Correcting Wide-range of Arabic Spelling Mistakes Using Machine Learning and Transformers

By

Raghda Diab Hasan

Supervisor

Dr. Gheith Ali Abandah, Prof.

This Thesis was Submitted in Partial Fulfillment of the Requirements for the Master's Degree in Computer and Networks Engineering

Faculty of Graduate Studies

The University of Jordan

April, 2023

COMMITTEE DECISION

This Thesis (Correcting Wide-range of Arabic Spelling Mistakes Using Machine Learning and Transformers) was Successfully Defended and Approved on

Examination Committee	**Signature**
Dr. Gheith Abandah (Supervisor) Prof. of Computer Engineering	----------
Dr. Iyad Jafar (Internal Member) Prof. of Computer Engineering	----------
Dr. Mohammad Abdel-Majeed (Internal Member) Assoc. Prof. of Computer Engineering	----------
Dr. Ahmad Al-Jaafreh (External Member) Prof. of Computer Engineering (Tafila Technical University)	----------

ACKNOWLEDGEMENT

My thanks are due to Prof. Gheith Abandah for his guidance, patience, and encouragement. I am grateful for giving me the chance to learn machine learning with him.

I would be remiss in not mentioning my family, especially my parents, husband, and children. Their belief in me has kept my spirits and motivation high.

Table of Contents

Subject **page**

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ASR	Automatic Speech Recognition
BART	Bidirectional and Auto-Regressive Transformer
BERT	Bidirectional Encoder Representations from Transformers
BiGRU	Bidirectional Gated Recurrent Unit
BiLSTM	Bidirectional LSTM
BLEU	Bilingual Evaluation Understudy
CA	Classical Arabic
CER	Character Error Rate
EGCM	Error-Guided Correction Model
FN	False Negative
FP	False Positive
GUI	Graphical User Interface
LSTM	Long Short Term Memory
MSA	Modern Standard Arabic
NLP	Natural Language Processing
NN	Neural Network
OCR	Optical Character Recognition
QALB	Qatar Arabic Language Bank
RNNs	Recurrent Neural Networks
SEC	Spell Error Correction
SVM	Support Vector Machine
SPIRAL	Spelling Error Corpora for Arabic Language
TN	True Negative
TP	True Positive
UTC	Universal Time Coordinated

Correcting Wide-range of Arabic Spelling Mistakes Using Machine Learning and Transformers

By

Raghda Diab Hasan

Supervisor

Dr. Gheith Ali Abandah, Prof.

ABSTRACT

Different languages are subject to several types of spelling mistakes. The mistakes are different from language to another, also the methods of solving those mistakes are dependent on the language. Natural language processing (NLP) techniques and machine learning algorithms became the most common procedure to handle these spelling mistakes efficiently. The transformer is considered one of the state-of-the-art methods among them. In this thesis, we use the transformer to handle a comprehensive set of Arabic language mistakes.

In this thesis, we use the transformer to correct many types of Arabic soft spelling mistakes, namely, lexical and semantic errors (due to the sound of the letter), keyboard errors (due to the character position on the keyboard), common typing errors (like deleting space, preposition errors, third pronoun errors), and some random typing errors when the user replaces a letter with any other letter.

The methodology of our work is divided into two stages: in the first stage we work on a huge dataset of Arabic texts which is error free called Wiki-40B dataset. We injected this dataset with the synthetic targeted errors with specific probabilities, then performed training and testing on this dataset. In the second stage, we took the training weights resulted from training on the Wiki-40B dataset and performed training and testing on a dataset that we collected for this purpose and contains real Arabic spelling mistakes, we call this dataset REALMS, which stands for REal Arabic Language Mistakes in Spelling. Collecting, preparing, and processing REALMS dataset gave us a good insight about the classes of errors in Arabic language, their frequencies, and the best types of synthetic errors to be injected in the Wiki-40B dataset to help in the training stage.

During the stage of error injection in Wiki-40B dataset, we used several error injection rates. The best result is for the model that is trained on Wiki-40B dataset at 20% error injection rate using 4-layer transformer configuration. The model can

correct 99.12% of the artificial errors that are injected into the test set and achieves a character error rate of 1.14%, and accuracy of 98.85% on REALMS dataset which contains many different error types.

Chapter 1

Introduction

All creatures have a way of communication with their kind. Different animals communicate using different types of signals, including visual; auditory or sound-based; chemical, involving pheromones; or tactile, touch-based, and cues. However, the mankind uses in communication mainly natural languages like English, Arabic, French, *etc*.

Communication languages can be either spoken or written. The spoken language is easier to use, easier to understand, less subject to different kinds of errors, and can be easily understood even with errors. However, written language is more prone to different types of errors.

People use written language for many reasons: to communicate over the Internet, to send emails, to chat with others, to write: letters, books, research papers, and articles, to communicate with people, and for many other reasons. Some written communication may tolerate having errors like chat with a friend. But the errors in other types of communication may be problematic like writing a research paper, thesis, email to your manager, writing a news article, and many others. Writing with errors may result in decreasing the writer's reputation value, losing job, facing a rejection of research papers, not being understood by others, generate translation errors, and many other kinds of effects.

Based on that, different people need to write error free text. But this requires that those people be expert in the language. Unfortunately, languages writing rules have a lot of details that are very hard for regular people from different backgrounds

to stay aware of these rules. For this reason, a lot of applications started to have automatic error detection and correction feature like Microsoft Word, browsers, and others.

Error correction is dependent on the language itself. For example, the types of errors in the English language are different from the errors in the Arabic language. Therefore, there should exist research and solutions for each language independently. Unfortunately, the research on the Arabic language is much less than the research done on other languages like English language.

On the other hand, there is a big need for doing experiments and research on the area of Arabic spelling mistakes correction for the following reasons: 1) 290 million people have Arabic mother language and 422 million people speak Arabic around the world (UNESCO, 2019). Their skills in writing is not the same. 2) The Arabic language is considered the fifth most spoken global language (Ethnologue, 2020). 3) Arabic language is considered the fourth most used language on the Internet (Johnson, 2020). 4) According to (Tinsley and Board, 2017), the demand for Arabic is increasing in many fields like politics, tourism, mobility, and others. Our focus in this thesis is on fixing Arabic typing errors.

1.1. Spelling Mistake Types

The errors in the Arabic language can be classified into:

a) **Discourse Structure and Pragmatic-level Errors**: these types of errors happen when not choosing the best suitable word in the sentence. Indeed, this is not a spelling mistake, instead the writer writes a correct word spelling-wise, but it is not suitable in that place. Such things happen a lot in automatic translation engines

like (Google translate). One example is translating the sentence “solving square equation is a piece of cake” which has the Arabic translation

"حل المعادلة التربيعية هو قطعة من الكعك"

b) Morpho-syntactic Errors: These errors are not spelling mistakes. However, Arabic language has specialty that verbs and numbers used with feminine are different from those used with masculine. This type of errors happens when mixing these forms. For example, “خمس كتب”, the word “خمس” should be “خمسة”.

c) Lexical and Semantic Errors: these types of errors are considered spelling mistakes, which are very popular in Arabic. These errors occurs when the writer changes a letter based on its phonetics. The resulted word could be a word that is lexically incorrect (lexical errors), or word that is not suitable for the sentence (semantic errors). For example, “يعتبر الظفدع” contains lexical error in “ظفدع” which is not an Arabic word. This lexical error happened due to the confusion between the letters *dad* (ض) and *tha’a (ظ). The correct word* should be“ضفدع”. Another example is the confusion between the letter *seen (س)* and the letter *sad* (ص) in a word. For example, the word ”صخرة” which means “rock” it may written mistakenly as “سخرة” and this has different meaning. Many such errors in different Arabic letter are existing in Arabic language.

d) Soft Errors: These spelling mistakes happen frequently due to the confusion that result from mixing the letters in very similar letters. These errors happen a lot in Arabic. Examples of these errors are: a) mixing various forms of Hamza. Many people face difficulty in writing the proper form of hamza. b) *teh* (ت), and *heh* (ه) at the end of the word. c) confusion between the letters which are adjacent in the keyboard and have small differences in the shape between them, like the letters

jeem (ج), *ha'a* (ح), and *kha'a* (خ), these letters are adjacent to each other in the keyboard and the only difference between them is the dot.

Our focus in this thesis is on types c and d. Specifically, we investigate fixing a comprehensive set of typing errors. To identify the set of errors exactly, we collected a dataset from social media like Facebook and Twitter. Then we analyzed this dataset to figure out the most common mistakes. Accordingly, we built an efficient machine learning based model that fixes the most common typing. The proposed model was able to fix more than 99% of the errors.

1.2. Thesis Contribution

In this thesis, our main contribution is summarized by the following:

1) Collect a real dataset of Arabic texts from the social media (like Facebook and Twitter). The dataset requires a lot of effort to be filtered, cleaned, classified, and analyzed.

2) Analyze the collected dataset to figure out the types of errors in the data. Then, we classified the typing errors and calculated the frequency of each type of errors.

3) Redesign the dataset to include column for the original text and column for the text after correcting the typing errors. This new design is used to train the machine learning model on errors.

4) Prepare a huge dataset called Wiki-40B which contains Arabic texts that are error free and inject it with statistical errors to train the machine learning model on a huge dataset. The error injection step is done based on our analysis of the collected dataset.

5) Extend the work done by (Al-Qaraghuli *et al.*, 2021) to allow it to correct many extra types of errors. More specifically, we will handle all errors solved by their model, as well as lexical and Semantic errors (due to the sound of the letter), keyboard errors

(due to the character position on the keyboard), common typing errors (like deleting space, adding space, preposition errors, third pronoun errors), and some random typing errors when the user replaces any letter with any other letter.

More details about the errors to be solved and the justification of handling these types of errors can be found later in Chapter 4.

1.3. Research Importance

The importance of this research can be summarized by the following:

- Correcting more spelling mistakes than the previous research which leads to go forward to a more comprehensive model that can correct all types of Arabic spelling mistakes.
- Indeed many people including native Arabic people may be subject to spelling mistakes when write in Arabic. Such a system could help both native Arabic speakers and nonnative.
- Such a system can be integrated with automatic translation engines (e.g. google translate) and optical character recognition (OCR) systems to generate error free texts.

1.4. Research Objectives

In this study, I have the following objectives:

- Survey the set of spelling mistakes that needs processing. To achieve that, I will build a sample dataset of Arabic texts which has some spelling mistakes. This dataset will be collected from different sources. Based on the collected dataset, I will choose the most common mistakes to work on them.

- Study the Arabic spelling mistakes and try to correct a larger set of mistakes that were not addressed in the previous works.
- Prepare the public datasets that were used in the previous works to be ready for the errors we will study using stochastic error injection techniques.
- Use and train transformers which is the state-of-the-art technique in correcting the Arabic spelling mistakes using the public dataset and the dataset we built.
- Perform tuning of the adopted model to improve the correction accuracy.

1.5. Questions

This study answers the following questions:

- Is it possible to correct more exhaustive Arabic spelling mistakes?
- Is it possible that the proposed model to help Arabic learners and native Arabic speakers to produce error-free Arabic text?
- What are the most common Arabic written language errors?
- What is the best model to correct these errors?

1.6. Methodology

We started our research by surveying the research that used the state-of-the-art technique in natural language processing (NLP) which is the transformers. Another survey that was done is checking which errors must be corrected and build a dataset from different sources like Facebook and others. The objective of the collected dataset is to check what are the most common errors. After that, we injected the Wiki-40B datasets with the target spelling mistakes. The training scheme is based on predicting the sequences of these datasets given inputs of these sequences with errors using the stochastic error injection scheme (Abandah *et al.*, 2022). Then, prepare these datasets to be usable for the machine learning model. Then, prepare the transformers model

and apply it on the two datasets. After that, a long tuning, testing, and evaluation process was followed to get the best results.

1.7. Thesis Outline

The rest of this thesis is structured as follows: in Chapter 2, we introduce a brief background about the transformer model and review the previous works that deal with spelling correction for both Arabic and other languages. In Chapter 3, we talk in more details about the collected dataset and the steps followed during dataset collection. After that, in Chapter 4, we explain how we fine-tuned our model and show our results in Chapter 5. Finally, we provide the conclusions and ideas for future work.

Chapter 2

Literature Review

Some Arabic spelling correction works make use of machine learning, notably BiLSTM, while the majority do not. Transformers for spelling correction have been used in most recent works for different languages. We present various methods for correcting spelling errors in various languages.

In this chapter, we introduce a brief background about the transformer model and discuss previous works that deal with spelling correction for both Arabic and other languages, as well as the metrics we used for evaluation the transformer model.

2.1. Transformer

The transformer was introduced in 2017 by (Vaswani *et al.,* 2017), they created the transformer that depends on the attention mechanism which already used in the RNNs. The main advantage of the transformer that its ability to process the input through several paths which allows for the simultaneous calculation of all words.

The self-attention or scaled dot-product attention is a novel attentional notion used in the construction of the Transformer model. Self-attention enables the model to look up a word's representation in the input sentence or to establish relationships between the word that is currently being processed and the other words in the input. Figure 1 provides an illustration of the idea of self-attention. 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”.

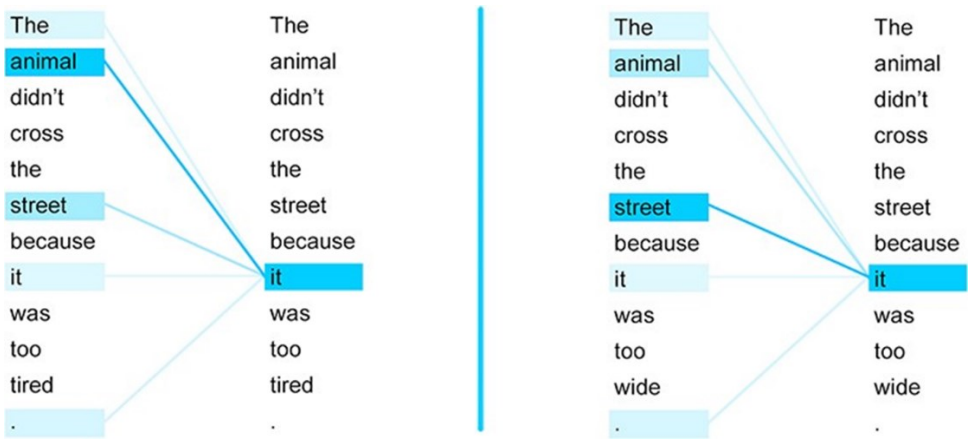

Figure 1. Example of Self-attention (van den Berg, 2020)

As an example, the three words in Figure 2. There are three candidate paths which may make the three words to go in the self-attention layer. Then and in the self-attention layer, these paths are interdependent, but not in the feed forward layer. Consequently, these paths can be processed in parallel by the feed forward layer, not in series like in RNNs. This is supposed to make the training to finish faster (Alammar, 2018).

