Mansoura Journal for Computer and Information Sciences (2025), Vol.**20**, No.**2**
doi:10.21608/mjcis.2025.353920.1009

Mansoura Journal for Computer and Information Sciences

ARTICLE

Arabic Grammar Correction for Arabic Text Summaries

Nada Essa,[*,1] Mostafa El-Gayar,[1,2] and Eman El-Daydamony[1]

[1]Information Technology Department, Faculty of Computers and Information Sciences, Mansoura University, Mansoura, Egypt

[2]Department of Computer Science, Arab East Colleges, Riyadh 11583, Saudi Arabia

*Corresponding author: nadaessa2012@gmail.com

(Received: 18 January 2025; Accepted: 13 July 2025; Published: 19 July 2025)

Abstract

The Arabic grammar correction is important due to the complexity of Arabic grammar, domain change, lack of training data, lack of standard databases, and many vocabularies. There is still a lot to do to get a satisfactory result in correcting the grammar for Arabic. In this paper, a new open-domain technique is presented for Arabic grammar correction. It consists of three main stages: the creation of a database for Arabic grammar representations of correct and incorrect sentences, a sequence-to-sequence gated recurrent unit encoder-decoder architecture for training the database, and testing the encoder-decoder for Arabic grammar correction. It is based on a database of correct and incorrect Arabic sentence structure using part-of-speech tags, dependency relations between words, and the features of words. In addition, the system is designed to be implemented in any domain. The Qatar Arabic Language Bank 2014 and 2015 test sets are used to test the system. The results show that the system has achieved 96.9, 94.8, and 95.83 percent for precision, recall, and F-measure.

Keywords: Natural Language Generation; Arabic Text Summarization; Arabic Grammar Correction

1. Introduction

Arabic Grammar Correction (AGC) is a Natural Language Processing (NLP) activity that tries to design and improve automatic systems to repair text for a variety of problems such as spelling, grammar, and inappropriate word choice. In recent years, AGC has been classified as a machine translation task, which entails translating or converting the source or faulty input sentence into a corrected output sentence. The wrong words in the context will be automatically replaced with the proper and best alternatives without affecting the grammar. Due to the extensive vocabulary and complex set of grammatical rules in the Arabic language, an issue may appear simple, but it is challenging. AGC depends on rule-based techniques, statistical classification models, and deep

learning specifically. Despite being one of the most spoken languages in the world, it has received little attention in speech, writing programs, and data processing [1]. The limitations of current spell-checker systems for Arabic languages prompted the authors to study a simple and flexible method to develop an Arabic spell-checker system [2]. Recurrent Neural Networks (RNN) and Convolutional Neural Networks (CNN) are the two most used (Deep Neural Network (DNN) designs [3, 4]. RNN design is sequential, and it is widely utilized in NLP applications such as translation, named entity identification, and text creation. The RNN fundamental design has a problem processing and dealing with huge data sequences during backward propagation. This is known as the vanishing gradient issue, and it is often solved using Long Short-term Memory (LSTM) and Gated Recurrent Units (GRU) [5, 6]. CNN has a hierarchical design and can perform classification tasks such as sentiment analysis and picture categorization.

Arabic is a rich language that is regarded as one of the world's oldest. It is now rated fifth in terms of popularity [7]. For Arabic Natural Language Processing (ANLP) in general and the work of grammatical error correction and error type annotation in particular, Arabic presents several problems. Arabic has a diverse morphology [8]. The Arabic language has inflected words for person, gender, number, aspect, voice, mood, state, and case. Due to the use of optional diacritics, which are usually never present, Arabic has a very unclear orthography. The analysis can differ according to the part of speech (POS) [9]. Arabic is gradually being used in many information retrieval systems, including the Internet. The study of approaches for automatic spell-checker systems for Arabic text is critical to the expansion of Arabic texts on the Internet and other systems. The lack of standard open-domain datasets for AGC, and the most currently available databases are for correcting spelling errors in words. In addition to the complexity of the Arabic Language structure and the large number of grammatical rules, a new challenge is presented to generate an AGC system based on an open-domain database. All these reasons leads to consider solutions that create an open-domain database containing a representation of Arabic sentence structure based on the grammar rules and a method for reducing the number of grammatical rules used.

This paper presents a new solution for the lack of available datasets for AGC and its closed domain. It presents a new technique for AGC based on the summarization of Arabic complex texts to Arabic simple sentences. A new open-domain database of AGC is presented based on the description of simple Arabic sentence structures. Each sentence structure representation is made by the characteristics of its words and the dependency relationship between these words. The database contains two files: one for inputting incorrect sentences and the other for outputting correct sentences. This paper consists of three main stages: the creation of an open-domain database for AGC, a sequence-to-sequence encoder-decoder training using the created database, and testing the encoder-decoder for AGC.

1.1 Arabic grammar

One of the most crucial aspects of human language is grammar, which helps listeners or readers. There are two main types of sentences: Nominal and Verbal sentences. The subject of a sentence and the predicate structure make up a sentence. A predicate might be a verbal or nominal sentence. A nominal sentence may also begin with Inna/Kan and its sisters, which alters the ending of its case. The verb and subject are the two components that make up a verbal sentence. The verb must include one or more objects for a verb to be transitive. It comprises a verb and an agent in the passive voice. Arabic grammatical rules are extremely complicated and may even be confusing to native Arabic speakers. The challenge is due to a variety of factors, such as (1) longer sentences and complicated syntax; and (2) the unstructured word order in Arabic [10]. This is why examining Arabic grammar for inaccuracies could automatically help improve the cleanliness of the written Arabic content. The endings of Arabic words change according to their grammatical role in the Arabic sentence.

The main types of grammatical roles for Arabic are Nominative, Accusative, Jussive, and Genitive [11].

The main errors are classified into four classes. The first and second classes are the number disparity and the gender disparity. The third and fourth classes are missing basic parts of sentences and are inaccurate grammatical cases. Some basic components of Arabic sentences must have the same number and gender. Violation of these rules results in the first and second classes of errors. The third class misses the elemental parts of sentences, such as the subject. The fourth class of errors is the incorrect grammatical case. For example, the subject in a verbal sentence is in the nominative case and not in the genitive case [12].

Some examples of Arabic errors are shown in Figure 1. In (a), the error is the number dissimilarity. The subject is singular and the predicate is dual. The type of error in (b) is gender dissimilarity. The subject is feminine and the predicate is masculine. In (c) the verb at the beginning of a verbal sentence must be singular.

