الجامعة الأردنية

نموذج التفويض

أنا ملاك أحمد يوسف الصمادي، أفوض الجامعة الأردنية بتزويد نسخ من رسالتي /أطروحتي للمكتبات، أو المؤسسات، أو الهيئات، أو الأشخاص عند طلبهم حسب التعليمات النافذة ف الجامعة.

التوقيع:

التاريخ: 2024/4/17

The University of Jordan

Authorization Form

I, Mallak Ahmad Yousef Al-Smadi, authorize the University of Jordan to supply copies of my Thesis/ Dissertation to libraries or establishments or individuals on request, according to the University of Jordan regulations.

Signature:

Date: 17/4/2024

CORRECTING AUDITORY AND EXAGGERATION SPELLING MISTAKES IN JORDANIAN DIALECT USING MACHINE LEARNING TECHNIQUES

By
Mallak Ahmad Al-Smadi

Supervisor
Dr. Gheith Abandah, Prof.

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

School of Graduate Studies
The University of Jordan

April, 2024

COMMITTEE DECISION

This Thesis/Dissertation (Correcting Auditory and Exaggeration Spelling Mistakes in Jordanian Dialect Using Machine Learning Techniques) was Successfully Defended and Approved on

Examination Committee	**Signature**
Dr. Gheith Abandah, (Supervisor) Prof. of Computer Engineering	----------
Dr. Iyad Jafar, (Member) Prof. of Computer Engineering	----------
Dr. Mohammad Abdul-Majeed, (Member) Assoc. Prof. of Computer Engineering	----------
Dr. Hisham Almasaeid, (Member) Assoc. Prof. of Computer Engineering Yarmouk University	----------

DEDICATION

I dedicate this thesis to my beloved parents, Khalidah and Ahmad. Your unconditional love, encouragement, and infinite support have been the cornerstone of my journey.

Through all the ups and downs, all the triumphs, and challenges, your presence has been the source of my strength.

Also dedicating my thesis to my siblings, Abdullah, Malik, Ola, and Leen. Your belief in my aspirations kept me going and gave me the faith I needed to complete this thesis.

And to all my friends who kept encouraging me and supporting me through the tough times and all the long nights of staying up working on my thesis, thank you I'll never forget these times.

AKNOWLEDGEMENT

My greatest gratitude is to God; without his blessing and grace, I would never have been able to finish my thesis.

My biggest gratitude to my advisor, Professor Gheith Abandah for his infinite patience, support, and guidance. His expertise and feedback played a huge part in shaping my thesis in the best way.

I am extremely grateful for the time and effort Professor Gheith spent providing suggestions and reviewing that enhanced this work. His mentorship has enriched my academic experience and significantly contributed to my growth as a researcher.

I would like to extend my appreciation to my friends Bashar AlRfou and Rozan AlJaber, for their help in various matters in my code, which added light to the process of perfecting my thesis. Additionally, I would extend my deepest gratitude to my friends in AI MilkStraw for their endless availability to be there for my support.

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
BERT	Bidirectional Encoder Representations from Transformers
BiLSTM	Bidirectional Long-Short Term Memory
BLEU	Bilingual Evaluation Understudy
CRF	Conditional Random Field
CER	Character Error Rate
DL	Deep Learning
EIR	Error Injection Rate
GPT	Generative Pretraining Transformer
LSTM	Long Short-Term Memory
LM	Language Model
MSA	Modern Standard Arabic
NLP	Natural Language Processing
OCR	Optical Character Recognition
QALB	Qatar Arabic Language Bank
RNNs	Recurrent Neural Networks

CORRECTING AUDITORY AND EXAGGERATION SPELLING MISTAKES IN JORDANIAN DIALECT USING MACHINE LEARNING TECHNIQUES

By

Mallak Ahmad Al-Smadi

Supervisor

Dr. Gheith Abandah, Prof.