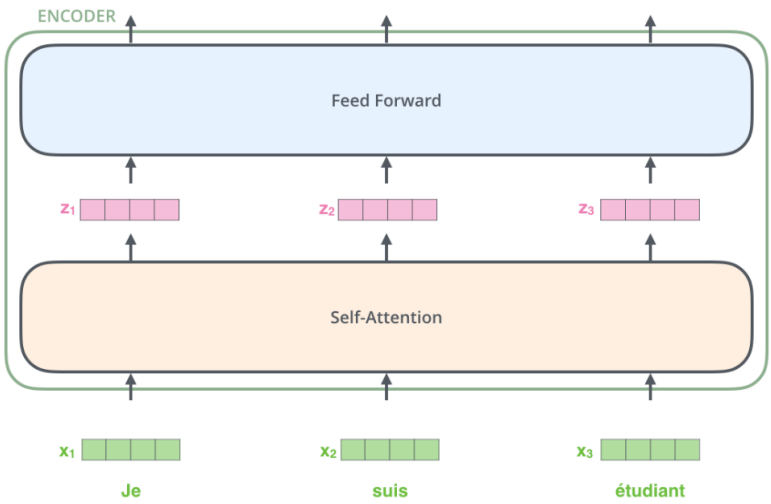

Figure 2. Input Paths in the Encoder (Alammar, 2018)

The transformer model is composed of eight heads (h), also called self-attention layers, which are combined to form multi-head attention. The use of multiple heads is useful to represent different points in the input sentence.

Transformer uses linear layer with softmax for the output and an encoder-decoder architecture with embedding and positional encoding for the input. Figure 3 displays an illustration of a two-layer transformer architecture made up of two encoder-decoder stacks.

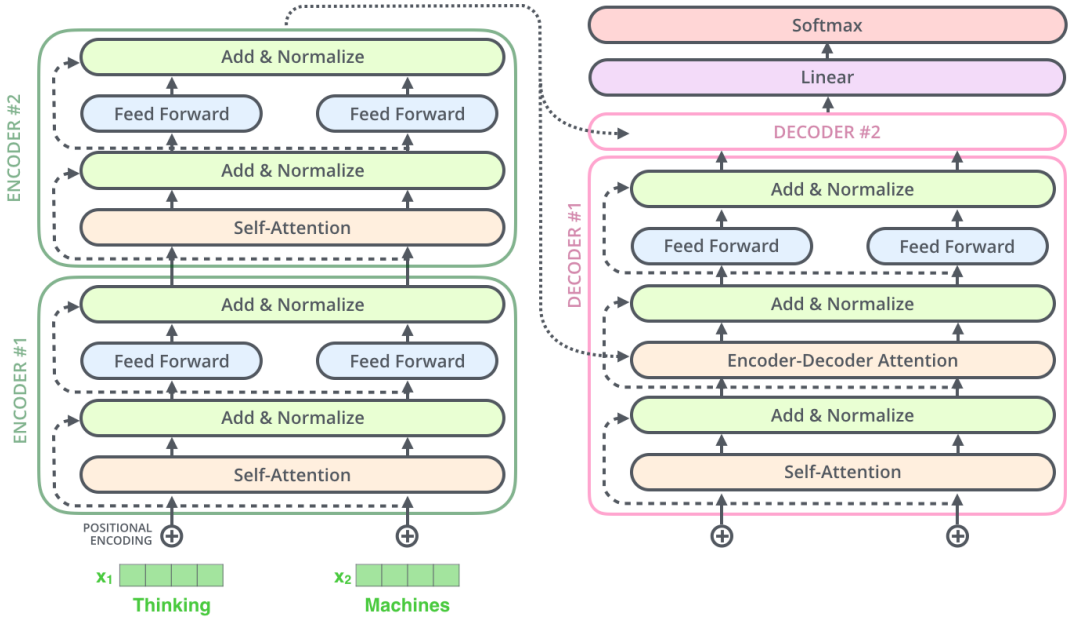

Figure 3. 2-Layer Transformer Architecture (Alammar, 2018)

The objective of the embedding layer is to encode the input as vectors of dimension d_model, after which these vectors are stored in an embedding matrix of dimension (vocabulary size, d_model). Both the encoder and the decoder embed data once (Alammar, 2018). The Transformer model does not recur. Therefore, the transformer uses positional encoding to describe the words' order in the input.

According to Figure 4, the encoder has a feed forward layer and a self-attention layer. When the data travel to the top encoders, these layers' residual

connections and layer normalization help the model retain the information from the input.

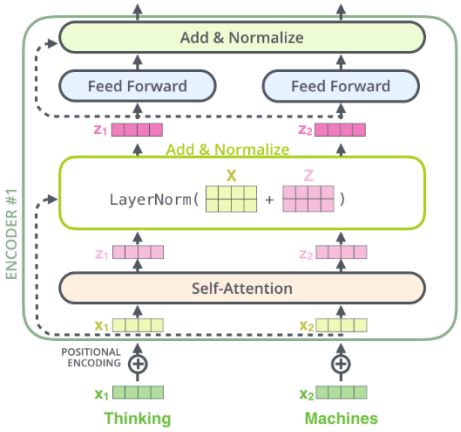

Figure 4. Encoder Structure (Alammar, 2018)

The transformer has N encoders, and N's default value is six. Self-attention layer generates its key, value, and query based on the preceding encoder output (Alammar, 2018).

Encoder-decoder attention is an additional layer that forms the query matrix in the decoder and contains the key and value matrices from the encoder stack. This layer enables the decoder in understanding its own input and seeing the encoder's input more clearly.

In the decoder, the self-attention layer functions differently. The previous words are only used from the decoder inputs to predict the next word (Alammar, 2018).

The transformer's final layer is the linear layer. It transforms the decoder's output vector into a logits vector. Logits are transformed into probabilities using the softmax. The highest probability is selected, and its index will be converted into a word (Alammar, 2018).

2.2. Machine learning approaches for correcting Arabic spelling mistakes

Azmi *et al*. (2019) developed a system that finds and fixes semantic mistakes in Arabic. The support vector machine classifier (SVM) determines if a word includes an error or not after the system has been trained by adding one artificial error to a sentence. The N-gram language model offers a candidate term to replace the word if it contains an error. The system received a 90.7% F1 score.

Alkhatib *et al*. (2020) used BiLSTM, word embedding, and a polynomial network classifier, where they presented a system for spelling and grammar error detection for word-level correction. They used simulated errors produced by linguistic knowledge algorithms to train the model. They gathered a set of 15 million words for testing, including with errors that were manually added and fixed by professionals. 93.89% of the F1 score was obtained.

Al-Qarghuli *et al.* (2021) created two transformer models: a Wiki model and a Tashkeela model to correct four types of soft spelling errors in Arabic, these errors include confusion between different hamza shapes, different alef shapes at the end of the word, the insertion and omission of alef after waw aljamaea, the confusion between teh and teh marbuta, and the final heh, also they used stochastic error injection approach for injecting errors in both wiki-40B and Tashkeela. They used Wiki-40B and Tashkeela datasets for training and evaluation and they achieved 97% accuracy on the injected wiki-40B with error injection rate 90%, and they achieved a character error rate (CER) of 0.86% on Test200 dataset which is a set that contains real soft spelling mistakes.

Abandah *et al.* (2022) created various bidirectional LSTM network configurations with two and four hidden layers, as well as with and without the use of the dropout approach. To train and test their models, they used the Tashkeela set, which represents classical Arabic, and the ATB3 set, which represents modern standard Arabic (MSA). They used transformed input and stochastic errors as their two training methods. Using the Tashkeela set with stochastic error injection technique, they were able to achieve a character error rate (CER) of 1.28%.

Aichaoui *et al.* (2022) used pretrained transformer model ARABERT, that was trained on corpus of 96 GB, and follow the same architecture of BERT Base, that has 6 encoder and 6 decoder layers and 768 hidden dimensions. They used Adam optimizer. They trained ARABERT transformer on their corpus named SPIRAL (SPellIng eRror corpora for Arabic Language) dedicated to the detection and correction of spelling errors in Arabic, and it is covering different types of errors. The results of training the model are in Table 1.

Table 1. Summary of Aichaoui et al. (2022) results

Categories	Accuracy	Precision	Recall	F1 score	Epochs
Morphology	0.889	0.802	0.802	0.802	2
Phonetic	0.898	0.816	0.817	0.816	15
Physical	0.843	0.730	0.730	0.730	3
Permutation	0.878	0.784	0.783	0.783	15
Keyboard	0.783	0.645	0.644	0.643	7
Delete	0.503	0.338	0.337	0.337	11
Space-issues	0.922	0.869	0.863	0.860	11
Tachkil	0.922	0.856	0.856	0.845	4

2.3. Machine learning approaches for correcting spelling mistakes in foreign languages

Zhang *et al*. (2020) proposed a soft-masked BERT to detect and correct Chinese spelling mistakes. Their model consists of an error detection network which is a bidirectional gated recurrent unit network (BiGR) and an error correction network that uses bidirectional encoder representations from transformers (BERT). Based on BERT, the two networks are connected via soft-masking technique. They achieved an F_1 score of 73.5% for detection and 66.4% for correction.

Tran *et al*. (2021) proposed a transformer model to detect and correct Vietnamese spelling mistakes using real-life texts. The model utilizes both character-level and word-level to detect errors and make corrections. The model achieved an F1 score of 68.88% for detection, and a correction accuracy of 96.01%.

Kuznetsov *et al*. (2021) introduced a transformer model for spelling correction. The model made use of a technique that creates synthetic spelling mistakes for several languages. The system gathered information from search engines and used a set of rules to produce mistakes. The accuracy of their model, when evaluated on four languages, was 83.33 percent for Arabic, 93.9 percent for Greek, 91.8 percent for Russian, and 94.48 percent for Setswana.

Zhao *et al*. (2021) used a transformer model to fix semantic mistakes in the Mandarin automated speech recognition (ASR) system, a transformer model was applied. They enhanced the ASR system's performance by correcting the output of the system using the bidirectional and auto-regressive transformer (BART) model. The CER of the ASR system that was merged with their model was 6.08%.

Samsudin *et al*. (2022) adopted a deep learning model that learns a character-level word mapping to a small embedding space while taking context into account. A word and all of its spelling variations are closely mapped in a Euclidean space in the

embedding space. By comparing the distances between their mappings and those of the correctly spelled words, they may spot and correct misspelled words. The method that the word embeddings are constructed captures the context of each word. This makes it easier to spot perfectly spelled terms that are used in inappropriate contexts (such as their/there and your/you're). Applications that use spell check benefit from their methodology.

Bijoy *et al.* (2022) proposed a denoising transformer-based detector-purificator-corrector framework for SEC of Bangla and low-resource Indic languages and name it DPCSpell. It merely fixes the wrong characters. Three networks make up the DPCSpell: a detector, a purificator, and a corrector. Also, they put forth a technique for building a sizable parallel corpus that addresses the resource-limitation problem for SEC (spell error correction) of any language written from left to right, such as Bangla, Hindi, or Telugu. Particularly, a sizable parallel corpus for the Bangla SEC is created and made accessible to the public.

Sun *et al.* (2023) proposed a Chinese spelling correction system. They suggested using an error-guided correction model (EGCM). They presented zero-shot error detection strategy to do a preliminary detection using BERT's potent capabilities. This method directs their model to focus more on the likely incorrect tokens in encoding and to avoid changing the correct tokens. Additionally, they developed a new loss function to incorporate the error confusion set, which helps their model to recognize tokens that are frequently misinterpreted.

2.4. Statistical approaches for correcting Arabic spelling mistakes

Mubarak *et al*. (2014) devised a technique that corrects Arabic spelling errors at the character levels. Their model looks up terms with comparable lexicons to those that have errors. They concentrated on errors that are challenging for the correction model to identify or repair. For the QALB dataset, the system received an F1 score of 63.43%.

AlShenaifi *et al*. (2015) established the Arib system, which employs a few algorithms and tools to identify and rectify Arabic spelling errors. 1) The MADAMIRA tool is used by the system to repair typographical problems involving the letters (ى, أ), 2) A rule-based corrector that changes English punctuation marks to Arabic punctuation, inserts a space after (على, إلى) prepositions and the letters (ة,ـة), if they merge with the next word, and eliminates letters that appear sequentially three or more times in a row in the same word, 3) The Bayes probability method is used to handle errors that can be repaired with more than one word. 4) The Ghaltawi tool, which has an external dictionary to fix mistakes. 5) This system also employs the Levenshtein distance technique. Arib earned an F1 score of 57.8% on the test.

Noaman *et al*. (2016) search a dictionary with 35.5k words. Specifically, if the word is not in the dictionary, it is considered an error, and a confusion matrix will generate a suggested word. The achieved accuracy is 89.69%.

Attia *et al*. (2016) improved the error model and language model, which are normally included in a spelling error detection and correction program. They mostly construct and adjust distance re-rankers and assess various mistake types. They also assess the degree of noise in various data sources and choose the best subset to train the system on. They built a tri-gram language model on strings for spelling error

detection. The N-gram language model corrects errors as they are found. Their system had a 93.64% accuracy rate.

Abdellah *et al*. (2020) employed the Levenshtein distance technique to fix Arabic text's errors of leaving a space between words. The method involves placing a space between the two words that have been combined. A dictionary of 30k words is used by the algorithm when the words still having mistakes. They used a collection of 1,000 words containing errors to test their system, and it had a 99.14% accuracy rate.

Table 2. Summary of the Literature Review

Author(s)	Language	Method	Metric	Score
Mubarak *et al*. (2014)	Arabic	Correction and Language Model	F1	63.43%
AlShenaifi *et al*. (2015)	Arabic	Multiple Algorithms and Tools	F1	57.8%
Noaman *et al*. (2016)	Arabic	Confusion Matrix	Accuracy	89.69%
Attia *et al*. (2016)	Arabic	N-gram	Accuracy	93.64%
Azmi *et al*. (2019)	Arabic	SVM and N-gram	F1	90.7%
Abdellah *et al*. (2020)	Arabic	Levenshtein Distance	Correction Rate	99.14%
Alkhatib *et al*. (2020)	Arabic	BiLSTM	F-Measure	93.89%
Al-Qaraghuli *et al.* (2021)	Arabic	Transformer	CER	0.86%
Abandah *et al*. (2022)	Arabic	BiLSTM	CER	1.28%
Aichaoui *et al.* (2022)	Arabic	ARABERT	Accuracy Precision Recall F1	Table 1 Summary of results
Zhang *et al*. (2020)	Chinese	BERT (transformer)	F1	66.4%
Tran *et al*. (2021)	Vietnamese	ALBERT (transformer)	Accuracy	96.01%
Kuznetsov *et al*. (2021)	Multiple Languages	Transformer	Accuracy	Multiple Scores
Zhao *et al*. (2021)	Mandarin	BART (transformer)	CER	6.08%

Samsudin *et al*. (2022)	English	Transformer	Euclidean distance	Their approach is effective
Bijoy *et al.* (2022)	Bangla	Transformer	F1	0.948
Sun *et al.* (2023	Chinese	BERT (transformer)	F1	77.5

Chapter 3

Data Collection and Handling

Any machine learning research should typically have a major part on data collection and processing before applying the different machine learning models on the data. These steps may differ from machine learning project to another. For example, the steps to be done for medical applications are different from those considered on natural language processing, and so on.