Figure 1. Examples of Arabic incorrect sentences

2. Related work

AGC is an essential task in natural language processing. AGC has two types of correction: spelling and grammar correction. In studies on AGC, specific domain and closed domain databases are used. There is a lack of databases for AGC for grammar. Previous research described numerous grammatical faults from various perspectives and used small, closed domains and different author databases. The strength in the proposed work is using an open domain database. Abdelaal et al. [13] examined the ability of Large Language Models (LLMs) to recognize and automatically correct syntax and grammatical mistakes in Arabic texts. They used Fine-tuning AraT5 and Prompting LLaMA-3 models. They obtained a precision score of 0.51, recall of 0.4, and F-score of 0.45 using Qatar Arabic Language Bank (QALB) for training and assessment. Mahmoud et al. [14] presented a cost-effective and efficient method for automated Arabic essay scoring on grammaticality assessment. The approach utilizes the pre-trained AraBART model while incorporating several parameter-efficient techniques. Initially, Parameter-Efficient Fine-Tuning (PEFT) is applied to optimize a minimal set of parameters tailored for each criterion. Additional strategies include Model Soup, Multi-Round Inference, and Edit Merging. Experiments were conducted using multiple datasets, including QALB-2014 and QALB-2015. They gained a precision of 84.9 85.1, a recall of 71.2, 75.4, and F-score of 74.7. AlOyaynaa et al. [15] presented a novel study for AGC using pre-trained language models based on transformers, which uses two methods—token level and sentence level. They suggested refined language models based on pre-trained language models named AraBERT and M-BERT. Several Arabic datasets that are publicly accessible were used for fine-tuning. With an F1-measure value of 0.87, recall of 0.90, and precision of 0.83 at the token level, and an F1-measure of 0.98, recall of 0.99, and

precision of 0.97 at the sentence level. Furthermore, this work demonstrates that the optimized language models based on the monolingual pre-trained models outperform the multilingual pre-trained models in Arabic. Solyman et al. [16] suggested an AGC model based on the sequence-to-sequence transformer. They initially suggested a noising technique for creating synthetic parallel data to get around the bottleneck caused by the dearth of corpora. In addition, they used the Expectation-Maximization routing algorithm to dynamically aggregate data across layers in the AGC, inspired by the success of capsule networks in computer vision. In addition, they added a bidirectional regularization term using Kullback-Leibler divergence to the training objective to enhance the agreement between Right-to-left and Left-to-right models to solve the exposure bias issue. Experiments on QALB-2014 and QALB-2015 databases with F1-measure of 74.18. Mahmoud et al. [17] improved the precision and comprehensibility of written language. But because there is a dearth of training data, creating an AGC framework for low-resource languages is extremely difficult. Using Arabic as a case study, they suggested a novel AGC framework for low-resource languages. They suggested the equal distribution of synthetic errors (EDSE), a semi-supervised confusion technique that produces a large amount of parallel training data, to produce additional training data. They also discussed two drawbacks of the traditional sequence-to-sequence AGC model: exposure bias during inference and imbalanced outputs from the unidirectional decoder. They used a knowledge distillation method from neural machine translation to get over these restrictions. Kullback-Leibler divergence is used as a regularization term in this method to quantify the agreement between two decoders: a forward decoder that runs from right to left and a backward decoder that runs from left to right. Their suggested framework performed better than the transformer baseline and two popular bidirectional decoding methods, namely synchronous and asynchronous bidirectional decoding, according to the testing results on two benchmarks. Additionally, their suggested framework reported the highest F1 score, and performance was significantly improved when synthetic data was generated for syntactic errors using the equal distribution technique. Their results showed how well the suggested framework works to enhance grammatical error correction for low-resource languages, especially. Solyman et el. [18] suggested novel aggressive transformation techniques to expand the spread of real data during training. Specifically, when the goal prefix is not useful for the following word prediction, it employs enriched data as auxiliary tasks to provide new contexts. Making the AGC model focus more on the encoder's text representations during decoding improved the encoder and raised its contribution. Arabic AGC served as a case study for the Transformer-based for low-resource AGC challenge, which was utilized to examine the effects of various strategies. With small training datasets and domain shift, AGC models trained with their data tend to source information more, have more domain shift robustness, and experience fewer hallucinations. According to experimental results, the suggested methods performed better than the baseline, the most popular data augmentation techniques, and traditional synthetic data techniques. Furthermore, a combination of the top three strategies—Misspelling, Swap, and Reverse—outperformed earlier Arabic AGC strategies and obtained the highest score in two benchmarks. Moukrim et al [19] used "Stanford Parser" with an ontology containing the Arabic language's rules to describe an automatic correction of these kinds of errors. They divided the text into sentences, extracted the word annotations using the syntactic relations from our parser, and then used the taxonomy to handle the relationships discovered. To find the mistake, they compared the original and corrected statements. The system that was put into place had an overall detection rate of about 94 percent. By comparing what comes out to the few Arabic grammar checkers that are currently available, it was concluded that the approach was promising. Madi et al. [20] showed the initial neural network model experiments for the job of detecting errors in texts written in Modern Standard Arabic (MSA). They examined various neural network designs and presented the evaluation findings obtained by using cross-validation on the data. Every experiment uses a dataset that they built and enhanced. 494 lines make up the corpus and 620 sentences after augmentation. Their models had a maximum precision of 78.09 percent, a recall of 83.95 percent, and F-0.5 score of 79.62 percent. They obtained a maximum accuracy of 79.21

percent, a recall of 93.8 percent, and an F-0.5 score of 79.16 percent using an LSTM. and a BiLSTM with a maximum precision of 80.74 percent, recall of 85.73 percent, and F-0.5 score of 81.55 percent produced the greatest outcomes. They compared the outcomes of the three models to a benchmark, which is an Arabic grammar checker that is accessible for purchase (Microsoft Word 2007). In terms of precision, F-0.5, LSTM, BiLSTM, and Simple RNN all beat the baseline. Preliminary findings indicated that neural network architectures for Arabic text error detection through sequence labeling can be effectively applied. The findings of their evaluation of the models in the dataset demonstrate that the Transformer-based automatic comparing the findings of the error correction model to those of earlier research models, significant and satisfactory results were obtained.