ABSTRACT

Dialects vary from one area to another, and some texts contain spelling errors, which are common in the Arabic language. Researchers have used multiple techniques to correct these spelling mistakes such as language models, confusion matrices, and neural networks. Recently, *transformers* have been introduced as an efficient neural network architecture, and since then they have been widely used to solve many natural language processing (NLP) tasks.

This thesis investigates using machine learning techniques based on efficient transformer model (Byt5) to correct two types of Arabic spelling mistakes: auditory and exaggeration mistakes. We train the model using training and validation datasets with synthetic errors inserted. These errors are created by a method called *stochastic error injection*, where they are applied to the large dataset used in this study (wiki40b/ar). For evaluation purposes, we collected from the social media networks two small datasets: TestAud200 that has 200 sequences with

example auditory mistakes and TestExg200 that has 200 sequences with example exaggeration mistakes.

As for the result, the model trained to correct auditory mistakes gives its best results when trained with error injection rate (EIR) of 75%, where the accuracy score is 79.78% on the validation set of wiki-40b/ar and the character error rate (CER) is 1.13% on TestAud200. The model trained to correct exaggeration errors, on the other hand, scores an accuracy of 99.78% on the validation set of wiki-40b/ar and a CER of 0.38% on TestExg200 when trained with an EIR of 10%.

Chapter One: Introduction

The Arabic language is the official language spoken in twenty-five countries worldwide, where approximately 313 million Arabic speakers are spoken (Callie, 2021). In 2022, the Arabic language was ranked as the 7th most spoken language in the world. The Arabic language has 28 letters and 12.3 million words. Some will be mispronounced as there are similar letters but with rough or soft pronunciation, and some words are used with exaggerated letters to show excitement, joy, sadness, *etc.*

1.1 Background

There are different spelling errors Arabs fall into while typing various words: discourse structure and pragmatic-level errors, morpho-syntactic errors, lexical and semantic errors, and soft errors (Al-Qaraghuli *et al.,* 2021). Our model solves a different type of error: local dialect replacement or duplication of letters.

A type of local dialects replacement is like "إنه ظابطٌ كبيرٌ في الجيش" the error is in the word "ظابط" the word must be "ضابط" but the confusion of the letter "ض" and "ظ" is present.

Machine Learning techniques are extremely helpful in many ways. Deep Learning (DL) has been an effective technique in solving natural language processing (NLP) problems, particularly when it comes to lingual mistakes.

1.2 Problem Statement

The spelling mistakes our research is going to resolve are the auditory mistakes such as "ظابط" for "ضابطً". The second spelling mistake people make while typing specific words is when they feel extreme excitement when admiring something. They type "يا سلاااام" even though the correct form is "يا سلام". Using a machine learning model, the wrong form should be replaced with the correct form.

Although the Arabic auditory and exaggeration mistakes were almost overlooked while correcting previous spelling mistakes, an evolving technique has been used over the years, using language models, long short-term memory (LSTM) recurrent neural network (RNN), mT5 Transformer, and even confusion metrics.

In communication platforms, exaggerated tones might be disturbing to some or hard to read by users, especially Arabic learners. Hence, a system that corrects this type of mistake, along with auditory mistakes, can facilitate communication. We, in this model, are solving these two issues.

1.3 Importance of Arabic Spelling Correction

The Arabic language commands considerable reverence and prestige within the spectrum of linguistic discourse, especially since it is the language of the Holy Quran. This research is of great importance because it employs machine learning techniques that have never approached exaggerating tones, correcting linguistic issues such as auditory and exaggerated tones, which both play a massive role in making people communicate more transparently and forwardly. This correction method will make it easier for Arabic learners to learn the language using systems like optical character recognition (OCR) and automotive speech recognition (ASR).

1.4 Thesis Objectives

The objectives of this thesis are to check if we can employ a machine learning technique suitable for correcting Arabic auditory and exaggeration mistakes and if it is possible to use it to correct these mistakes with good results automatically.

1.5 Thesis Questions

The questions of our thesis are going to answer:

- Can we obtain good results in correcting auditory and exaggerating mistakes using ML techniques?
- Will our model obtain better results than related work?