This chapter is divided into three sections. In the first section, the steps followed during data collection phase are presented. Then, I will talk about the steps I followed to prepare the data to be ready for applying machine learning models. Finally, I will give some statistics about the data and some conclusions.

3.1. Data Collection

The core part of any machine leaning project is usually done by applying several machine learning models on some collected data. After that, several performance tuning steps and procedures are followed. The performance tuning steps enhance the performance depending on the data under study.

If the data under study has some problems like biased data, small size, or not representative data, then the performance may not be good. And sometimes, even if high accuracy is achieved, the results are not logical and cannot be generalized to real life data. Therefore, the data collection step must be carefully done, must be well documented, and with high focus and preparation.

To get the data, there are usually three options: first, download ready datasets prepared by some researchers, who published their dataset online like the datasets that can be downloaded from Kaggle website. Second, prepare self-prepared dataset which may take very long time to prepare reasonable dataset with size suitable for training and testing. Third is a hybrid from both, where the researcher can download large dataset that can be used in training, and then the researchers can prepare their own dataset customized to their research problem and do more training on part of the data and testing on the remaining part. The third type will be used in this thesis.

The dataset that we will work on it, is composed of two parts: downloaded dataset from the Internet (WiKi-40B dataset), and self-collected dataset.

3.1.1 WIKI-40B dataset

The Wiki-40B dataset was downloaded from the Internet. Wiki-40B is multilingual language model benchmark that contains error-free 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. WiKi-40B is written in modern standard Arabic (MSA) (Guo *et al.,* 2020).

3.1.2 Collected dataset (REALMS)

Since our research problem is mainly about correcting Arabic errors, the data that must be used in the training and testing phases must have some errors as were mentioned previously in Chapter 1. Therefore, it is not enough to get a dataset that has correct Arabic sentences and words.

To prepare a dataset that contains errors, it is not suitable to insert the errors by hand because it will be subject to biasing. Therefore, we collected real dataset that is composed of sentences and paragraphs typed by different people over different

social websites like Twitter and Facebook. The following explains how these steps were done.

Firstly, I used a dummy Twitter account that has no friends and does not follow anyone, any journals, or others. This is to avoid any kind of biasing like downloading tweets from only the author's friends who may have similar writing behavior. Instead, using the new account, we only downloaded tweets and their comments from public groups and from public hashtags.

To automate this step and be able to download hundreds or even thousands of real tweets and their comments per day, I created a Twitter Developer account which allows me to use Twitter's APIs functions. Creating such kind of account requires contacting Twitter and send them emails to explain some issues like: the purpose from using Twitter APIs, how the collected tweets will be used, and whether the privacy of the users will be affected or not, and some other issues.

After creating the developer account and getting the permission, I used some scripts to download the tweets and their comments. These scripts allowed me to download tweets which contain some keywords or hashtags after specifying the number of tweets that the user chooses. For example, using the scripts, I downloaded tweets according to some random words that came to my mind like:

- أوكرانيا: 98 tweets
- حزيرة: 36 tweets
- كورونا : 98 tweets

Regarding the tweets which use hashtags, I tried to use hashtags that may involve people from around the world like (#فلسطين_قضيتي، #الرئيس_الاوكراني), as these

words are general and many people around the world may participate in such hashtags. This may help in downloading statements which may contain errors from people of different nationalities. Specifically, I downloaded:

- 2080 tweets for #فلسطين قضيتي
- 2059 tweets for #. الرئيس_الاوكراني

In addition to the above, I used a special website (https://exportcomments.com/) dedicated to export social media comments which exports comments on any post in social media websites like (Facebook, Twitter, Instagram, etc.) to an Excel file. The free version of this website allows downloading around 100 comments for each post, and up to 5 posts per day. Therefore, at most, 500 comments can be downloaded per day for free.

I used the "export comments" website to download comments from Twitter as follows: 366 comments from some news pages and for public journalist's posts and classified them as political comments, and 100 comments for Islamic site posts. With these 466 comments from Twitter, number of comments and tweets I downloaded from Twitter became 4837 rows distributed as shown in Figure 5.

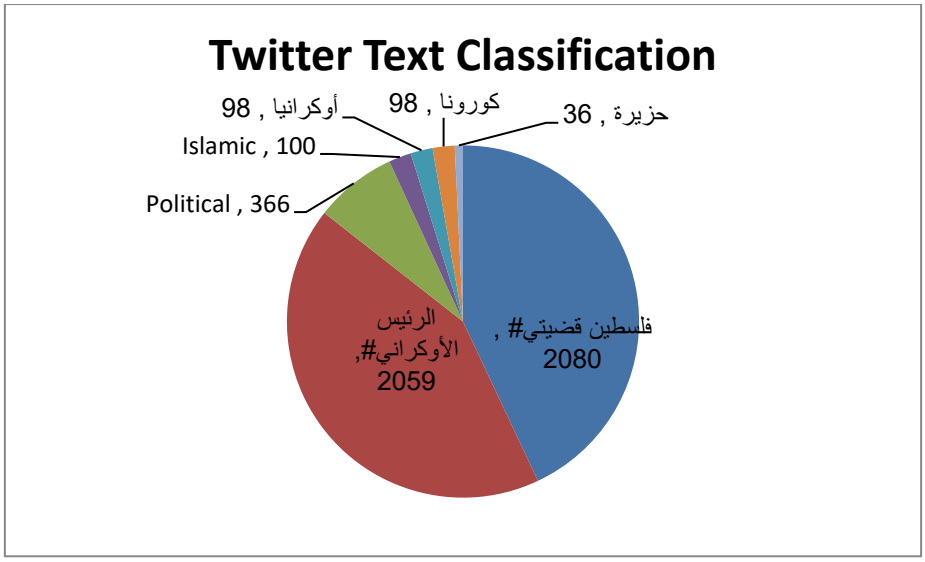

Figure 5: Classification of records collected from Twitter

Regarding Facebook website, I downloaded comments from Facebook using the same website that is mentioned above. I downloaded comments on posts from different fields; economy (198), Islamic (397), political (2947), sport (966), literature (396), documentary (495), and social (101). Accumulatively, I collected 5500 rows from Facebook platform as shown in Figure 6.

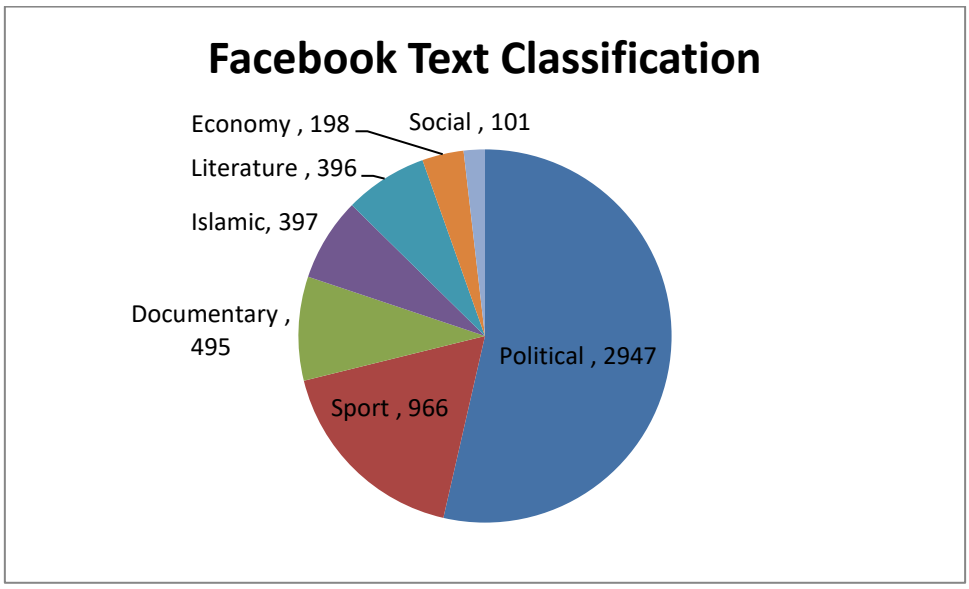

Figure5 : Classification of records collected from Facebook

Finally, the data collected from Twitter using the two methods mentioned above, along with the comments collected from Facebook were stored in one Excel file. In total, I have collected 10,337 rows with 31 columns as shown in Figure 6. All these attributes for each record are downloaded automatically with each comment or tweet.

Most of the features are self-explanatory except the following: Type column which contains the source of each text whether it is from Facebook, twitter, or one of the hash tags that I searched on. Also, the UTC (Universal Time Coordinated) column which is a standard used to set all time zones around the world. So, for instance, New York City is in the time zone UTC minus five, meaning that it is 5 hours earlier in NYC than the reading on a UTC clock (except during U.S. daylight savings, when it is 4 hours earlier). The main text of the tweets and comments are in the column "Text".

```
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   Type                            10337 non-null  object
 1   Name                            9923 non-null   object
 2   User Id                         9512 non-null   object
 3   Username                        4837 non-null   object
 4   Tweet/post ID (click to view url)  4905 non-null   object
 5   in_reply_to_status_id           33 non-null     float64
 6   Author Location                 2463 non-null   object
 7   UTC                             4139 non-null   object
 8   Created At                      4139 non-null   object
 9   Date                            6138 non-null   object
 10  retweet_ count                  4494 non-null   float64
 11  Favorites                       10046 non-null  object
 12  Comments                        18 non-null     float64
 13  Is Retweet?                     310 non-null    object
 14  Author Followers                2369 non-null   float64
 15  Author Friends                  310 non-null    float64
 16  Author Favorites                2458 non-null   float64
 17  Author Statuses                 2369 non-null   float64
 18  Author Bio                      3589 non-null   object
 19  Author Image                    310 non-null    object
 20  Tweet Source                    310 non-null    object
 21  Language                        4139 non-null   object
 22  Client                          4139 non-null   object
 23  Tweet Type                      4139 non-null   object
 24  Status/post URL                 809 non-null    object
 25  Hashtags                        4139 non-null   float64
 26  Mentions                        4139 non-null   float64
 27  Media Type                      2029 non-null   object
 28  Media URLs                      2029 non-null   object
 29  User URL                        555 non-null    object
 30  Text                            10280 non-null  object
dtypes: float64(9), object(22)
```

Figure 6 : Collected dataset description

3.2. Data Processing

In the previous section, we described the steps followed during the data collection phase. In this section, I exaplain the steps followed to prepare the dataset after being collected. All of the processing steps are done on the Column “Text” because it has the downloaded texts.

The steps that were followed in dealing with the data are:

1) Clean the texts, by removing null rows, remove duplicates and remove symbols.

2) Classify the texts into slang and formal texts, and remove the slang texts.

3) Label the remaining formal texts with either has spelling mistakes or no spelling mistakes.

4) Finally, for the formal texts which have spelling mistakes, correct the spelling mistakes in a new column.

As a first step in cleaning, I handled rows which has empty value for the “Text” column. As shown in Figure 6, the “Text” column in the dataset contains 10280 non-null texts. Consequently, there are 57 rows with null Text values. These rows were dropped because the Text attribute is the main attribute which we want to study. Any row has no value for this attribute, does not have any benefit, and as a result, it is dropped.

As a second step of data cleaning, is duplicate removal. There was a lot of duplicates in the dataset (around 25%), especially in hashtags group of tweets. Therefore, I removed the duplicates from the text column, and the dataset is left with 7715 rows.

The third cleaning step is cleaning the Text column from different shapes of emojis, URL, Arabic and English hashtag #, and @username. To keep the original texts untouched, the result of cleaning is stored in a new column called ‘Text cleaned’ and it has 7714 text values. This is because after data cleaning, one row became empty, and I dropped this row. Similarly, after the cleaning step, some rows became duplicates. After removing the new duplicates, the remaining dataset became composed of 7165 rows.

The second step after cleaning the data is classifying the Text into formal and slang. To do this, we counted the number of the slang words in each row. Then we calculated the percentage of the slang words to the total words of Text in each row. After that, the row was classified as a slang row if more than 20% of Text words are slang words. Otherwise, it is considered as formal. The slang text is filtered out and is not corrected, as my aim in this thesis is to correct the spelling errors in the formal texts.

To simplify the step of counting the slang words in the text, I prepared a graphical user interface (GUI) tool. The tool displays each time one text row of the data, "+" button, and "-" button. Once the text is shown, I start reading the text, and press the "+" button for each slang word which increases the number of slang words in the text. Once I click on the "save" button, it stores the number of slang words for this row in a column in the dataset, and displays the next row, and so on. This tool is shown in Figure 7.

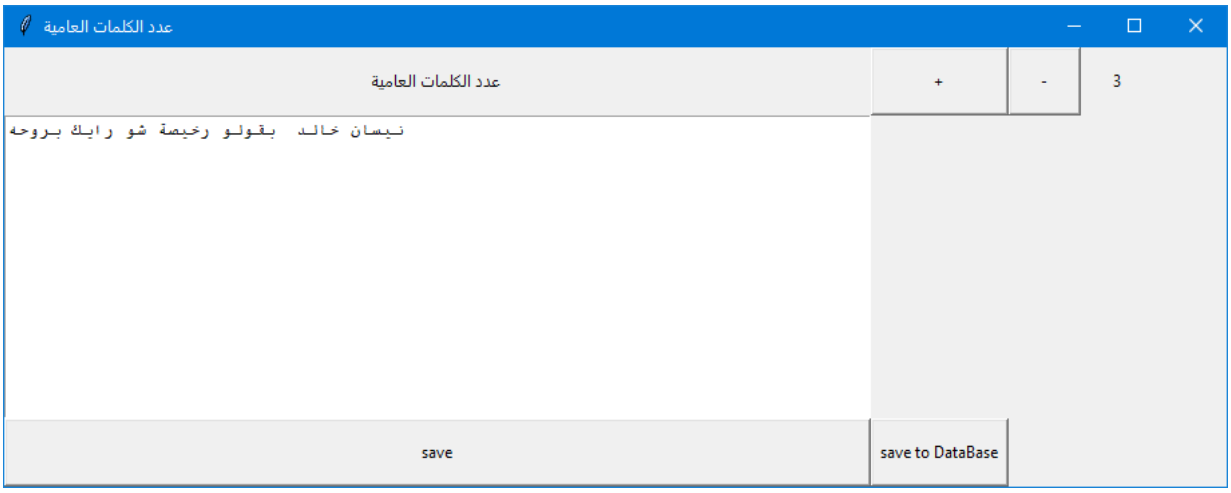

Figure 7 : Slang words counting tool

This classification step is time consuming because I have to go over all the rows, read them, count number of slang words, and save them. Moreover, many words are confusing and hard to decide whether they are slang or formal, which complicates this step and makes it more time consuming.

During this step, if the text is written in any language other than the Arabic language, contains only one Arabic letter (e.g., 'ت'), or has any sequence of same Arabic like (هههههه) which express an audio expression that means that the person is laughing, then, I set the counter to -1. These kinds of texts have no meaning in our work, and I dropped them from the dataset later. Additionally, I put 0 for the texts that

are formal and has no slang words. The result is 443 rows are marked with -1 (i.e., should be deleted).