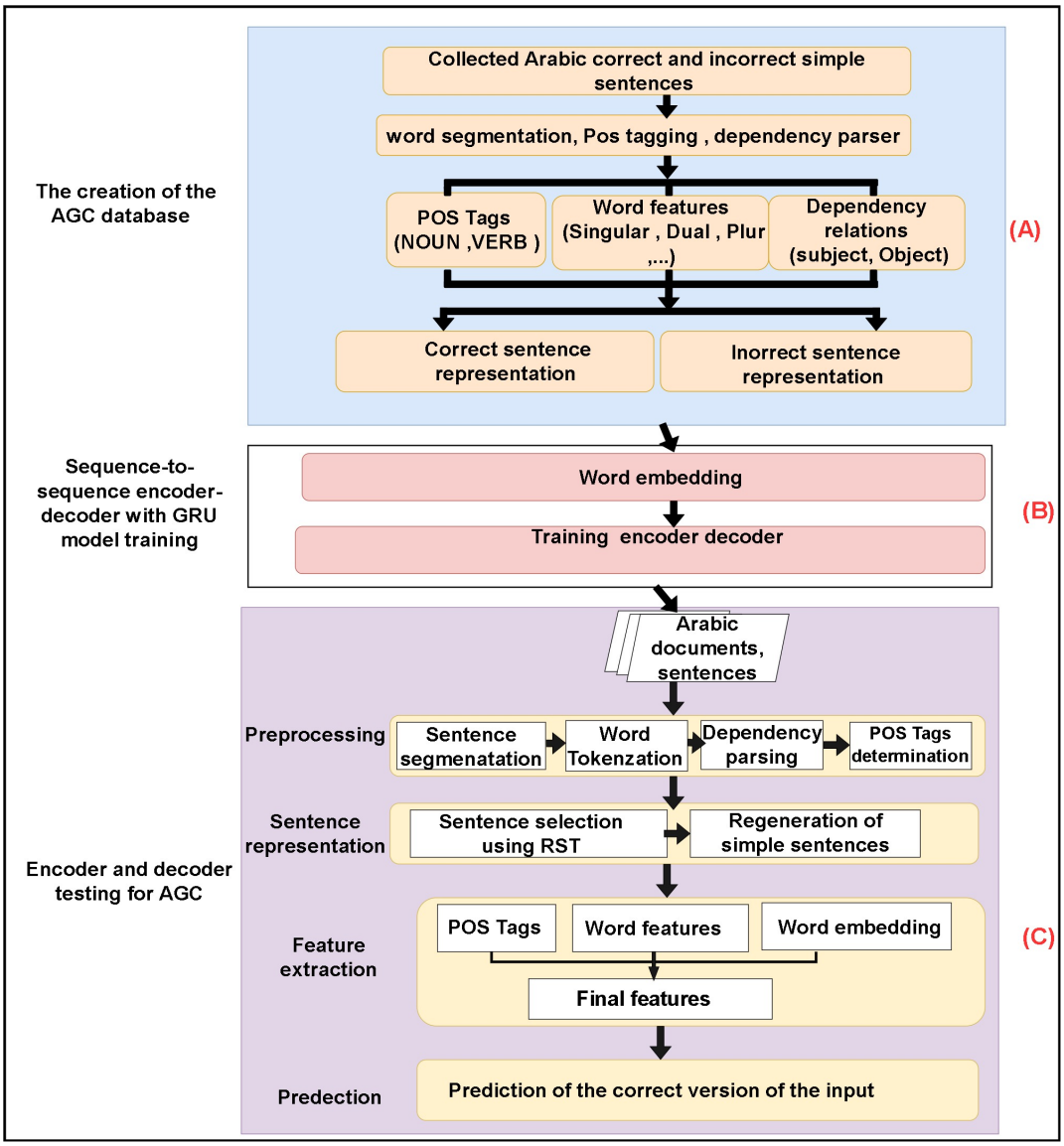

Figure 2. Illustration for all stages for the proposed system

3. The proposed technique

The AGC system is designed to convert incorrect sentences into grammatically and syntactically correct sentences. The basic idea of this technique is to convert complex Arabic sentences to simple sentences and correction of The AGC system consists of three main stages. First, an open-domain database for AGC is created. The second stage trains a sequence-to-sequence encoder-decoder after applying word embedding to the created database. The third stage includes testing the encoder and decoder for grammar error detection for generated summaries using the trained encoder-decoder. It consists of two main parts. An encoder that accepts variable-length sequences. A decoder that determines the wrong words in the target sequence using the encoded input. Figure 2 describes all stages of the proposed system for AGC.

3.1 Problem overview

This section briefly shows how all stages of the system work and the input and output of each stage. The proposed system contains three main stages. The first stage describes how the database is collected. The second stage is to train the sequence-to-sequence encoder-decoder using the database that was created. The third stage shows the AGC examples by testing the trained encoder-decoder. The database creation stage consists of three sub stages as shown in part (A) of Figure 2. First, the different Arabic grammar errors are collected. Second, this pair of correct and incorrect sentence structures is passed to the word segmentation, POS tagging, and dependency parser stage to extract the main features. The output of this stage is the representation of Arabic words as features in correct and incorrect sentences. Finally, incorrect and correct sentence representations are collected in two files that form the AGC database. The input file contains the representation of the structure for incorrect Arabic sentences. The output file contains the representation of the structure for the correct Arabic sentences. The output of the database creation stage is the open-domain AGC database. The second main stage of the system is training a sequence-to-sequence encoder-decoder, as shown in part (B) of Figure 2. It consists of two sub stages: word embedding and encoder-decoder training. The third main stage of the suggested system is the test of the encoder-decoder for AGC, as shown in part (c) of Figure 2. It consists of four sub-stages. Firstly, Arabic documents or sentences as input to the system and the trained encoder-decoder are passed to the preprocessing stage. The output of this sub stage is a list of sentences, words, POS tags, and dependency parser features. The second sub stage is the sentence representation stage. It consists of two steps. The first step is the removal of unwanted sentences using Rhetorical Structure Theory (RST) [21]. The second step is to generate new sentences by the conversion of Arabic sentences from complex to simple form. The output of this sub stage is the list of newly generated sentences in simple form. The third stage is the feature extraction and word embedding. The output of the third substage is the corrected Arabic sentence represented as vectors of learned word embeddings. The fourth sub-stage is the prediction of the correct Arabic sentence.

3.2 The creation of the AGC database

This stage involves the steps necessary to create the AGC database. The AGC database contains representations of the structure of incorrect Arabic simple sentences and their corrections. The main idea of this suggested database is to represent the structure of the Arabic sentence and not to depend on the spelling of the words that make up the sentences. First, most Arabic language errors were collected from more than one source. [22, 23, 24]. Second, 17724 examples of correct and incorrect simple Arabic sentences are collected using 684 representations of the Arabic sentence structure as shown in part (A) of Figure 3. Third, sentences are divided into words. The POS tag of each word, the features that represent each word, and the relationship between the words are extracted by dependency parsing as shown in part (c) of Figure 3 [25]. Part (c) shows that the Arabic

