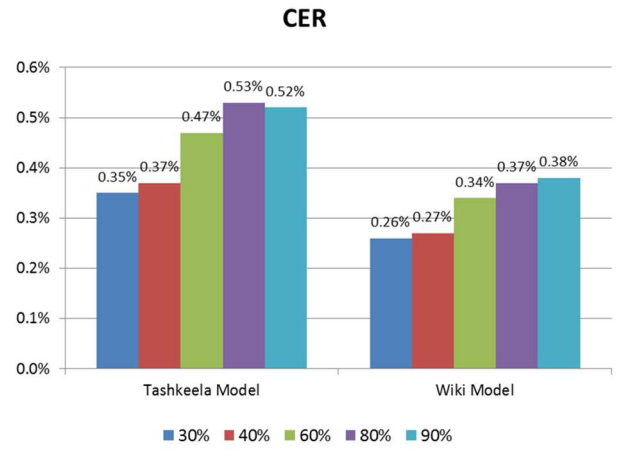


Fig. 8. Tashkeela and Wiki models CER results with five error rates

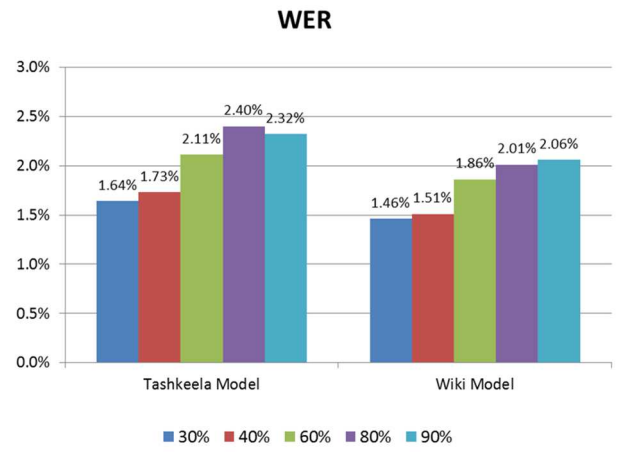


Fig. 9. Tashkeela and Wiki models WER results with five error rates

In Fig. 10, we show the results of FP/Changes metric to assert the observation that increasing the error injection rate helps the model to correct better. One can observe that both models trained with 90% error injection rate have the lowest values. While the ones trained with 30% error injection rate have the highest values even though they have fewer errors.

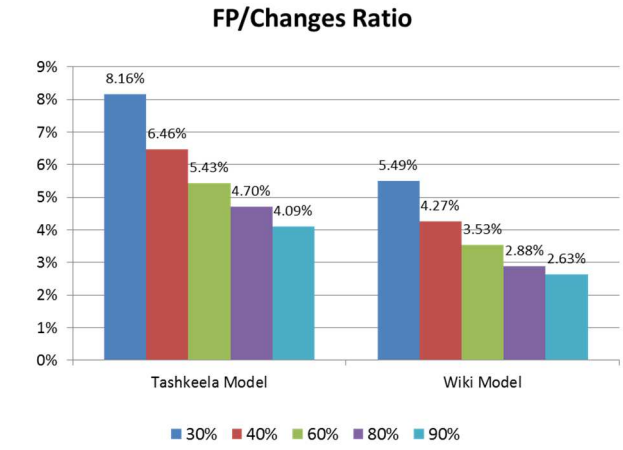


Fig. 10. Tashkeela and Wiki models FP/Changes results with five error rates

## C. *Test200 Results*

The results on Test200 in Fig. 11 show that the Wiki model shows lowest CER of 0.86% with 90% error injection rate. This is a reduction of 72.35% compared to the 3.11% that is obtained with 40% error injection rate. The Tashkeela model does not show this large improvement with the error injection rate increase; the score is 1.88% with 90% error injection rate, it is only 8.29% reduction compared to the 2.05% CER with 40% error injection rate. These results demonstrate the importance of the dataset size that is used in transformer model. Small sets like Tashkeela are likely to work better with LSTM models rather than transformer model. Lastly, we tested whether increasing the error injection rate to 100% will improve the results of the Wiki model; the CER was 0.89% which is not better than the one obtained using 90% error injection rate.