Then, to calculate the percentage of slang words in each row, we count the number of words per row. To calculate the total number of words of each row, we remove the leading and trailing white spaces in each text because they make the counting wrong. Similarly, the spaces before the punctuation marks are removed. Then, the built-in functions in Python were used to do the counting. Removing these extra spaces made some rows as duplicates. After duplicates removal, the dataset was left with 7046 cleaned text rows. Also, after removing the 443 rows that has -1 count in the slang column, the dataset size became equal to 6603 rows. In conclusion, the slang column has the counts shown in Figure 8.

```
df['Total Slang words '].value_counts()
```

```
0.0     4722
1.0      687
2.0      483
3.0      287
4.0      135
5.0      110
6.0       54
8.0       28
7.0       24
9.0       18
11.0      14
10.0       8
12.0       5
17.0       5
13.0       5
14.0       4
16.0       2
22.0       2
44.0       1
15.0       1
27.0       1
20.0       1
23.0       1
25.0       1
26.0       1
30.0       1
35.0       1
41.0       1
Name: Total Slang words , dtype: int64
```

Figure 8 : Counts of slang words after removing 443 rows

Then, I calculated the slang percentage of each row. To do this, we divided the number of slang words by number of words in the row, and multiplied the result by 100%. If the percentage = 0, it means that these rows have no slang words in them,

and I will consider them in my work. If the percentage is less than or equal to 20%, I also work on them and I coded them with 1. The rows that have slang percentage greater than 20% are not of interest in this thesis. Therefore, these are dropped from the dataset. The result is shown in Figure 9.

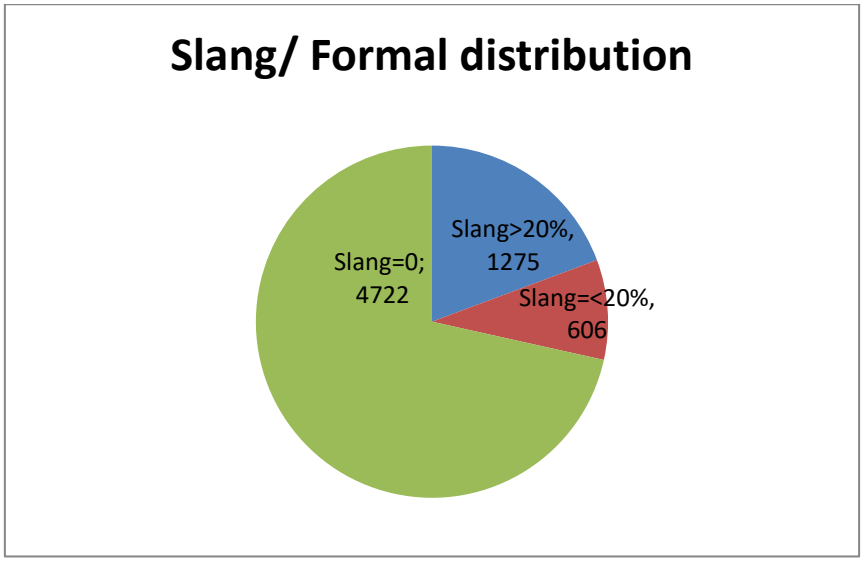

Figure 9 : Formal slang words classification

After the slang sentences (i.e., those have percentages greater than 20%) were dropped, we are left with 5328 rows. These rows are 4722 rows of pure formal and 606 that contain 20% or less slang words.

3.3. Text Correction

In this stage, I worked on correcting the text errors through reflecting the corrected text in a new column. To facilitate this, I created a tool for showing the text in a cell and in the other cell I can correct the text as shown in Figure 10.

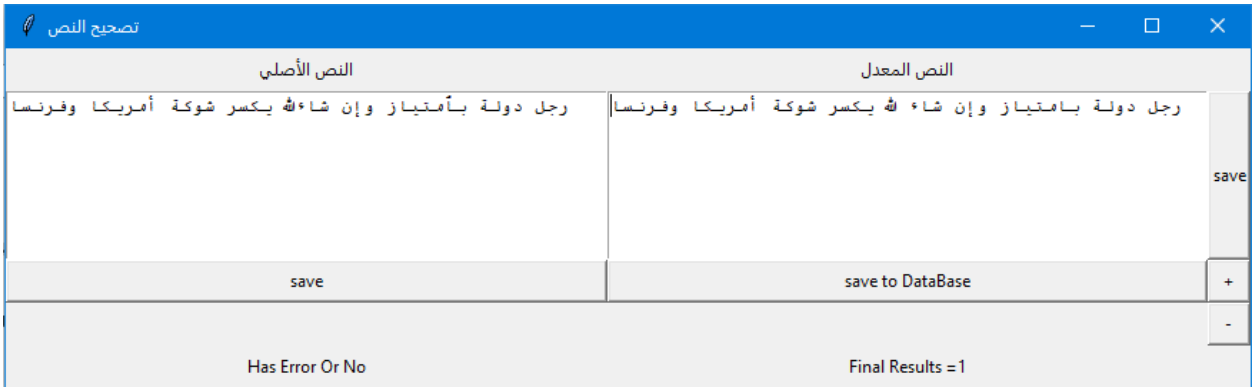

Figure 10: Correcting spelling mistakes tool

I started inspecting the dataset row by row to correct the spelling mistakes (if exist). Specifically, if the displayed text has no spilling mistakes, I mark it with 0. However, if the row has one or more mistakes, I mark it with 1. These marks are in a separate column called (error or no?).

During the correction step, I faced several cases which urged me to modify the original texts like:

1) (يوني h ص): there are many comments which are written like this. The person who writes this way is trying to work around the social network filtering in order not to filter out his/her comment. I correct it to be (صهيوني) in both the original text cell and the corrected cell.

2) Several slang words have formal corresponding word and need simple modification to convert them into formal words. Examples on these words are: the word "مين" which means "who", I considered this mistake as an adding letter which is the middle "ya'a" "ي", so once I delete this letter, the word became correct and formal ("من"). A second example on this case is the word "وين" which means "where", it should be written like "أين" so I considered this error as a replacement error to replace Alef instead of waw.

The column “error or no?” gives an insight about the ratio of formal texts without errors to the total formal texts. Specifically, the results are 2504 rows with errors, and 2218 rows without errors out of the total 4722 formal rows (as shown in Figure 11). This means that more than half of the formal texts have writing errors. This is strong evidence for the need for a tool that helps in correcting Arabic writing errors like the one we are working on it in this thesis.

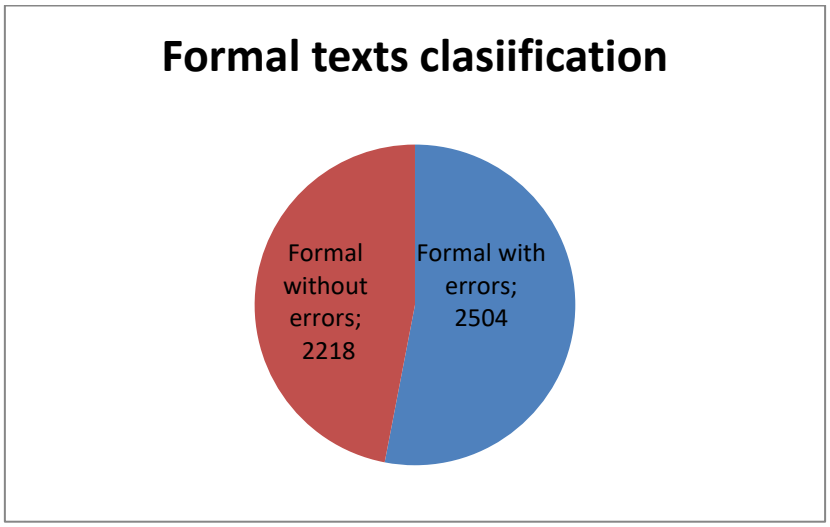

Figure 11 : Formal text mistakes distribution

In summary, the total rows are 5328 rows divided into two parts: 4722 formal text rows and 606 rows which have percentage of slang words 20% or less. All the 606 slang sentences have errors. Therefore, the total number of rows which have errors is (2504 + 606 = 3110 rows) out of the 5328 rows as shown in Figure 12.

```
[4]  df['error or no'].value_counts()

     1.0    3110
     0.0    2218
     Name: error or no, dtype: int64
```

Figure 12 : Total number of rows which have errors is 3110

After that I started a long step of counting the errors in each row, correcting the errors in the 3110 rows which have errors, and classifying the errors into their

main types. Indeed, inspecting the data several times, revealed to me the errors and helped me in adopting a classification for the errors divided into 24 types as shown in Table 1. Additionally, I counted the number of errors from each type in each row as shown in the last column of Table 3.

Table 3. Types and number of errors in each type

Type of Error	Example on error	Correction	Count
Hamzat Qataa	**الامر**	**الأمر**	5878
Writing Ha'a instead of Taa marbotah in the end of the word	الأند**يه** الواله**ه**	الأند**ية** الوالد**ة**	1196
Substitution	**ط**نهض **ظ**رب **اظغط** **سمعتو**	تنهض ضرب اضغط **سمعته**	1196
Missing space	الأمريكيةلكن توافقلك أما **مايحصل** في أوكرانيا	الأمريكية لكن توافق لك أما **ما يحصل** في أوكرانيا	1175
Slang words	**حنا** بلاد المسلمين **اللي** الدول المتطورة **تخليك**	نحن بلاد المسلمين **الذين** الدول المتطورة **تجعلك**	1110
Missing letter	كل شي مخدوم رجعوه	كل شيء مخدوم أرجعوه	710
Adding letter	ليها بيه معاه اللتي	لها به معه التي	692
Adding space after Conjunctive (WAW AL-Atf)	**و يمكرون**	**ويمكرون**	449
Hamzat wasel	**احسن** من	**أحسن من**	232
adding or missing WAW ALJAMAA	يقو**لو**	يقو**لوا**	231
Hamzet Madd	الاخر الان	الآخر الآن	197
Hamzah mutawassetah	امراءة	امرأة	139
Hamzah mutatarrrefah	يهزء	يهزئ	69
Writing Taa marbotah instead of Ha'a in the end of the word	المياة اللة	المياه الله	38
Writing Alef mamdodah instead of alef maqsora	لكل المرء ما **نوا** لو ملكت كنوز قارون **ستبقا**	لكل المرء ما **نوى** لو ملكت كنوز قارون **ستبقى**	27
Inversion (writing two adjacent letters in the word instead of each other)	ينساو الهنب العربية	ينسوا النهب العربية	26
Writing teh instead of the marboutah	هذا يقاتل تحت رايت من؟ هزيمت	هذا يقاتل تحت راية من؟ هزيمة	26
Deletion of AL-Altareef	من أكثر التجارب متعة بنسبة لي كل هذه الانتصارات لجيش الروسي	من أكثر التجارب متعة بالنسبة لي كل هذه الانتصارات للجيش الروسي	17

Writing teh marboutah instead of teh	المستشفياة الأجندة التي **وضعة** له	المستشفيات الأجندة التي **وضعت** له	12
Deletion of AL-Lam AL shamseiah	وتنكرون لشجر والبشر لكن انظام متخاذل يتمتع بأموال اشعب اروسي	وتنكرون للشجر والبشر لكن النظام متخاذل يتمتع بأموال الشعب الروسي	10
Writing Alef maqsora instead of alef mamdodah	فالله دعى فلا أجر ولا عقاب إلى إذا تحققت النية	فالله دعا فلا أجر ولا عقاب إلا إذا تحققت النية	9
Adding alef after tanweeen the words that end with hamzah	اتخذوه غطاءا رجاءا هباءا	اتخذوه غطاء رجاء هباء	7
Adding alef & noon after tanween fath	تتصدق **سراان** عنده حق **طبعان**	تتصدق **سرا** عنده حق **طبعا**	2
Writing noon instead of tanween fath	**فعلن** هذا هو الفشل أفضل لاعب **حالين** بمصر	**فعلا** هذا هو الفشل أفضل لاعب **حاليا** بمصر	2
		Total errors	13,440

To automate the step of correcting, classifying, and counting the errors, I created a GUI tool that has two cells showing the text before correcting and the text after correcting. Additionally, it has plus and minus buttons to count the mistakes in each row for each kind of the error. There is also a button labeled “Save” which saves the total number of errors in a new column called (Total of mistakes) and stores number of errors of each type of error also in a separate column according to the classification. Table 4 summarizes the number of rows in the dataset after each processing step. The collected data is named (REALMS) which stands for REal Arabic Language Mistakes in Spelling. This concludes the steps done in dataset collections and preparation.

Table 4. Summary of the collected data after each step

Step		**Number of rows**
Total data collected		10337
From Facebook = 5500	From Twitter = 4837	
Removing null texts		10280
Removing duplicates		7715
Removing emoji, photos, Arabic & English hashtags, URL, and usernames		7714
Removing duplicates again after last step		7165
cleaning spaces and remove duplicates again		7046
Removing none-Arabic text		6603
Removing the rows which have more than 20% slang words		5328

Rows which have errors	3110

3.4. Evaluation Metrics

To evaluate our models, we use the following metrics: accuracy and CER. Accuracy is a general metric that measures the overall performance of a model. CER is more specific metric that measures the performance of the model for specific tasks such as speech recognition, machine translation and spelling correction.

3.4.1. Accuracy

Correcting spellings can be seen as a categorization task. A model can generate four different types of predictions in this type of task. These categories include false positive (FP), true positive (TP), true negative (TN), and false negative (FN). The higher the accuracy, the better performance of the model, Equation 1 illustrates how the accuracy calculated.

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \quad (1)$$

3.4.2. Character error rate (CER)

CER is a widely used indicator of how well an automatic speech recognition (ASR) system is performing. The ratio of erroneous characters in the text is how CER calculates character errors. The following equation can be used to get the character mistake rate:

$$CER = \frac{S+D+I}{N} \quad (2)$$

where N is the number of characters in the target sequences, S is the number of substitutions, D is the number of deletions, I is the number of insertions, and D is the number of deletions. When the CER value is low, the model performs well.

Chapter 4

Experiments Setup and Results Analysis

This chapter is divided to three parts. The first part describes the simulation environment setup that was used to perform the experiments and extract the results. It also sheds the light on the challenges when dealing with this environment. The second part presents the steps followed to perform training and testing. The last part presents and analyzes the experimental results.

4.1. Simulation Environment

Performing machine learning experiments which includes huge datasets and use heavy machine learning models like transformers requires very powerful machines. We started performing the experiments on a local machine which has i7 processor and 8 GB RAM. However, it was impractical due to the very long time needed for training and testing. Also, it is high cost to buy powerful machines which contain accelerator processors like GPU and TPU and large memory sizes.

The other solution that we used is to use a cloud service like Kaggle and Google Colab. These cloud services, have two types of service: these are free service and paid service. In the free service which we used, we may use some basic processors for long time. But, to use GPUs and TPUs, there is limited number of hours per week, and they can be used according to their availability. The paid service introduces longer service time on GPUs and TPUs. But you pay based on the needed service time and the required machine capabilities.

We performed our experiments on Kaggle cloud service. Specifically, we used the free service which provided us 20 hours per week on a TPU processing unit, and

30 hours per week on a GPU processing unit. Also, it gives unlimited time on regular CPUs. Kaggle TPUs are equipped with 128 GB of high-speed memory and 8 cores, while Kaggle GPUs are NVIDIA TESLA P100 that come in 16 and 32 GB configurations.