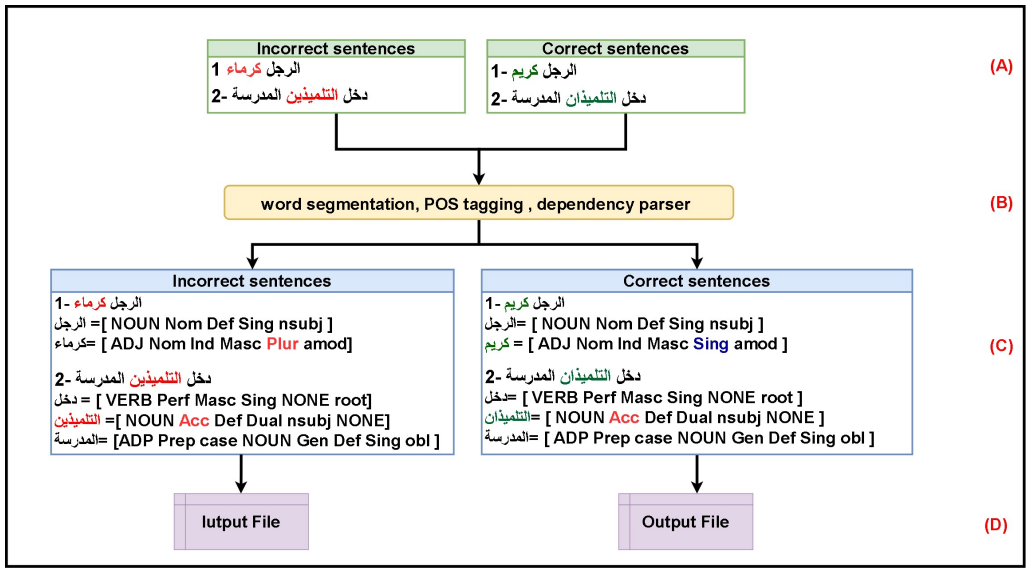

Figure 3. Steps of creation of the AGC database

sentence is separated into two words. Each word is represented by the POS tag, the grammatical case, the definiteness, the number, and the dependency relationship between the words [26]. A language can be represented as graphs using dependent grammar. Edges are dependencies, while nodes are words. The procedure assumed that each linguistic component of a sentence is linked to the others. Dependency structures, directed arcs that convey relationships between linguistic units or phrases in a sentence, are used to describe this. Directed graphs that adhere to the following restrictions are called dependency structures. They only have one designated root node and it does not receive any incoming arcs. In addition to the root node, each node has a single incoming edge. Each node can be reached via a different route from the root node. The dependency tag is a crucial component of dependency parsing. An indicator of the connection between two phrases is a dependency tag. An example of dependency parsing of an Arabic sentence is shown in Figure 4. In this stage, a Python NLP Toolkit for several human languages called Stanza is used [27]. Stanza supports many modules, such as tokenization, named entity recognition, lemmatization, dependency parsing, parsing, and POS tagging. The extracted words, their features, and dependency parsing relations of sentence number 1 in the table of correct sentences in Figure 3, using Stanza dependency parser, are shown in Figure 5. These features are combined to represent the structure of the correct and incorrect simple sentences. Finally, these representations are collected into two files, one for input and one for output. Every entry in the database represents an error type of the structure of the simple sentence in Arabic in the input file and its correction in the output file. The representation of a verb sentence structure in the AGC database is shown in Figure 6. Part (a) represents the Arabic sentence. Part (b) is the database representation of the features that describe the structure of the sentence. Part (b) represents the features of word number one and word number two that make up the sentence.

3.3 Sequence-to-sequence encoder-decoder with GRU model training

This stage explains the encoder-decoder training steps. First, the database with the two generated files, the representations of the structure of the incorrect input sentences and correct output sentences, is passed to the GRU-based encoder-decoder for training. Second, word embedding using the Continuous Bag-of-Words (CBOW) model is applied to the AGC database using three steps: vocabulary creation of the AGC database, vectorization of the sentence features, and vector representation

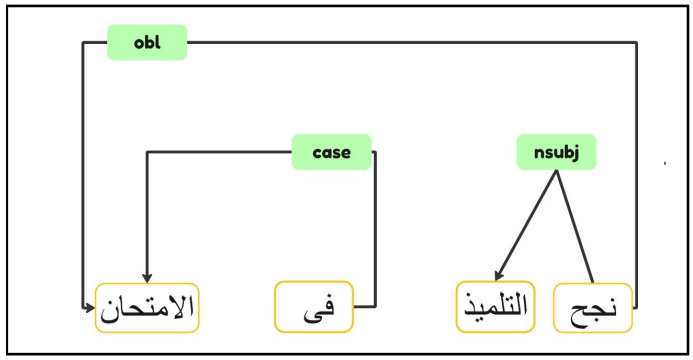

Figure 4. Dependency parser for Arabic sentence


```
{
      "id": 1,
      "text": "الرجل",
      "lemma": "رَجُل",
      "upos": "NOUN",
      "xpos": "N------S1D",
      "feats": "Case=Nom|Definite=Def|Number=Sing",
      "head": 0,
      "deprel": "root",
      "start_char": 0,
      "end_char": 5,
      "ner": "O",
      "multi_ner": [
        "O"
      ]
    },
    {
      "id": 2,
      "text": "كريم",
      "lemma": "كريم",
      "upos": "X",
      "xpos": "U---------",
      "head": 1,
      "deprel": "nmod",
      "start_char": 6,
      "end_char": 10,
      "ner": "O",
      "multi_ner": [
        "O"
      ],
      "misc": "SpaceAfter=No"
    }
```


Figure 5. Word features and dependency relation between words

of sentences. The CBOW word embedding model makes predictions about words using the words in their context. The word embedding is created by adding or averaging the word vectors of the context words. The vocabulary is created as a collection of unique words from the entire AGC database.