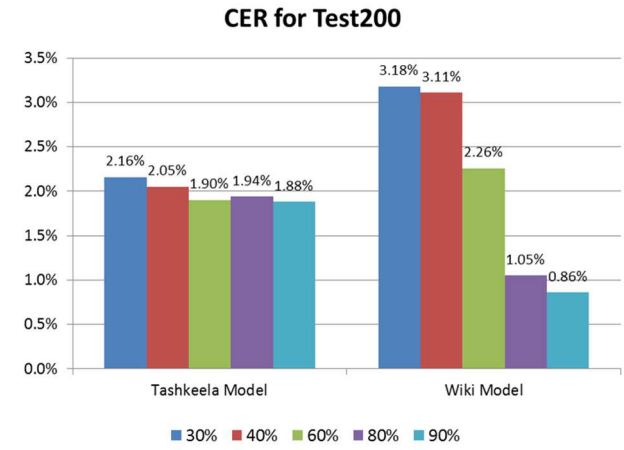


Fig. 11. Tashkeela and Wiki models CER results on Test200 set with five error rates

## D. *Comparison*

We compared our results on Test200 with the ones reported in [2] using the Tashkeela and ATB3 datasets and BiLSTM network, see Fig. 12. Our Wiki transformer model achieves a CER of 0.86%, which is lower than the 1.28% obtained by Tashkeela BiLSTM model. On the other hand, our Tashkeela transformer model does not outperform Tashkeela and ATB3 BiLSTM models.

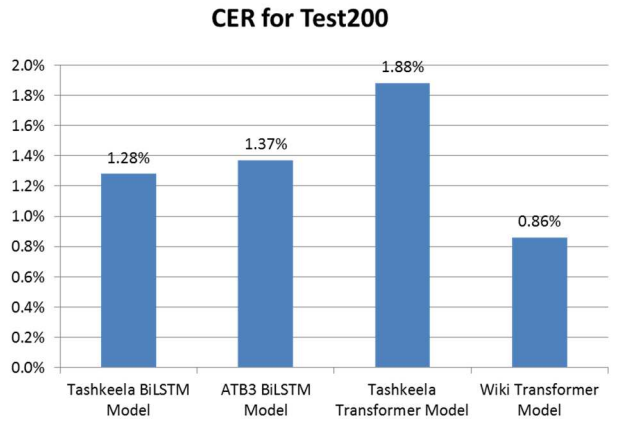


Fig. 12. Tashkeela and Wiki transformer models CER results on Test200 compared to ATB3 and Tashkeela BiLSTM models in [2]

## V. Conclusions and Future Work

In recent years, transformer models have proven to be powerful neural networks that can handle many NLP tasks quite well. Spelling correction is one of these tasks where transformers started to replace LSTM models and language models in the correction process. In this paper, we implemented a machine learning approach based on the original transformer to correct Arabic soft spelling mistakes in modern standard Arabic and classical Arabic. We handled four types of soft spelling mistakes, namely, confusion among *hamza* shapes, both shapes of *alef* at the end of the word, insertion and omission of *alef* after *waw aljamaea*, and confusion among *teh*, *teh marbuta*, and *heh* at the end of the word. The literature lacks a large, annotated dataset designed for Arabic spelling correction systems. To overcome this problem, we used artificial errors generated using a stochastic error injection approach that was proposed in [2]. We used Wiki-40b and Tashkeela sets for training and evaluation, and Test200 set to report our final results. We produced two transformer models, a Wiki model and a Tashkeela model. We compared our results on Test200 set with the ones that were obtained using BiLSTM models in [2]. Our Wiki model achieves a CER reduction of 32.81% over the best BiLSTM model that has been reported. Additionally, the Wiki model can correct 97.37% of the artificial errors that were injected into the test set.