Performing the training and testing on Kaggle machines has great advantages like free powerful machines, can be accessed from anywhere, and the files and codes are not prone to get lost or corrupted because they are stored on the cloud. However, the work on Kaggle free servers rather than on local powerful computer with either GPU or TPU has several challenges:

1) It gives only limited time per week, if the available time gets exhausted, we must wait to the next week to continue the experiments. It is very regular to exhaust the free available weekly time when dealing with huge datasets. Also, many experiments (especially training) may take more than 10-20 hours to finish which may be broken because the available time is over. Additionally, each single session time on Kaggle TPU accelerator must not exceed 9 hours long, and for the GPU accelerator is 12 hours maximum. Thus, we need to divide an experiment on multiple sub-experiments to finish. For example, do several epochs per session.
2) The GPU and TPU sessions are granted based on availability. For example, if I need a TPU to perform an experiment, I should start the TPU first, then run the experiment. However, if at that moment, there is no available TPU, Kaggle will put me on a queue until a TPU is available. The waiting time in the queue is dependent on the number of people in the queue, the time consumed by each one in the queue, and even the service time needed by each one who is using the TPU currently. Many times, we must wait for several hours before getting assigned a TPU.

3) To deal with the dataset, it must be uploaded to Kaggle. Also, every time we want to modify the dataset using local programs, we must download the dataset, process it using local programs (*e.g.*, MS Excel), then upload it again. This is a high cost in terms of bandwidth and time, especially with datasets of big size (as large as hundreds of Mega Bytes or few Giga Bytes). This process needs to be repeated for different code files.

4) Every time we run code that needs installing package, the package must be installed to run the code. This consumes some extra time. This is usually not required on local machines, because when the package is installed, we do not need to install it again.

5) It is harder to debug the code when write the code on Kaggle notebook compared to when use code editors like Anaconda.

But, even with these challenges, it is still very beneficial to use Kaggle machines because there is no way do the experiments on regular PC, and it is very expensive to buy strong machine for only performing a project.

4.2. Data Preparation

Several steps have been done on the dataset to prepare it for the transformer model. First, as the dataset that we have collected is considered not enough to train a large model, we trained the transformer on Wiki-40B dataset. Then, we trained the transformer on our own collected dataset. Some preparation steps are the same for both datasets, however some steps may be done on one of the datasets only.

This section is divided into two subsections: the first talks about the steps that were done on wiki dataset, and the second talks about the steps that were done on our collected dataset.

4.2.1. Wiki-40B preparation

The steps that were done on the Wiki-40B dataset are as follows:

1) Wrapping long lines: for rows of text which contain more than 300 characters length, we divided them into multiple rows of maximum length 300 characters. If character number 300 is in the middle of a word, we ended the row on the last word before 300 characters such that we do not split the word into two rows. This step is required to speed up the training and testing phases. The disadvantage of this step is it may break the context of the rows. However, 300 characters is considered a good tradeoff value that was tested and adopted by the previous work of (Abandah *et al.* 2022). After this step, Wiki-40B dataset became composed of 3,845,520 rows in the training set, 218,316 rows in the validation set, and 215,987 rows in the test set.
2) Then, we cleaned the data from any character that is not an Arabic character like punctuation marks, numbers, special characters, other languages, and diacritization. These steps were done on the training, validation, and testing sets.
3) Error Injection: the size of Wiki-40W dataset is good enough but has no spelling mistakes. Therefore, we decided to inject it with spelling mistakes. Manual spelling error injection is not possible and may be subject to biasing. Therefore, we did statistical error injection on the wiki dataset. But we did not inject our collected dataset with any errors to keep it more real and not touched. The errors injected in the Wiki-40b dataset are classified in the following classes:

Class A: Previous work errors

Our work is considered an extension to (Al-Qaraghuli *et al.* 2021) work, where we fix the errors, they have fixed, in addition to other spelling errors. Their fixed errors, which we are also targeting to fix, are summarized by the following:

- Hamza = (['ا','ء','ئ','ؤ','أ','إ','آ','اء'])
- Taa last = (ت، ـة، ـه)
- alef_last = (['ى' ,'ا'])
- Waw AL-jamaa = (['وا' ,'و'])

Therefore, each met hamza in the text is replaced with probability 30% with any shape of hamza shapes in the set given above and kept the same with probability 70%. Also, any haa or taa at the end of the word is replaced with one letter of Taa last set with probability 30% and kept the same with probability 70%. The same for alef at the end of the word and waw aljamaa.

Class B: Phonetically errors

In this class of errors, some characters have similar sounds, and according to our study of spelling mistakes on the collected real data, many people write them incorrectly by mixing them. These letters are shown in Figure 13, where these sets are surrounded by the same shapes (triangle, star … etc.). Therefore, each time one of these characters is found in wiki dataset, it is replaced by any character from the same set with probability 30% and kept the same with probability 70%. The sets of the characters are as follows:

- Seen = (['س','ص'])
- Dad = (['ض','ظ'])
- Dal = (['د','ض'])
- Taa = (['ت','ط'])
- Thal = (['ذ','ظ'])
- Zain = (['ز','ذ'])

Figure 13: Arabic phonetic letters on the keyboard

Class C: Keyboard errors

Many Arabic characters are different in one dot only, and they are very close to each other on the keyboard. Also, when anyone prints those characters mistakenly, it is hard to be detected because they are very similar, especially when the font size is small. These letters are shown in Figure 14. Similar to what have been done with the errors above is done here, where each character is replaced by one character from the same set with probability 30% and kept the same without change with probability 70%. These sets are as follows:

- Jeem= (['خ','ح','ج'])
- Baa = (['بـ','يـ'])
- Noon = (['تـ', 'نـ'])
- Ein= (['غ','ع'])
- Faa = (['ق','ف'])
- Sheen = (['س','ش'])
- Raa = set(['ر','ز'])

Figure 14: Arabic keyboard similar and close letters

Class D: third pronoun errors (Ha'a AL-Ghaeb)

A very popular error happens these days is replacing the third pronoun (i.e., Haa Alghae'b) with waw like (سمعو يتكلم) instead of (سمعه يتكلم). Therefore, we added a set: HaLast = set(['و','ه']), and injected instead of each last 'haa' with 'waw' with probability 30%.

Class E: Errors of removing space between words

Arabic letters can be divided into continuous letters and discontinuous letters. The discontinuous letters should be disconnected from the next letter in the same word, these letters are (ا، د، ذ، ر، ز، و). The continuous letters are the remaining letters which must be connected to the next letter in the same word. Many people when write a word ends with a discontinuous letter, they connect this word with the next word and remove the space between the words. The reason is that people can still easily read the two words, but it is considered a spelling error. Example on this error is (ينقذطفلا). To mimic this very common error, we removed the space with probability 30% after any word that ends with discontinuous letter.

Class F: Arabic preposition

Replace some popular patterns errors: many people when write (على ال) replace it with (ع ال) like (ع البيت), and replace (في ال) with (ف ال) like (ف البيت).

Class G: Error Generalization

A last set constitutes of all characters, in which we replace each character with any other character with low probability like 2% or 5%. The probability is lower here because we want to reduce the probability that we change two letters from the same word. This case is a generalization for all types of spelling mistakes where the person may print a character instead of another character by mistake.

We name all classes above except errors in class G as “intelligent errors” because they happen frequently due to specific reasons as was mentioned above. The last set is called “random error”. We used 30% injection probability for intelligent errors. Table 5 is a sample of the data after injecting intelligent errors with probability 30% and random errors with probabilities 2%, 5% and 10%.

Table5 . Sample of wiki data after injection with errors with different random errors probability

The correct sentence	**Sentence after injection 2% random errors plus intelligent errors**	**Sentence after injection 5% random errors plus intelligent errors**	**Sentence after injection 10% random errors plus intelligent errors**
شيخار خان وهو مصمم الألعابالإلكترونية الذي يعمل في شركة صناعات بارون ومقرها لندن وقد تلقت الشركة العديد العديد من الإخفاقات التجارية ولكن شيخار لم يلق له الأمر وبدأ بصنع لعبة تهزم العلامات التجارية الأخرى بصنع لعبة يكون فيها الشرير القوي وسمى الشخصية الشريرة راوان	**شيجار** خان وهو مصمم **الالعاب الإلكتروتية** الذي يعمل في شركة **صنآعاة** بارون ومقرها **لتدن** وقد تلقت **آجشركت العضيد** العديد من الإخفاقات **إلتجارية** ولكن شيخار لم يلق له **النمر ويدأ بسنع** لعبة **نهزم وٰلعلاماه نلتجارية الأخرا** بصنع **لعبه** يكون **فبها نلشرير** القوي وسمى **الشحصية** الشريرة را **وإن**	شيخار **حان** وهو مصمم **الاءلعاب الإلكترونيه** الذي يعمل في شركة **سناعات** بارون ومقرها لندن و**��د تلقة** الشركة العديد **الغديد** من **الإخفاقنت ألنجارية** ولكن شيخار لم يلق له **الاءمر ويدأ** بصنع لعبة تهزم **العلا��إت** التجارية **الأخرا** بصنع لعبة يكون فيها **إلشرير** القوي وسمى **الشخصبة الرريرة** را وان	**شيخءر** خان **وهوا** مصمم **اءللالغاب ادإلكترونية ءصذي يعتل** في شركة **ستاعات ب ءرون** ومقرها لندن وقد **نلقت** الشركة العديد **العضيد** من **آطإخفافات التحنرية وككن عيخار قم** يلق له **اءلأمر** وبدأ بصنع **لغبة تلزم العلاماه التجاريه اوأخرى** بصنع لعبة يكون **قبها** الشرير **وٰلقأي وسجى** الشخصية الشريرة **سى** وان

وبجانبها الشخصية الطيبة والتي سماها جي وان لكن في حفل افتتاح اللعبة حدثت مشكلة بعدما لعب ابن شيخار باللعبة ولم يكمل لعبته بحيث شخصية را وان وعد ابن شيخار أنه سيخرج من اللعبة إلى العالم الحقيقي ليقتله وهذا ما حصل خرج را وان من اللعبة وقتل شيخار وظل يبحث عن ابن شيخار براتيك	**وبجاءتيها** الشخصية **اءلطً◌بت** والتي **سماءها** جي وان لكن في **خفل أفتتاح نللعبة** حدثت مشكلة بعدما لعب ابن **شبخار** باللعبة ولم يكمل **لعيته** بجيث **شجصية رى وءن وغد** ابن **ظيخار** أنه سيخرج من اللعبه إلى العالم **اءلحقيفي ليفتله** وهذا ما حصل خرج را وان من اللعبة وقتل **شيحنر** وظل **يبجث غن** ابن شيخار براتيك	**وبخانبها الشجصية الطيية وآلتي** سماها **حي** وان لكن في **جفل** افتتاح اللعبة **جدثت بقكلة** بعدما لعب ابن شيخار باللعبة ولم **بكمل** لعبته بحيث **شخصيه** را وان وعد **آين شيجار أتت سيجضج** من **اللغبة** إلى العالم **الخفيفي ليفتله** وهذا ما حصل خرج **رى** وان من اللعبة وقتل **شيجار وضل غن إبن** شيخار براتبك	**وبخنبها** الشخصية **وٌطبية** والتي سماها جي وان لكن **قي** حفل **وٌفنتاح** اللعبة حدثت **زلكلة بعدمى** لعب **اين** شيخار **باءيلغبة** ولم **يعرل لغبتة** بحيث شخصية **سا ووٌن** وعد **أبن** شيخار أنه سيخرج من اللعبة إلى **آلعالم وٌلحقيقي** ليقتله وهذا ما حصل خرج را **زان** من **اثلعية** وقتل **جيحار** وظل يبحث عن ابن **شيحأر** براتيك
قام بتجميع هذه الأشعار فريق مكون من أربع شعراء تحت إشراف كي نو تسورايوكي	قام **بنجميع هظه ألاءشعار** فريق مكون من أربع **شغرآء** تحت **ءشراف** كي نو **تصسرابوكي**	قام **يتحميع هذة الأشعار** فريق مكون من **اريع** شع◌ً◌اء تحت **إضراف** كي نو تسورايوكي	قام **بتخليع** هذه **اخآظعار** فريق مكون من **اءربع معراء** تحت **اءشراف** كي **تو تسوراءيوكي**

As can be seen in the table above, mixing all errors may produce corrupted text that may result in texts that are very hard to correct and understand even by human, and if we choose more than 5% for random errors, this will produce rubbish text, so that we choose either 2% or 5% random error rates. Indeed, achieving high accuracy in fixing the spelling errors with 5% random errors in addition to the intelligent errors is a challenging issue, where many words are very hard to fix.

4.2.2. Our collected dataset preparation (REALMS)

First, we split the dataset into train set and test set using stratified shuffling with 20% for testing and 80% for training. When doing the split, we prefer that each of the 24 types of errors, mentioned previously, to exist in the training set and in the testing set. Each row may have several types of errors. Therefore, to do efficient stratified split, we created a new column in the dataset with name class. For each row, I assigned the class attribute a value equal to the type of error that exist in that row which is the least frequently error in the dataset. More specifically, we arranged the columns of errors in ascending order, so the error type (Writing noon instead of tanween fath) is the first column, and the error type (Hamzat Qate'e) is the last

column, and then we wrote python code to fill the class column value according to the type of the first error in each row among the 24 types of errors mentioned previously. Then, I stratified split the data to train and test data according to class column. The resulting training and testing sets have numbers of errors per class as shown in Tables 6 and 7, respectively.

Table 6. REALMS train set (2488 rows)

Type of Error	Number of errors
Hamzat Qataa	4821
Writing Ha'a instead of Taa marbotah in the end of the word	967
Substitution	980
Missing space	982
Slang words	894
Missing letter	593
Adding letter	555
Adding space after Conjunctive (WAW AL-Atf)	347
Adding or missing WAW ALJAMAA	187
Hamzat wasel	176
Hamzet Madd	157
Hamzah mutawassetah	107
Hamzah mutatarrrefah	57
Writing Taa marbotah instead of Ha'a in the end of the word	32
Writing Alef mamdodah instead of alef maqsora	21
Inversion (writing two adjacent letters in the word instead of each other)	22
Writing teh instead of teh marboutah	21
Deletion of AL-Altareef	14
Writing teh marboutah instead of teh	9
Deletion of AL-Lam AL shamseiah	7
Writing Alef maqsora instead of alef mamdodah	6
Adding *alef* after *tanweeen* the words that end with *hamzah*	6
Adding alef & noon after *tanween fath*	2
Writing noon instead of *tanween fath*	2
Total number of errors	10965

Table 7. REALMS test set (622 rows)

Type of Error	Number of

	errors
Hamzat Qataa	1057
Writing Ha'a instead of Taa marbotah in the end of the word	229
Substitution	216
Missing space	193
Slang words	206
Missing letter	117
Adding letter	137
Adding space after Conjunctive (WAW AL-Atf)	102
Adding or missing WAW ALJAMAA	45
Hamzat wasel	55
Hamzet Madd	40
Hamzah mutawassetah	32
Hamzah mutatarrrefah	12
Writing Taa marbotah instead of Ha'a in the end of the word	6
Writing Alef mamdodah instead of alef maqsora	6
Inversion (writing two adjacent letters in the word instead of each other)	4
Writing teh instead of teh marboutah	5
Deletion of AL-Altareef	3
Writing teh marboutah instead of teh	3
Deletion of AL-Lam AL shamseiah	3
Writing Alef maqsora instead of alef mamdodah	3
Adding alef after tanweeen the words that end with hamzah	1
Adding alef & noon after tanween fath	0
Writing noon instead of tanween fath	0
Total number of errors	2475

The next step is to reduce the lines lengths by wrapping the long lines into maximum 300 characters as was done on the Wiki-40b dataset and as mentioned in the previous subsection. However, I faced a problem in the lengths of the lines after wrapping the long lines. The problem is that my data has slang words, where those words were corrected in the target column into words of different length. Also, the length of the input erroneous text is different from the length of the target correct text, which is dependent on the type of errors, where sometimes we may add one or more

characters to fix the errors. As a result of this problem, the input and target in the dataset is not aligned in the dataset after wrapping the long lines, where many rows do not have the same words for the input and the target because they are shifted to next rows. To fix this problem, we must manually iterate over all rows and assure that they are aligned by removing a word from a line and adding it to the next line, or vice versa. This is a tidy process.