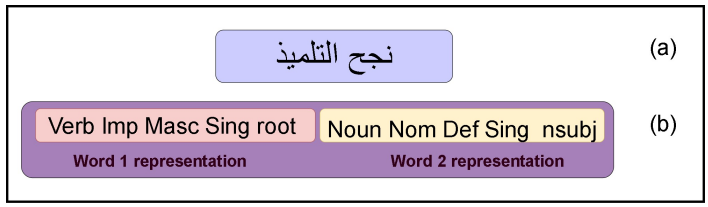

Figure 6. Example of correct sentence Representation in the database

Every word has its own index as shown in part(C) of Figure 7, such as a noun has index number one and a verb has index number seven. The output of this stage is a list of integers representing the words in the input sentence. The CBOW word embedding is a neural network consisting of three layers. The batch size used is 16, and the number of epochs is 100. The loss function used is a sparse categorical cross-entropy. The optimizer used is Adam. For each word, the context words surrounding it are determined. One-hot word vectors are created, and then the average of all the embeddings is calculated for dimensionality reduction of the data. The hidden layer converts the input to vectors. Multiplying a single hot vector by a weight matrix performs a lookup, obtaining the row in the matrix that represents the vector word. The output layer represents the resulting probabilities of each word in the vocabulary is the target word. The cosine similarity between the context vector and additional vocabulary word embeddings is calculated. The obtained cosine similarity is then subjected to the activation function "softmax." The possibility that each word is the target word is then displayed by softmax, which transforms cosine similarity into a probability distribution throughout the lexicon. The learned embedding vectors for each word are returned as shown in part (D) of Figure 7. Every entry in the database is replaced by its learned vector before passing to the encoder-decoder. Each encoder and decoder is a GRU network. The encoder network is used to obtain the vectors of the incorrect input sentence, which illustrates the input sequence that conveys its characteristics or context and produces a vector. The context vector is the name given to this vector. The decoder network obtains the context vector and learns to produce the corrected output sentence from it [28]. The model built with GRU layers seeks to resolve the problem of the vanishing gradient that arises with a typical recurrent neural network. Given their comparable designs and sometimes equally reliable results, GRU and LSTM can alternatively be thought of as variations of each other. The decoder is created using the attention layer [29]. For each step of the decoder's own output, attention enables the decoder network to focus on a different portion of the encoder's output. The model allows for a varied encoding of the source sentence at each decoding time step, and the attention mechanism selectively weights various sections of the source sentence during decoding. The learning rate used is 0.1. The loss function is the mean square error. The number of epochs is 500. The Adam optimizer is used. This model is implemented using PyTorch in Python [30]. The output of the decoder is as shown in part (E) of Figure 7.

3.4 Encoder and decoder testing for AGC

This stage is used to test the encoder-decoder trained for AGC. It includes an explanation of all the sub stages through which Arabic texts go from entering them into the system to obtaining correction of linguistic errors in them. The system is designed to detect errors in gender and number, in addition to grammatical errors. Repeated words and uncompleted sentences are also detected. This stage includes four sub-stages. The preprocessing stage includes sentence segmentation, word separation, dependency parsing, and determination of POS tags. The second stage is the representation of sentences. It combines RST to ignore unnecessary sentences and the formation of a simple Arabic sentence (SAS). Regeneration of SAS using only words that make up a simple sentence structure. Feature extraction is the third stage. It involves Pos tags, word feature detection, and the represen-

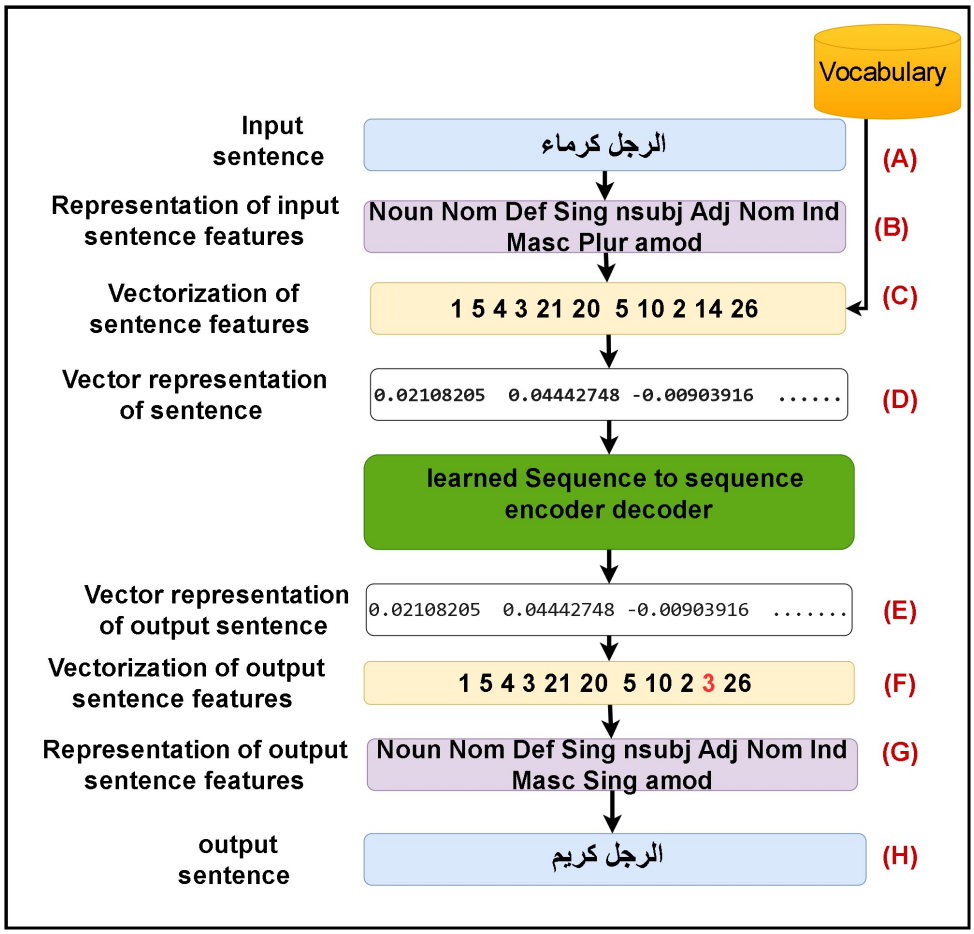

Figure 7. Input and output of encoder and decoder

tation of words as vectors using word embedding. The sub-stages from preprocessing to feature extraction of the AGC system are shown in Figure 8.

3.4.1 Preprocessing

The Arabic text input to the proposed technique is segmented into sentences, clauses, and words as shown in part (B) of Figure 8. Arabic text is passed to the Stanza parser to obtain the pos tags and features of words, in addition to the dependency relations between the words that make up the sentences.

3.4.2 Sentence representation

RST is used to laminate extraneous sentences and unnecessary and meaningless sentences from the Arabic text Input. Next, only basic words are selected to convert complex Arabic sentences to SAS as shown in part (c) of Figure 8.

3.4.3 Feature extraction

The newly generated Arabic simple sentences are passed to the Word Embedding step to obtain the representation of words as real-valued vectors in a lower-dimensional space and capture inter-word