For future work, we suggest looking at more advanced transformer models such as BART [18] and T5 [19]. These models contain encoder-decoder architecture and may be useful in spelling correction. We also suggest the proposed approach to correct other types of Arabic spelling mistakes such as semantic errors.

## References

[1] UNESCO, "World Arabic Language Day," 2019, https://en.unesco.org/commemorations/worldarabiclanguageday.

[2] G. Abandah, A. Suyyagh, and M. Khedher, "Correcting soft Arabic spelling mistakes using RNN-based machine learning," arXiv preprint arXiv:2108.01141, 2021.

[3] M. Alkhatib, A. A. Monem, and K. Shaalan, "Deep learning for Arabic error detection and correction," ACM Transactions on Asian and Low-Resource Language Information Processing, vol. 19, no. 5, pp. 1–13, 2020.

[4] A. M. Azmi, M. N. Almutery, and H. A. Aboalsamh, "Real-word errors in Arabic texts: A better algorithm for detection and correction," IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 27, no. 8, pp. 1308–1320, 2019.

[5] M. Attia, P. Pecina, Y. Samih, K. Shaalan, and J. Van Genabith, "Arabic spelling error detection and correction," Natural Language Engineering, vol. 22, no. 5, pp. 751–773, 2015.

[6] H. Noaman, S. Sarhan, and M. Rashwan, "Automatic Arabic spelling errors detection and correction based on confusion matrix-noisy channel hybrid system," Egypt Comput Sci J, vol. 40, no. 2, 2016.

[7] Y. Abdellah, A. S. Lhoussain, G. Hicham, and N. Mohamed, "Spelling correction for the Arabic language space deletion errors," Procedia Computer Science, vol. 177, pp. 568–574, 2020.

[8] N. AlShenaifi, R. AlNefie, M. Al-Yahya, and H. Al-Khalifa, "Arib$@$QALB-2015 shared task: A hybrid cascade model for Arabic spelling error detection and correction," Proceedings of the Second Workshop on Arabic Natural Language Processing, 2015.

[9] H. Tran, C. V. Dinh, L. Phan, and S. T. Nguyen, "Hierarchical Transformer Encoders for Vietnamese Spelling Correction," Advances and Trends in Artificial Intelligence, Artificial Intelligence Practices, pp. 547–556, 2021.

[10] A. Kuznetsov, and H. Urdiales, "Spelling correction with denoising transformer," arXiv preprint arXiv:2105.05977, 2021.

[11] Y. Zhao, X. Yang, J. Wang, Y. Gao, C. Yan, and Y. Zhou, "BART based semantic correction for Mandarin automatic speech recognition system," arXiv preprint arXiv:2104.05507, 2021.

[12] S. Zhang, H. Huang, J. Liu, and H. Li, "Spelling error correction with soft-masked BERT." Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, 2020.

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, AN. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," Advances in neural information processing systems, 2017.

[14] T. van den Berg, "Parametric Neural Networks," CQF Institute, https://www.cqfinstitute.org/sites/default/files/param-nn-1.3f.pdf.

[15] M. Guo, Z. Dai, D. Vrandečić, and R. Al-Rfou, "Wiki-40b: Multilingual language model dataset," In Proc. of the 12th Language Resources and Evaluation Conference, pp. 2440–2452, 2020.

[16] A. Fadel, I. Tuffaha, B. Al-Jawarneh, and M. Al-Ayyoub, "Arabic text diacritization using deep neural networks," 2019 2nd International Conference on Computer Applications & Information Security, 2019.

[17] M. Post, "A call for clarity in reporting BLEU scores," Proceedings of the Third Conference on Machine Translation: Research Papers, 2018.

[18] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," arXiv preprint arXiv:1910.13461, 2019.

[19] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and PJ. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," arXiv preprint arXiv:1910.10683, 2019.