As a result, and after wrap the long line and made the alignment manually between the input and the target, REALMS train set input is composed of 4123 sequences which contains 61819 words, and the test set input is composed of 871 sequences which contains 13621 words.

REALMS dataset was posted on Github for public use on the link https://github.com/RaghdaDiab/REALMS. The dataset contains spelling mistakes that are corrected as well as slang words which are also corrected in both the input and the target columns.

4.3. Experimental Results and Analysis

This section is divided into two subsections, the first subsection describes the training and testing done on the Wiki-40B dataset, and the second subsection describes the training and testing done on our own collected dataset. During training and testing, we tried several values of the model hyperparameters. The best set of hyperparameters values are as follows:

- Number of layers (number of encoders and decoders) = 4
- Model dimension = 128
- Number of heads self-attention layers = 8
- Units = 512
- Dropout rate = 0.1

4.3.1. Training and testing on Wiki-40B dataset

First, we have built three variations of the Wiki-40B dataset, these are: dataset injected with intelligent errors only, dataset injected with intelligent plus 2% random errors, and dataset injected with intelligent plus 5% random errors.

To achieve high testing accuracy, we executed extensive experiments. In these experiments, we were changing the types of errors, probability of injection, and data split ratios. Among these experiments, we trained the transformer model on the training set that is injected with intelligent errors of classes (A, B, C, and D) with probability of injection 30% and random error 5%. The training accuracy after 53 and 106 epochs are shown in Table 8. The time consumed for each epoch is on average 19 minutes, which is almost 35 hours for the 106 epochs.

Table 8. Results of training wiki-40B that has 30% intelligent error and 5% random errors

Wiki-40B	Training Accuracy	Training Loss	Validation accuracy	Validation losses
After epoch 53	0.9899	0.0334	0.9907	0.0307
After epoch 106	0.9906	0.0311	0.9913	0.0290

Figures 15 and 16 illustrate the loss and accuracy, respectively for training and validation. The improvement in loss and accuracy is very slow after the first few epochs.

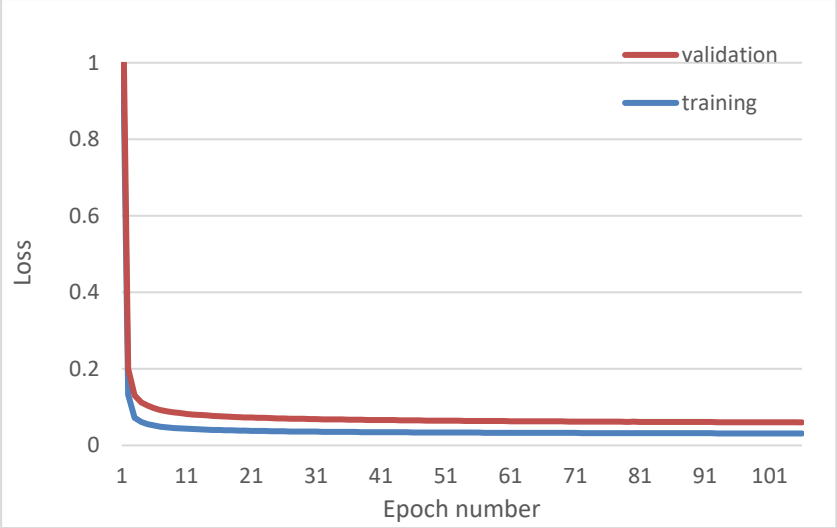

Figure 15: Wiki-40B Loss on training and validation sets

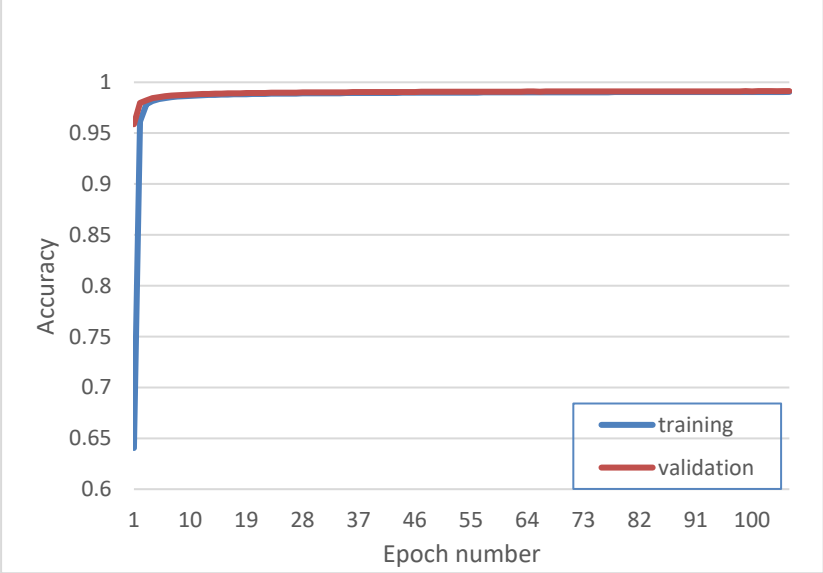

Figure 16: Wiki-40B Accuracy on training and validation

Based on this training, we performed testing for the transformer on Wiki-40B test set. We randomly selected 2000 sequences from the test set to reduce the testing time, since it takes approximately 600 hours to test the entire test set, and Kaggle offers only 30 hours per week to use its GPU accelerators. The results of testing wiki according to the result of 53 epochs and 106 epochs training are shown in Table 9. The results are impressive compared to the number of errors that were injected. It is

also clear that as the ratio of errors injected increases, the accuracy decreases, yet very acceptable. The reason is that as more errors are injected, some words are very hard to correct, especially when several characters in the same word are changed.

Table 9. Results of testing wiki-40B injected with different random errors rate

Error rate Type	Accuracy		Character error rate	
	After 53 epochs training	After 106 epochs training	After 53 epochs training	After 106 epochs training
Test set contains both intelligent errors (A-D) and 0.05 random errors.	98.959%	99.027%	1.04%	0.973%
Test set contains both intelligent errors (A-D) and 0.02 random errors.	99.227%	99.263%	0.773%	0.737%
Test set contains only intelligent errors (A-D)	99.403%	99.454%	0.597%	0.546%

Several selected samples of spelling mistakes correction are shown in Table 10 (for intelligent errors injection), Table 11 (for 2% random errors), and Table 12 (for 5% random errors).

Table 10 . Sample of input, target and prediction of wiki-40B injected with only intelligent errors (A-D)

Input (intelligent)	Target correct text	Prediction result
ويعتبر **هظا** المشروع **اول** مشروع **للاصتثماراة** الأجنبية ينفذ في **روسيى** بصورة مستقله **ويغمل** بموجب **ألتكنولوحيا** المتكاملة وتنتج شركة **جنرنل** موتورز سوية مع شركة **نقتوفاز** **السبارة** شيفي **نبفى الجاصة** بالطرق **اءلوعرة** **بمصتع وٰلسيارات** في مدينة **نوليءني** تبلغ قدرة **اءنتاج نلمسنع** ألف **سيئرة** سنويا وكلفه المشروع مليون دوزلار وبدأ في **روسبى** **تخميع نلنموذج ألمدني للصيارت هأمر وٰلخاصة** بالطرق **الوغرة**	ويعتبر **هذا** المشروع **أول** مشروع **للاستثمارات** الأجنبية ينفذ في **روسيا** بصورة مستقلة **ويعمل** بموجب **التكنولوجيا** المتكاملة وتنتج شركة **جنرال** موتورز سوية مع شركة **افتوفاز** **السيارة** شيفي **نيفا الخاصة** بالطرق **الوعرة** **بمصنع السيارات** في مدينة **تولياتي** تبلغ قدرة **إنتاج المصنع** ألف **سيارة** سنويا وكلفة المشروع مليون دوزلار وبدأ في **روسيا تجميع النموذج المدني للسيارة هامر** **الخاصة** بالطرق **الوعرة**	ويعتبر **هذا** المشروع **أول** مشروع **للاستثمارات** الأجنبية ينفذ في **روسيا** بصورة مستقلة ويعمل بموجب التكنولوجيا المتكاملة وتنتج شركة جنرال موتورز سوية مع شركة **التلفاز** السيارة شيفي **تبقى الخاصة** بالطرق **الوعرة** **بمصنع السيارات** في مدينة **تولياتي** تبلغ قدرة **إنتاج المصنع** ألف **سيارة** سنويا وكلفة المشروع مليون دوزلار وبدأ في **روسيا تجميع النموذج المدني للسيارة** **وأمر الخاصة** بالطرق **الوعرة**
محب الدين الحموي **سيرنه** هو محمد بن **آبي** بكر بن داود بن عبد **ألرحمن العلوٰني** الحموي **ءبو** الفضل المعروف **يمخب** الدين بن تقي **وٰلدين** ولد سنة ه م بحماة من **مشابخه أيو** الوفاء **نبن** ولي الله الشيخ **غلوان وأيو البفاء** وغيرهما من تلامذته **نلناج** القطان **والشبخ غبد** الرحمن **نلعمادي** وغيرهم ولي **القضىء** بمصر وحمص **وحسن نلأكروٰد ومعرت** النعمان ومعرة نسرين وكلس **وآعزاز** ثم **آستقر** بدمشق فولي بها القضاء **نيأبة**	محب الدين الحموي **سيرته** هو محمد بن **أبي** بكر بن داود بن عبد **الرحمن العلواني** الحموي **أبو** الفضل المعروف **بمحب** الدين بن تقي **الدين** ولد سنة ه م بحماة من **مشايخه أبو** الوفاء **ابن** ولي الله الشيخ **علوان وأبو البقاء** وغيرهما من تلامذته **التاج** القطان **والشيخ عبد** الرحمن **العمادي** وغيرهم ولي **القضاء** بمصر **وحمص وحصن** **الأكراد ومعرة** النعمان ومعرة نسرين وكلس **واعزاز** ثم **استقر** بدمشق فولي بها القضاء **نيابة**	محب الدين الحموي **سيرته** هو محمد بن **أبي** بكر بن داود بن عبد **الرحمن العلواني** الحموي **أبو** الفضل المعروف **بمحب** الدين بن تقي **الدين** ولد سنة ه م بحماة من **مشايخه أبو** الوفاء **ابن** ولي الله الشيخ **علوان وأبو البقاء** وغيرهما من تلامذته **التاج** القطان **والشيخ عبد** الرحمن **العمادي** وغيرهم ولي **القضاء** بمصر **وحمص وحسن** **الأكراد ومعرة** النعمان ومعرة نسرين وكلس **وأعزاز** ثم **استقر** بدمشق فولي بها القضاء **نيابة**

Table11 . Sample of input, target and prediction of wiki-40b injected with 30% intelligent errors and 2% random errors.

Input with 2% random error	Target correct text	Prediction result
ويعتبر هذا المشروع أول مشروع للاستثمارات **علؤجنبية** ينفذ في **رلسيا** بصورة مستقلة **ويغمل** بموجب **علتكنولوجيا علمتكاملة** وتننج شركة **خنرأل** موتورز سوية مع **شركت افنوفاز السبارة** شيفي نيفا **ولجاصة** بالطرق **الظعرت** بمصنع **السيآرآت** في **مدبنة** توليأتي **تبمغ** قدرة **إنتآج اعلمصنع آلف** سيارة **صنويا** وكلفة المشروع مليون دوزلار **وبضأ** في روسيا **تجمبع** النموذج المدني للسيارة هامر **علخاصة** بالطرق الوعرة	ويعتبر هذا المشروع أول مشروع للاستثمارات **الأجنبية** ينفذ في **روسيا** بصورة مستقلة **ويعمل** بموجب **التكنولوجيا المتكاملة** وتنتج شركة **جنرال** موتورز سوية مع **شركة افتوفاز السيارة** شيفي نيفا **الخاصة** بالطرق **الوعرة** بمصنع **السيارات** في **مدينة** توليأتي **تبلغ** قدرة **إنتاج المصنع ألف** سيارة **سنويا** وكلفة المشروع مليون دوزلار **وبدأ** في روسيا **تجميع** النموذج المدني للسيارة هامر **الخاصة** بالطرق الوعرة	ويعتبر هذا المشروع أول مشروع للاستثمارات **الأجنبية** ينفذ في **روسيا** بصورة مستقلة **ويعمل** وتنتج شركة **جنرال** بموجب **التكنولوجيا المتكاملة** موتورز سوية مع **شركة النوفاز السيارة** شيفي نيفا **الخاصة** بالطرق **الشعرة** بمصنع **السيارات** في **مدينة** توليأتي **تبلغ** قدرة **إنتاج المصنع ألف** سيارة **سنويا** وكلفة المشروع مليون دوزلار **وبدأ** في روسيا **تجميع** النموذج المدني للسيارة هامر **الخاصة** بالطرق الوعرة
مجب الدين الحموي **صبرته** هو محمد بن **آبي** بكر بن داود **بن عيد** الرحمن **علعلواني الخموي أبو** الفضل المعروف بمحب الدين بن تقي الدين ولد **سنه** ه م بحماة من مشايخه أبو **اعلوفؤء** ابن **تلي اللة اعلشيخ** علوان **وأبوا** البقاء **وغبرهما** من **تلامذته التنج** القطان **وعلشيخ** عبد **الرجمن العموضي** وغيرهم ولي **القضأء** بمصر وحمص وحصن **اعلاكراد** ومعرة النعمان **ومعضة** نسرين وكلس واعزاز ثم استقر بدمشق فولي بها القضاء نيابة	**محب** الدين الحموي **سيرته** هو محمد بن **أبي** بكر بن داود **بن عبد** الرحمن **العلواني الحموي أبو** الفضل المعروف بمحب الدين بن تقي الدين ولد **سنة** ه م بحماة من مشايخه أبو **الوفاء** ابن **ولي الله الشيخ** علوان **وأبو** البقاء **وغيرهما** من **تلامذته التاج** القطان **والشيخ** عبد **الرحمن العمادي** وغيرهم ولي **القضاء** بمصر وحمص وحصن **الأكراد** ومعرة النعمان **ومعرة** نسرين وكلس واعزاز ثم استقر بدمشق فولي بها القضاء نيابة	**محب** الدين الحموي **سيرته** هو محمد بن **أبي** بكر بن داود **بن عبد** الرحمن **العلواني الحموي أبو** الفضل المعروف بمحب الدين بن تقي الدين ولد **سنة** ه م بحماة من مشايخه أبو **تلي الله الشيخ** علوان **وأبو** البقاء **وغيرهما** من **تلامذته التاج** القطان **والشيخ** عبد **الرحمن العمادي** وغيرهم ولي **القضاء** بمصر وحمص **وحسن الاكراد** ومعرة النعمان **ومعدة** نسرين وكلس واعزاز ثم استقر بدمشق فولي بها القضاء نيابة

Table12 . Sample of input, target and prediction of wiki-40B injected with 30% intelligent and 5% random errors.