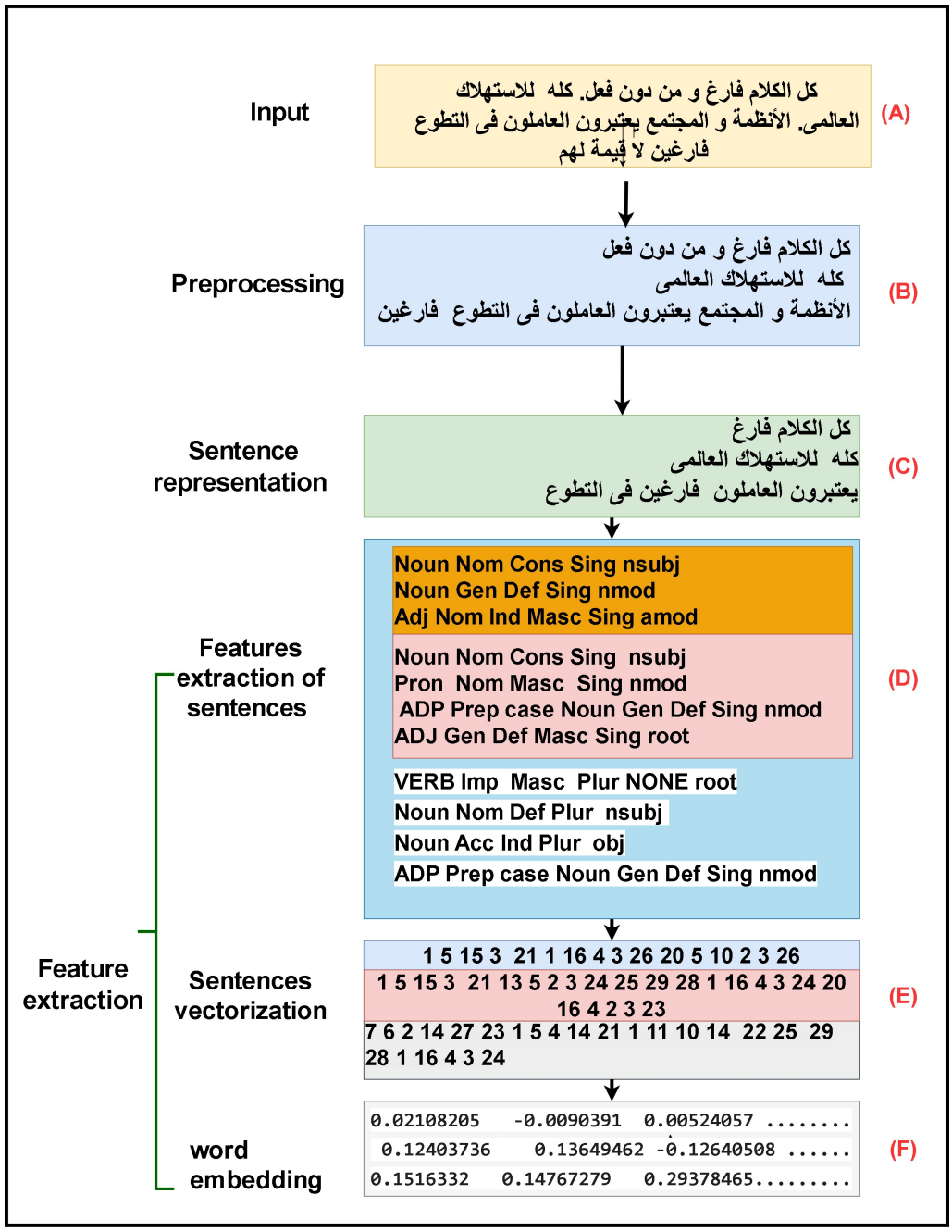

Figure 8. Preprocessing, sentence representation, and feature extraction sub-stages of encoder and decoder testing for AGC

semantics. Each word is represented by a vector of real values using the CBOW model as shown in parts (D, E, F) of Figure 8. The resulting vectors are passed to the trained encoder-decoder.

3.4.4 Prediction of the corrected sentences

This step shows how the correct sentence is predicted. It represents the conversion of the output vectors from the encoder and decoder trained previously into the words in the database used for training. The sentence vectors are returned as shown in part (A) of Figure 9. The cosine similarity of each word vector in the embedding matrix is computed by multiplying the target vector's dot product by the learned embedding matrix. Words that have the highest cosine similarity scores are given their indices as shown in part (B) of Figure 9. Every index is translated into its corresponding word as shown in part(C) of Figure 9. If there is an incorrect word, then it will be transformed to its lemma. After that, some characters will be added, removed, or changed according to a set of rules as shown in part (D) of Figure 9. The first verb in the third sentence in part (D) of Figure 9 is plural and is supposed to be singular; then the waw and noon characters will be removed from the end of the verb. And if the verb is supposed to be feminine, then the taa character will be added at the beginning of the verb.

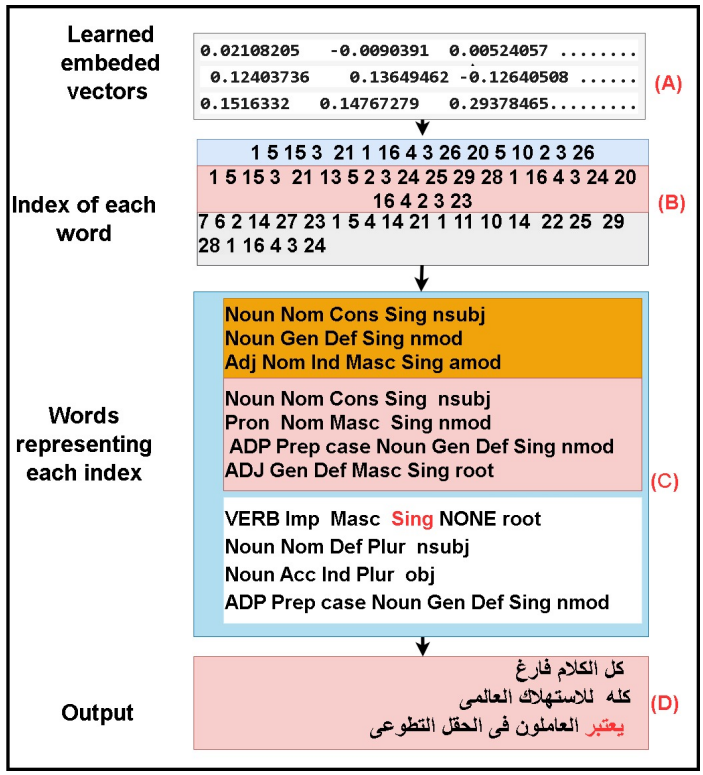

Figure 9. Prediction of the corrected sentences of encoder and decoder testing for AGC

4. Experimental Results

Due to the lack of standard databases for AGC and the existence of databases that are either small, for a specific field, or to correct spelling errors. To ensure an acceptable comparison with previous works in AGC, we adhere to the same test database used by the most recent research, specifically, the test sets in the QALB 2014 and 2015 databases are used to test the proposed AGR technique [31, 32, 33]. This is despite the difference in the database for the training stage. Shared tasks were organized as part of the QALB project, which aimed to develop a large manually corrected corpus of Arabic texts.