Input with 5% random error	Target correct text	Prediction result
وبعنبر هذا المشروع أول **ششروع للاستثمءرات الاءحنيية** ينفذ في روسيا بصورة **مستفلة** ويعمل بموجب **إلتكنولوحيا اشمتكاملة وننتج** شركة جنرال موتورز **سويت** مع **شركت افتوفاز اعلصيارة** شيفي نيفا **اوخاصة بؤلطرق ألوعرة** بمصنع السيارات في مدينة توليأتي **تيلغ فدرة** إنتاج المصنع **اعلف سبارة ستويا وتلفة اكمشروع مليهن** دوزلار وبدأ **قي ظوسيا** تجميع **ادنموذج** المدني للسيارة **هؤمر** الخاصة بالطرق **اعلوعرة**	**ويعتبر** هذا المشروع **أول مشروع للاستثمارات الأجنبية** ينفذ في روسيا بصورة **مستقلة** ويعمل بموجب **التكنولوجيا المتكاملة وتنتج** شركة جنرال موتورز **سوية** مع **شركة افتوفاز السيارة** شيفي نيفا **الخاصة بالطرق الوعرة** بمصنع السيارات في مدينة توليأتي **تبلغ قدرة** إنتاج المصنع **ألف سيارة سنويا وكلفة المشروع مليون** دوزلار وبدأ **في روسيا** تجميع **النموذج** المدني للسيارة **هامر** الخاصة بالطرق **الوعرة**	**ويعتبر** هذا المشروع **أول مشروع للاستثمارات الأجنبية** ينفذ في روسيا بصورة **مستقلة** ويعمل **وتنتج** شركة بموجب **التكنولوجيا المتكاملة** جنرال موتورز **سويت** مع **شركة التلفاز السيارة** شيفي نيفا **الخاصة بالطرق الوعرة** بمصنع السيارات في مدينة توليأتي **تبلغ قدرة** إنتاج المصنع **ألف سيارة سنويا وتلفت المشروع** وبدأ **في روسيا** تجميع **النموذج مليون** دوزلار المدني للسيارة **هامر** الخاصة بالطرق **الوعرة**
محب **علدين الجموي سبرته** هو **محزد** بن أبي بكر بن **دئود** بن عبد الرحمن **اعلعلواني إلجموي اعبو القضل إلمعروف** بمحب الدين بن تقي **اندين** ولد **ستة** ه م **بحمنة** من مشايخه **ابو الوفأء اين** ولي **آللة** الشيخ علوان **وؤبو اعلبقئء وعبرهما** من **تصامذته** التاج **الفطان** والشيخ عبد **وْلرحمن ألعمإضي** وغيرهم ولي **إلقضاء يمصر حممص** وحصن الأكراد ومعرة النعمان ومعرة **نصرين ظلكس** واعزاز ثم **أستقر بضمشق** فولي بها **وْلقضاءء نياءبة**	محب **الدين الحموي سيرته** هو **محمد** بن أبي بكر بن **داود** بن عبد الرحمن **العلواني الحموي أبو الفضل المعروف** بمحب الدين بن تقي **الدين** ولد **سنة** ه م **بحماة** من مشايخه **أبو الوفاء ابن** ولي **الله** الشيخ علوان **وأبو البقاء وغيرهما** من **تلامذته** التاج **القطان** والشيخ عبد **الرحمن العمادي** وغيرهم ولي **القضاء بمصر وحمص** وحصن الأكراد ومعرة النعمان ومعرة **نسرين وكلس** واعزاز ثم **استقر بدمشق** فولي بها **القضاء نيابة**	محب **الدين الحموي سيرته** هو **محمد** بن أبي بكر بن **داود** بن عبد الرحمن **العلواني الحموي أبو الفضل المعروف** بمحب الدين بن تقي **الدين** ولد **سنة** ه م **بحماة** من **مشايخه أبو الوفاء ابن** ولي **الله** الشيخ علوان **وأبو البقاء وغيرهما** من **تلامذته** التاج **القطان** والشيخ عبد **الرحمن العمادي** وغيرهم ولي **القضاء بمصر وحمص** وحصن الأكراد ومعرة النعمان ومعرة **نصرين وكلس** واعزاز ثم **استقر بدمشق** فولي بها **القضاء نيابة**

4.3.2. Training and testing on our collected dataset

After we have trained the model on Wiki-40B dataset, we trained the model on part of our collected dataset. This step is needed because our collected dataset has

many types of errors including errors that are not injected in Wiki-40B dataset. Also, our dataset has slang words. Therefore, it is better to perform training on our dataset before testing the model on our dataset. For example, when we tested on our collected dataset directly after training on Wiki-40B, the accuracy was 96% and CER 3.35%. Therefore, another phase of training on our collected dataset must be done first. Therefore, we trained the model on our dataset after training on Wiki-40B for 53 and 106 epochs. The results of training on our dataset are shown in Table 13. The testing accuracy after this training became 97.76% and CER 2.24%.

Table 13. Results of training on REALMS train data

REALMS train set	**Training accuracy**	**Training Loss**	**Validation accuracy**	**Validation loss**
After 53 epochs train on wiki	0.9721	0.1135	0.9713	0.1316
After 106 epochs train on wiki	0.9797	0.0792	0.9742	0.1265

The results above show that the performance is not good enough compared to the performance results which were achieved on Wiki-40B dataset. Therefore, many experiments and tuning steps in this stage were done trying to increase the accuracy. After deep inspection and heavy analysis, we discovered two major reasons behind the lower accuracy:

1) The slang words: among the errors in our dataset, there are slang words, which we corrected them to their formal format manually in the target output column of the dataset. We were expecting that the transformer may learn to correct the input slang words by converting them into exactly their corrected formal output format. However, almost none of those slang words were corrected in the testing phase. Also, even if it will be corrected into a formal word, there is no guarantee that it

will be converted to the same word we inserted in the output column of the dataset.

It is logical that the transformer will not fix the slang words, because the training on the wiki dataset has no slang words and did not learn to fix slang words. Additionally, even though we trained the transformer model on our dataset which contains slang words, these words are not enough. Also, the words in the training part of the dataset are not necessarily like the words in the testing part of the dataset.

Handling this point is done by correcting the slang words in both the input text and the target text to be identical. This step was done manually by iterating over all the slang words in our dataset and correct the slang words in the input text column. Therefore, the slang words are not within the scope of our work, where we will focus on correcting spelling mistakes only. The results of training after this step are shown in Table 14. It shows a considerable enhancement in the validation accuracy. Also, the achieved testing accuracy was 98.813% and CER 1.187%

Table 14. Results of training REALMS train data without slang words on two stages

REALMS train set	**Training accuracy**	**Training Loss**	**Validation accuracy**	**Validation loss**
After 53 epochs train on wiki	0.9994	0.0023	0.9910	0.0705
After 106 epochs train on wiki	0.9995	0.0018	0.9911	0.0741

2) More error types: our collected dataset has many errors that were not injected in Wiki-40B dataset. Therefore, we relied on the transformer to learn from our collected dataset, but the size of our dataset is not enough to do enough training. Handling this point requires either collecting much larger dataset or injecting more errors. Collecting larger dataset is very time consuming as was shown in Chapter 3. Therefore, we decided to go to injecting more errors. Consequently, we injected Wiki-40B dataset with more errors. Specifically,

we injected it with the errors of classes (E, F, and G) in addition to classes (A, B, C, and D).

With these comprehensive classes of errors, we tried to mimic all errors that any person may produce and can be automatically injected in the data without corrupting other texts. On the other hand, since we extensively increased the classes of errors, we reduced the error injection probability to 20% rather than 30% for all errors classes to decrease the probability of generating multiple errors in the same word.

So, we trained the model on Wiki-40B that injected with only intelligent errors from classes (A-G) with probability 20%, the results of training is in Table 15.

Table 15. Results of training Wiki-40B with 20% intelligent errors classes from (A) to (G)

Wiki-40B	Training accuracy	Training Loss	Validation accuracy	Validation loss
Training for 57 epoch	0.9932	0.0203	0.9938	0.0185

The testing results on 2000 rows of Wiki-40B dataset after the above training are: testing accuracy = 99.12% and CER = 0.87%. Then, we trained the model on REALMS dataset, and the results are in Table 16.

Table 16. Results of training on REALMS train dataset after training Wiki-40B with 20% intelligent errors classes from (A) to (G)

REALMS train set	Training accuracy	Training loss	Validation accuracy	Validation loss
Training after Wiki-40B train for 57 epoch	0.9913	0.0347	0.9883	0.0585

The testing results on REALMS test set according to the conducting training above, gives Accuracy 98.85% and CER: 1.14%.

As expected, the accuracy has been enhanced after excluding the slang words and after training the model on more types of errors. The accuracy became very close to the testing that was done on the Wiki-40B dataset which is injected with 30% intelligent errors and 5% random errors. However, the accuracy is a little lower because we still have several types of errors that were not injected in the Wiki-40B dataset. During the injection step, we tried to statistically inject the most common errors. Some examples on corrections that were done on our collected dataset are shown in Table 17.

Table 17. Sample from REALMS and wiki-40B datasets with prediction

Input	Target	Prediction
بعيد **ع** السياسة أجمل مشهد **أنسحاب** الشاب السعودي والأجمل منه تصرف الشاب الإيراني **شاهد و**	بعيد **عن** السياسة أجمل مشهد **انسحاب** الشاب السعودي والأجمل منه تصرف الشاب الإيراني **شاهدوا**	بعيد **عن** السياسة أجمل مشهد **انسحاب** الشاب السعودي والأجمل منه تصرف الشاب الإيراني **شاهدوا**
فوز للسينغال **ان شا** الله يا ربي	فوز للسينغال **إن شاء** الله يا ربي	فوز للسينغال **إن شاء** الله يا ربي
كل الاحترام والتقدير **ع** جهودكم	كل الاحترام والتقدير **على** جهودكم	كل الاحترام والتقدير **على** جهودكم
أمريكا **الأن** ستستعمل الإرهاب كمحاولة منها **لإظعاف** بوتن	أمريكا **الآن** ستستعمل الإرهاب كمحاولة منها **لإضعاف** بوتن	أمريكا **الآن** ستستعمل الإرهاب كمحاولة منها **لإضعاف** بوتن
هذه الحرب سيكون مداها أطول من المتوقع **والمختط** له	هذه الحرب سيكون مداها أطول من المتوقع **والمخطط** له	هذه الحرب سيكون مداها أطول من المتوقع **والمخطط** له
المولدوالنشأت	**المولد والنشأة**	**المولد والنشأة**
طبيعة رون **المتردضة**	طبيعة رون **المترددة**	طبيعة رون **المترددة**
تنذيف بالحمض	**تنظيف** بالحمض	**تنظيف** بالحمض
لأنو ذهب **الي** الملعب	لأنه ذهب **إلى** الملعب	لأنه ذهب **إلى** الملعب
لا يمتلك رون أي **قذزات** خاصة ولا أي براعة في أي شيء	لا يمتلك رون أي **قدرات** خاصة ولا أي براعة في أي شيء	لا يمتلك رون أي **قدرات** خاصة ولا أي براعة في أي شيء
مع ذلك **يتضبر** أمره في دراسته	مع ذلك **يتدبر** أمره في دراسته	مع ذلك **يتدبر** أمره في دراسته
ولاحقاطوره بيل توت ل **متعددحدود توة**	ولاحقا طوره بيل توت ل **متعدد حدود توت**	**ولاحقا طوره** بيل توت ل **متعدد حدود توت**

Figures 17 and 18 illustrate the effect of increasing the error injection rate on CER and Accuracy, respectively. We injected Wiki-40B test set with the intelligent errors classes A-F (not class G), and with different errors rate, then we tested on a 500 subset of each test set. Our goal is to find the relation between the injection error rate and the evaluation metric.

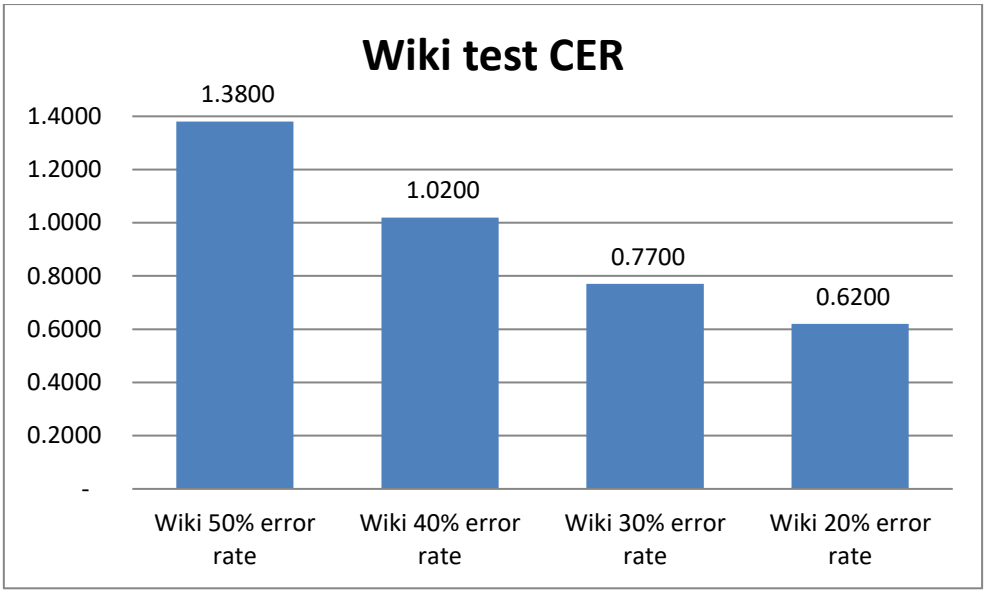

Figure 17: Wiki-40B CER with different error injection rates for intelligent errors only(A-F)

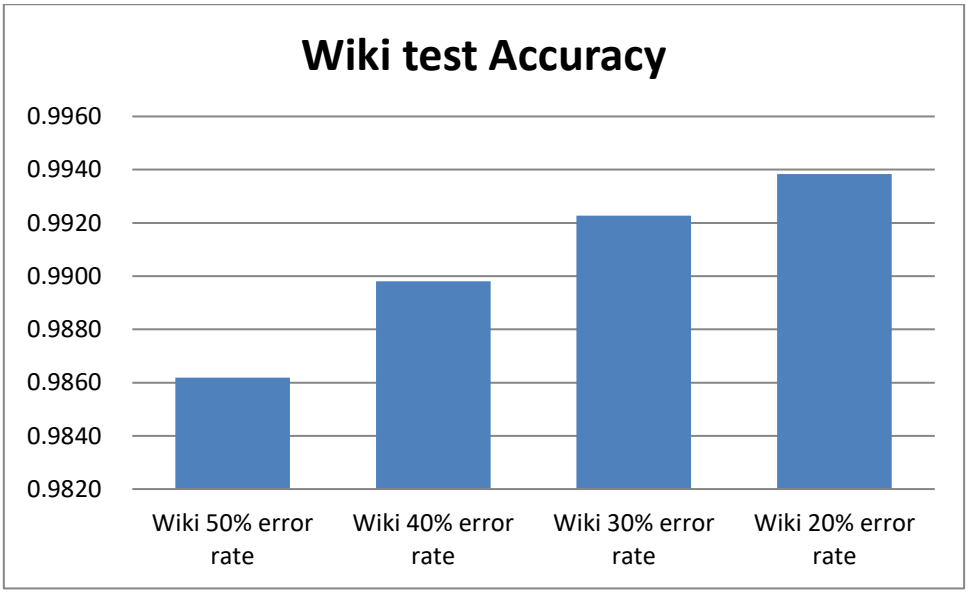

Figure 18: Wiki-40B accuracy with different error injection rates for intelligent errors only(A-F)

The decrease in accuracy is justifiable. For example, with 50% error rates, it means that most of the letters are replaced by other letters with probability 50%. The achieved accuracy in this case is 98.62%. This means that 1.38% of the injected errors (around 50%) were not corrected while the others were corrected. On the other hand, for 20% errors injection, the accuracy is 99.39% which means that 0.61% out of the 20% errors were not corrected.