The first QALB shared task, QALB-2014, focused on correcting errors in online comments written by native Arabic speakers. The second shared task, QALB-2015, extended the task to include a track to correct errors in texts written by Arabic learners. The output is considered correct if the system gives a syntactically correct sentence. The precision, recall, and F-measure are calculated for the proposed system for AGC according to equations (1-3) as 96.9,94.8,93.86 percent. A straightforward table that compares the predictions of the classification model with the actual outcomes is called a confusion matrix [34]. Correct predictions for both classes (True Positives (TP) and True Negatives (TN)) and inaccurate predictions (False Positives (FP) and False Negatives (FN)) are the four categories into which that division is made. The number of occurrences the model generated on the test data is shown in the matrix. Precision is the ratio between the True Positives and all the Positives. Recall is the measure of our model that correctly identifies True Positives [35]. Not all grammatical errors are tested. Our database does not support all types of errors, such as complex sentence structure errors, hidden pronoun cases, and some idafa construction cases.

$$Precision(P) = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{1}$$

$$Recall(R) = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{2}$$

$$F1Score(F1) = 2\frac{\mathrm{PR}}{\mathrm{P} + \mathrm{R}} \tag{3}$$

The results of our experiments on the test set of QALB 2014 and QALB 2015 are presented in Table 1 compared to other recent work. Figure 10 represents a comparison between the other work and the presented work.

Table 1. The different results between other work and the presented work

Method	Database	Precision	Recall	F1	Year
AraT5 and LLaMA-3 models [13]	QALB	0.51	0.4	0.45	2024
AraBART mode [14]	QALB	85.1	75.4	74.7	2024
knowledge distillation technique with two decoders [17]	QALB	64.37	45.51	53.32	2023
Transformer-based for low-resource [18]	QALB	75.99	58.29	65.98	2023
seq2seq Transformer [16]	QALB	79.06-	70.43-	74.18	2022
The proposed system	QALB	96.9	94.80	95.83	2025

After examining the findings of earlier research, the findings of Fine-tuning using a trained language model, such as Arabert [14], are better than those of other work. All previous work listed in Table 1 mentioned that AGC has had a challenge until now due to the lack of a basic database and the limited number of databases that can be relied upon during the training phase. They also discussed the limited amount of data available in the available databases. The works in [17] [18] proposed solutions to this problem by increasing the amount of data used in the training stage. This paper presented another solution to the mentioned database problems: relying on a database that represents various incorrect and correct simple Arabic sentences based on Arabic word type and features, and the dependency relation between words. The suggested system performed better than other recent work with the difference in the database used in the training stage. The challenges facing the database are that it does not represent all the rules of the Arabic language. The possibility of dividing words that contain the letter waw. Stanza dependency parser cannot identify some types of words or the grammatical case correctly.

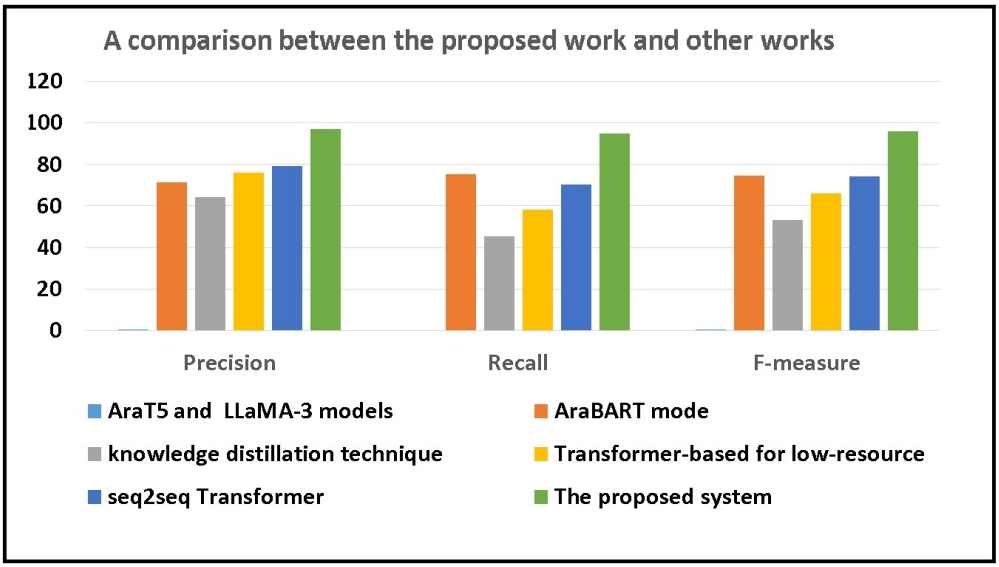

Figure 10. A comparison between the proposed work and other recent work

5. Conclusion and future work

The databases used in AGC are not standard and differ according to the domain. In this paper, we present a new technique for AGC based on an open-domain database that contains incorrect and correct representations of the SAS structures. These representations are based on the dependency parser relations and word features such as POS tags. The proposed technique contains the main stages: the creation of the database, training a sequence-to-sequence GRU encoder-decoder, and testing the encoder and decoder for AGC. The system is implemented using the Stanza and Pytorch libraries. The test sets in the QALB 2015 and 2014 databases are used for evaluation. The percentages for precision, recall, and the F score are 96.9, 94.80, and 95.83. However, there are still some challenges with unsupported Arabic errors. There are drawbacks to the inability of the stanza to recognize POS tags and grammatical case. Stanza does not recognize all the words for Inna or Kan and their sisters. The database has difficulties because it does not include all the grammatical rules of Arabic. The potential for word divisions including the letter "waw". We will expand the created database to cover all combinations of Arabic errors by adding more entries in the database, we will try to solve unsolved challenges by using different dependency parser tools. We will apply the system to other databases to test with different Arabic errors.

Open data statement

The training dataset used in this research is freely available and can be requested via the following link: Arabic Grammar Correction Datbase. Additionally, the QALB (Qatar Arabic Language Bank) dataset is also freely available upon request via: QALB Shared Task 2015

References

[1] Yu Wang, Yuelin Wang, Kai Dang, Jie Liu, and Zhuo Liu. “A comprehensive survey of grammatical error correction”. In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 12.5 (2021), pp. 1–51.

[2] Hasan Muaidi and Rasha Al-Tarawneh. "Towards arabic spell-checker based on N-grams scores". In: *International Journal of Computer Applications* 53.3 (2012), pp. 975–8887.

[3] Jianqiong Xiao and Zhiyong Zhou. "Research progress of RNN language model". In: *2020 ieee international conference on artificial intelligence and computer applications (icaica)*. IEEE. 2020, pp. 1285–1288.

[4] Jianxin Wu. "Introduction to convolutional neural networks". In: *National Key Lab for Novel Software Technology. Nanjing University. China* 5.23 (2017), p. 495.

[5] Alex Sherstinsky. "Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network". In: *Physica D: Nonlinear Phenomena* 404 (2020), p. 132306.

[6] Rahul Dey and Fathi M Salem. "Gate-variants of gated recurrent unit (GRU) neural networks". In: *2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS)*. IEEE. 2017, pp. 1597–1600.

[7] Imane Guellil, Houda Saâdane, Faical Azouaou, Billel Gueni, and Damien Nouvel. "Arabic natural language processing: An overview". In: *Journal of King Saud University-Computer and Information Sciences* 33.5 (2021), pp. 497–507.