In summary, our developed model achieved high accuracy on both Wiki-40B dataset which is statistically injected with errors and on our collected dataset. Additionally, we concluded that spelling errors correction using machine learning models will not be 100%, and human should always do another round of error correction for the following reasons:

a) Many names are very hard to be corrected using machine learning. Names include names of people, pets, countries, cities, and many others. Examples like in Table 18.

Table 18. Sample of wrong names prediction

Input	Target	Prediction
وتخرج من **هوجووزتس**	وتخرج من **هوجوورتس**	وتخرج من **هو جووزتس**
حلف وارسو	حلف وارسو	**حلف وأرسوا**
اربد	إربد	أريد

b) Many words were corrected to other words which are correct but not the required words. An example is in Table 19.

Table 19. Sample of words predicted incorrectly to other correct words

Input	Target	Prediction
وإعادة **احيائها**	وإعادة **إحيائها**	وإعادة **أحيائها**
بأغلى الدرجات	**بأعلى** الدرجات	**بأغلى** الدرجات
ويسبح حكيما	**ويصبح** حكيما	**ويسبح** حكيما
اي اته يرهنان كل مخطط مستو يمكن تلوينه بواسطة خمس **الوان** فقط	**أي أنه برهن أن** كل مخطط مستو يمكن تلوينه بواسطة خمس **ألوان** فقط	**أي أنه يرهن أن** كل مخطط مستو يمكن تلوينه بواسطة خمس **الوان** فقط
اعتقدالجميع ان المساءلة قد **ونتهت**	**اعتقد الجميع** أن **المسألة** قد **انتهت**	**اعتقد الجميع** أن **المساءلة** قد **انتهت**

c) Statistical error injection is not possible to cover all types of errors. Also, collecting samples of all possible errors and handling them is impossible. Therefore, there will be always a small factor (less than 1%) which need human correction.

To compare the proposed model to state of the art research, we compare our results on REALMS dataset with the ones on Test200 set that were obtained using the transformer models in (Al-Qargholi *et al*., 2021). Our model achieves a CER 1.14%

on REALMS data that has many more types of errors. Specifically, REALMS contains the errors of classes A to G and other errors that were not injected in wiki-40B dataset. While (Al-Qargholi *et al*., 2021) achieved 0.87% CER on Test200 which contained only errors in Hamza in its various forms, and errors in the *ta'a* at the last of the word. Table 20 shows a comparison of the results.

Table 20. Summary of comparison to (Al-Qargholi et al., 2021)

	Model	Dataset	Accuracy	Dataset	CER
Al-Qargholi et al., 2021	Transformer	Wiki-40B	97%	Test200	0.86%
Proposed work	Transformer	Wiki-40B	99.12%	REALMS	1.14%

Another state-of-the-art work which was published recently is the work done by (Aichaoui *et al*.,2022). They collected a dataset from different sources including (Riyadh newspaper, Okaz, BBC, Learning al-jazeera education website, and al-Maktaba al-shamela library). Their goal is to collect words that are orthographically correct which is composed of 423,778 words.

Like what we did, they performed statistical error injection but on word basis. Therefore, their dataset is only composed of one word per row, and each word was injected with errors of the following types:

1- Delete a letter: where duplicated the word several times, and each time, they delete one letter.
2- Permutate letters: they changed the positions of the letters inside the word.
3- Phonetic ({ح ,ه}, {ظ،ذ}, {ض،د}, {ص،س}, {ط،ت}): by replacing each letter among the pair with the other letter from the same pair.

4- Physical in hand writing between the letters that are same in shape ({ف،ق}, {ر,ز}, {ى،ي}, {خ،ح،ج}, {ظ،ط}, {ض،ص}, {ذ،د}, {غ،ع}, {ا،أ،إ،آ}, {س،ش})
5- Keyboard swapping: where they replaced each character in the correct word with letter immediately adjacent to it on the keyboard (beside, below, or above it on the keyboard).
6- Add space: where they inserted a space after each position in the correct word.
7- Morphological errors: for the letters that are pronounced but not written like "لكن" , "هذا", the third form is confusion between {ة،ه},{ا،ى}.
8- Diacritization (*Tashkil*) errors that results from mistakes in *tanween* and harakat *fatah*, *dammma*, and *kasrah.*

The above errors were injected in the dataset they collected which is called (SPIRAL). The methodology they followed is a little bit naive and not suitable. More specifically, they built a version of the dataset for each type of the errors above, where each row of the dataset contains only one correct word and the erroneous word of one type. Then, they performed training and testing on this version of the dataset for one type of errors. Based on the testing they did; they reported the testing result. Therefore, the training and testing were done on a version of the dataset which contained only words rather than texts and one type of errors. Then, they repeated the same steps on each version of the dataset which contains another type of errors. Therefore, the testing results can not be generalized to texts which contain several types of errors. The model they used is ARABERT which is a word level transformer, and trained the data of each category separately, and their achieved results are as shown in Table 21.

Table 21. Summary of Aichaoui et al. (2022) results

Categories	Accuracy	Precision	Recall	F1 score	Epochs
Morphology	0.889	0.802	0.802	0.802	2
Phonetic	0.898	0.816	0.817	0.816	15
Physical	0.843	0.730	0.730	0.730	3
Permutation	0.878	0.784	0.783	0.783	15
Keyboard	0.783	0.645	0.644	0.643	7
Delete	0.503	0.338	0.337	0.337	11
Space-issues	0.922	0.869	0.863	0.860	11
Tachkil	0.922	0.856	0.856	0.845	4

Even on a word level basis, we achieved accuracy higher than any of their types.

Chapter 5

Conclusions and Future Work

In this thesis we built a dataset of Arabic texts which contain real Arabic spelling mistakes called REALMS dataset. In this dataset, we did big effort of cleaning the data, classifying the spelling mistakes, correcting spelling mistakes, clean error free rows, and correctly slang words. The size of the dataset initially was 10337 rows, and after all steps, the resulting dataset contained 3110 rows, all of them contain Arabic spelling mistakes. We published REALMS dataset on GitHub on "https://github.com/RaghdaDiab/REALMS".

To prepare the machine learning model, we decided to choose state of the art machine language model which is based on Neural Network called Transformer. Recently, transformer showed excellent capabilities in handling many natural language processing jobs including correcting spelling mistakes. Additionally, transformers started replacing models like LSTM.

Transformers and Neural Network machine learning models require large datasets. Therefore, REALMS is not of enough size. For that reason, we downloaded a huge dataset called Wiki-40B dataset which contains millions of error free Arabic texts. Then, we statistically injected the Wiki dataset with the most frequent errors and did training and testing on Wiki dataset. After that, we used the training weights from Wiki training to do training and testing on REALMS dataset. The results were excellent where the proposed model achieved a high accuracy of 99.12% on Wiki-40B dataset and character errors rate of 1.14% on REALMS dataset.

In this thesis, we handled many comprehensive types of errors. These errors are based on the study performed on REALMS dataset. These errors included confusion among *hamza* shapes, confusion among letters that have almost the same sound like *seen* and *sad, dad* and *tha'a*, confusion between letters that are adjacent to each other on the keyboard and have almost the same shape and the difference between them is only dot like *jeem, ha'a,* and *kha'a (خ ح ج),* insertion spaces between the prepositions (*fee* and *ala)* and the words after them that start with "*AL- Atareef*"'.

For future work, we suggest injecting more types of errors that were not included in this study like spaces. For example, by adding spaces inside the word itself, deleting spaces between any two words, and removing the space between the words which will affect the shape of the two words. Another form of error that can be injected is deleting one letter randomly from the word, inversion or permutation of letters in which we switch any two letters in the same word.

Another dimension for future work is to use other transformer models which contain encoder-decoder architecture like BART (Lewis *et al*.,2019) and T5 (Raffel *et al*., 2019). Finally, we may work on collecting larger real dataset to enhance the performance of correcting spelling mistakes.

REFERENCES

Abandah, G., Suyyagh, A., & Khedher, M. Z. (2022). Correcting Arabic Soft Spelling Mistakes using BiLSTM-based Machine Learning. International Journal of Advanced Computer Science and Applications, 13(5).

Abdellah, Y., Lhoussain, A. S., Hicham, G., and Mohamed, N. (2020), Spelling Correction for the Arabic Language Space Deletion Errors. **Procedia Computer Science**, 177, 568-574.

Aichaoui, S. B., Hiri, N., Dahou, A. H., & Cheragui, M. A. (2022). Automatic Building of a Large Arabic Spelling Error Corpus. SN Computer Science, 4(2), 108.

Alammar, J. (2018), The Illustrated Transformer. **Github**. http://jalammar.github.io/illustrated-transformer/

Alkhatib, M., Monem, A. A., and Shaalan, K. (2020), Deep Learning for Arabic Error Detection and Correction. **ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)**, 19(5), 1-13.

Al-Qaraghuli, M., Abandah, G., & Suyyagh, A. (2021, November). Correcting Arabic Soft Spelling Mistakes Using Transformers. In 2021 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT) (pp. 146-151). IEEE.

AlShenaifi, N., AlNefie, R., Al-Yahya, M., and Al-Khalifa, H. (2015, July), ARIB@ QALB-2015 Shared Task: A Hybrid Cascade Model for Arabic Spelling Error Detection and Correction. **In Proceedings of the Second Workshop on Arabic Natural Language Processing,** pp. 127-132.

Attia, M., Pecina, P., Samih, Y., Shaalan, K., and Van Genabith, J. (2016), Arabic Spelling Error Detection and Correction. **Natural Language Engineering**, 22(5), 751-773.

Azmi, A. M., Almutery, M. N., and Aboalsamh, H. A. (2019), Real-Word Errors in Arabic Texts: A Better Algorithm for Detection and Correction. **IEEE/ACM Transactions on Audio, Speech, and Language Processing**, 27(8), 1308-1320.

Bijoy, M. H., Hossain, N., Islam, S., & Shatabda, S. (2022). DPCSpell:A Transformer-based Detector-Purificator-Corrector Framework for Spelling Error Correction of Bangla and Resource Scarce Indic Languages. arXiv preprint arXiv:2211.03730.

Guo, M., Dai, Z., Vrandečić, D., & Al-Rfou, R. (2020, May). Wiki-40b: Multilingual language model dataset. In Proceedings of the 12th Language Resources and Evaluation Conference (pp. 2440-2452).

Kuznetsov, A., and Urdiales, H. (2021), Spelling Correction with Denoising Transformer. **arXiv preprint** arXiv:2105.05977.

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., and Zettlemoyer, L. (2019), Bart: Denoising Sequence-To-Sequence Pre-Training for Natural Language Generation, Translation, and Comprehension. **arXiv preprint** arXiv:1910.13461.

Mubarak, H., and Darwish, K. (2014, October), Automatic Correction of Arabic Text: A Cascaded Approach. **In Proceedings of the EMNLP 2014 Workshop on Arabic Natural Language Processing (ANLP)**, pp. 132-136.

Noaman, H. M., Sarhan, S. S., and Rashwan, M. (2016), Automatic Arabic Spelling Errors Detection and Correction Based on Confusion Matrix-Noisy Channel Hybrid System. **Egypt Comput Sci J**, 40(2).

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., and Liu, P. J. (2019), Exploring the Limits of Transfer Learning with a Unified Text-to-text Transformer. **arXiv preprint** arXiv:1910.10683.

Samsudin, N., & Hassen, H. R. (2022, November). Transformer-Encoder Generated Context-Aware Embeddings for Spell Correction. In Artificial Neural Networks in Pattern Recognition: 10th IAPR TC3 Workshop, ANNPR 2022, Dubai, United Arab Emirates, November 24–26, 2022, Proceedings (pp. 129-139). Cham: Springer International Publishing.

Sun, R., Wu, X., & Wu, Y. (2023). An Error-Guided Correction Model for Chinese Spelling Error Correction. arXiv preprint arXiv:2301.06323.

Tran, H., Dinh, C. V., Phan, L., and Nguyen, S. T. (2021), Hierarchical Transformer Encoders for Vietnamese Spelling Correction. **arXiv preprint** arXiv:2105.13578.

UNESCO. (2019), World Arabic Language Day. **UNESCO**. https://en.unesco.org/commemorations/worldarabiclanguageday

van den Berg , T. (2020), Parametric Neural Networks. **CQF Institute**. https://www.cqfinstitute.org/sites/default/files/param-nn-1.3f.pdf

Zaghouani, W., Mohit, B., Habash, N., Obeid, O., Tomeh, N., Rozovskaya, A., and Oflazer, K. (2014), Large Scale Arabic Error Annotation: Guidelines and Framework. **Carnegie Mellon University**.

Zhang, S., Huang, H., Liu, J., and Li, H. (2020), Spelling Error Correction with Soft-masked BERT. **arXiv preprint** arXiv:2005.07421.

Zhao, Y., Yang, X., Wang, J., Gao, Y., Yan, C., and Zhou, Y. (2021), BART Based Semantic Correction for Mandarin Automatic Speech Recognition System. **arXiv preprint** arXiv:2104.05507.