[8] Yasser Al-Dossari, Abdullah Al-Yahy, Abdulaziz Al-Sumari, et al. "Preprocessing Arabic text on social media". In: *Heliyon* 7.2 (2021), pp. 2–15.

[9] Wasan AlKhwiter and Nora Al-Twairesh. "Part-of-speech tagging for Arabic tweets using CRF and Bi-LSTM". In: *Computer Speech & Language* 65 (2021), p. 101138.

[10] William Wright. *A Grammar of the Arabic Language: Vol. I*. BoD–Books on Demand, 2022.

[11] Kees Versteegh. "The Arabic Grammatical Tradition". In: *Inference: International Review of Science* 5.3 (2020), pp. 1–9.

[12] Wajdi Zaghouani, Behrang Mohit, Nizar Habash, Ossama Obeid, Nadi Tomeh, Alla Rozovskaya, Noura Farra, Sarah Alkuhlani, and Kemal Oflazer. "Large scale arabic error annotation: Guidelines and framework". In: *Proceedings of the 9th International Conference on Language Resources and Evaluation*. European Language Resources Association (ELRA). 2014, pp. 2362–2369.

[13] Ahmad Abdelaal, Abdelrahman A Medhat, Mohamed Elsayad, Muhammed Foad, Salma Khaled, Ahmed Tamer, and Walaa Medhat. "Text Correction for Modern Standard Arabic". In: *Procedia Computer Science* 244.C (2024), pp. 371–377.

[14] Somaia Mahmoud, Emad Nabil, and Marwan Torki. "Automatic Scoring of Arabic Essays: A Parameter-Efficient Approach for Grammatical Assessment". In: *IEEE Access* 12 (2024), pp. 142555–142568.

[15] Sarah AlOyaynaa and Yasser Kotb. "Arabic Grammatical Error Detection Using Transformers-based Pretrained Language Models". In: vol. 56. 04009. EDP Sciences. 2023, pp. 1–14.

[16] Aiman Solyman, Zhenyu Wang, Qian Tao, Arafat Abdulgader Mohammed Elhag, Rui Zhang, and Zeinab Mahmoud. "Automatic Arabic Grammatical Error Correction based on Expectation-Maximization routing and target-bidirectional agreement". In: *Knowledge-Based Systems* 241.108180 (2022), pp. 1–13.

[17] Zeinab Mahmoud, Chunlin Li, Marco Zappatore, Aiman Solyman, Ali Alfatemi, Ashraf Osman Ibrahim, and Abdelzahir Abdelmaboud. "Semi-supervised learning and bidirectional decoding for effective grammar correction in low-resource scenarios". In: *PeerJ Computer Science* 9.e1639 (2023), pp. 1–25.

[18] Aiman Solyman, Marco Zappatore, Wang Zhenyu, Zeinab Mahmoud, Ali Alfatemi, Ashraf Osman Ibrahim, and Lubna Abdelkareim Gabralla. "Optimizing the impact of data augmentation for low-resource grammatical error correction". In: *Journal of King Saud University-Computer and Information Sciences* 35.6 (2023), pp. 1–15.

[19] Chouaib Moukrim, Tragha Abderrahim, Almalki Tarik, et al. "An innovative approach to autocorrecting grammatical errors in Arabic texts". In: *Journal of King Saud University-Computer and Information Sciences* 33.4 (2021), pp. 476–488.

[20] Nora Madi and Hend Al-Khalifa. "Error detection for Arabic text using neural sequence labeling". In: *Applied Sciences* 10.15 (2020), p. 5279.
[21] Ahmed Ibrahim and Tarek Elghazaly. "Arabic text summarization using rhetorical structure theory". In: *2012 8th International Conference on Informatics and Systems (INFOS)*. IEEE. 2012, pp. 34–38.
[22] Mohammed Sawaie. *Fundamentals of Arabic grammar*. 1st ed. Routledge, 2015.
[23] Mohammad Alhawary. *Arabic grammar in context*. 1st ed. Routledge, 2016.
[24] Maha S Al-Rabiah and AbdulMalik Al-Salman. "An XML-based semantic parser for traditional Arabic". In: *2010 4th International Universal Communication Symposium*. IEEE. 2010, pp. 312–319.
[25] Yuval Marton, Nizar Habash, and Owen Rambow. "Dependency parsing of Modern Standard Arabic with lexical and inflectional features". In: *Computational Linguistics* 39.1 (2013), pp. 161–194.
[26] Sandra Kübler and Emad Mohamed. "Part of speech tagging for Arabic". In: *Natural Language Engineering* 18.4 (2012), pp. 521–548.
[27] Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. "Stanza: A Python Natural Language Processing Toolkit for Many Human Languages". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Association for Computational Linguistics. 2020, pp. 101–108.
[28] Rajvardhan Patil, Sorio Boit, Venkat Gudivada, and Jagadeesh Nandigam. "A survey of text representation and embedding techniques in nlp". In: *IEEE Access* 11 (2023), pp. 36120–36146.
[29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is All you Need". In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc. 2017.
[30] Sagar Imambi, Kolla Bhanu Prakash, and G. R. Kanagachidambaresan. "PyTorch". In: *Programming with TensorFlow: Solution for Edge Computing Applications*. Springer International Publishing, 2021, pp. 87–104.
[31] Nizar Habash, Behrang Mohit, Ossama Obeid, Kemal Oflazer, Nadi Tomeh, and Wajdi Zaghouani. "QALB: Qatar Arabic Language Bank". In: *Proceedings of Qatar Annual Research Conference (ARC-2013)*. Hamad bin Khalifa University Press (HBKU Press). 2013.
[32] Behrang Mohit, Alla Rozovskaya, Nizar Habash, Wajdi Zaghouani, and Ossama Obeid. "The first QALB shared task on automatic text correction for Arabic". In: *Proceedings of the EMNLP 2014 workshop on arabic natural language processing (ANLP)*. Association for Computational Linguistics (ACL). 2014, pp. 39–47.
[33] Alla Rozovskaya, Houda Bouamor, Nizar Habash, Wajdi Zaghouani, Ossama Obeid, and Behrang Mohit. "The second QALB shared task on automatic text correction for Arabic". In: *Proceedings of the Second workshop on Arabic natural language processing*. Association for Computational Linguistics (ACL). 2015, pp. 26–35.
[34] S Sathyanarayanan and B Roopashri Tantri. "Confusion matrix-based performance evaluation metrics". In: *Afr. J. Biomed. Res* 27.4 (2024), pp. 4023–4031.
[35] Cyril Goutte and Eric Gaussier. "A probabilistic interpretation of precision, recall and F-score, with implication for evaluation". In: *European conference on information retrieval*. Springer. 2005, pp. 345–359.