1.6 Thesis Contribution

In our thesis, we are creating a model (to be detailed below) that will be resolving the dialects mistakes Jordanians' fall into, where each city or even villages in a city has its own way of pronunciation, such as the people from some villages in Irbid; whereas in modern standard Arabic (MSA) the correct form of "إذرب" is "اضرب" or some of the people of Amman they pronounce "عقرب" as "عأرب", as for the exaggeration in a moment of a player scoring a goal, they would exaggerate writing the name "نيماااااااااار" instead of typing "نيمار".

We are enforcing a new token-free character-level model called ByT5 to adjust words from the wrong form to the right one, where we found that our model was suitable to correct such type of mistakes with high rates of accuracy and low rates of character error rate (CER).

1.7 Thesis Methodology

The methodology stages followed for our thesis are:

- Studying and surveying different types of spelling mistakes correction systems that are based on ML.
- Gathering and preparing datasets to be used for the chosen ML model.
- Studying and analyzing the dataset.
- Testing and evaluating the model by using standard performance metrics such as CER, and accuracy.
- Improving and developing the model by keeping the edit process of the model for better performance.
- Documenting the work to present the outcome.

1.8 Thesis Outline

The rest of this thesis is structured as follows:

- Chapter 2 (Literature Review) reviews the previous work where researchers dealt with spelling and auditory correction for Arabic and other dialects.
- Chapter 3 (Data) discusses datasets processing.
- Chapter 4 (Machine Transcription Models) discusses the model used and the coding steps in detail.
- Chapter 5 (Experimental Setup and Results) explains how we assembled our model and shows the results.
- Chapter 6 (Conclusion and Future Work) provides the conclusions of our work and ideas for future research.

Chapter Two: Literature Review

This chapter examines the recent models and techniques researchers created to solve Arabic and other language lingual mistakes. Previously, most models worked without ML techniques, but recently, ML has become more common, especially BERT. For other languages' lingual errors, the transformer model is already in use.

2.1 Arabic Spelling Mistakes Correction

The following are some approaches that were used to correct Arabic spelling mistakes in general, not only machine learning techniques.

2.1.1 Machine Learning Approaches

Azmi *et al.* (2019) proposed a model that detects and corrects semantic or context-sensitive spelling errors in the Arabic language. For the detection part they proposed using stem *n*-gram (n = 1-3) and ploy word beside ML: the results were 83.5% for precision and 99.2% for recall metric. As for the correction, an accuracy of 98% accrued using the n-gram language model. Their model was performing well even with a high percentage of error words, so they considered their model suitable for post-OCR recognition of the Arabic language text.

Alkatib *et al.* (2020) developed a system for detecting and correcting spelling and grammar errors at the word level, using a bidirectional long-short-term memory (BiLSTM) mechanism along with word embedding. They used a corpus that contained 15 million Arabic words, collected by them from varied text sources and revised by specialists. Their system outperformed Microsoft Office 2013 and Open Office Ayaspell 3.4, with an accuracy of 93.89%.

Al-Qaraghuli *et al.* (2021) tried fixing four types of soft spelling mistakes in Arabic. Their paper suggested using Transformers to resolve the confusion between different shapes of *hamza*, the shape of *alef* at the end of words, the insertion and omission of alef after *waw aljamaea*, and to fix the confusion among *teh*, *teh marbuta* and *heh* at the end of words. Using artificial errors for both training and evaluation sets, on a percentage of 30% on the wiki-40b dataset and using the Tashkeela dataset, they achieved an accuracy of 97.37% on correcting the artificial errors that were injected into the test set and for the set that contains factual soft spelling mistakes they achieved a CER of 0.86%.

Abandah *et al.* (2022) corrected some of the mistakes Arabs fall into while writing in the Classical Arabic language. Their research suggested BiLSTM network, and they've used it to correct spelling errors at the character level. This matter was handled by treating it as a one-to-one sequence transcription matter with a considerably light encoder-decoder architecture. BiLSTM training is done using multiple approaches: transformed input and stochastic error injection. With an injection rate of 40%, the best model they've come up with corrected 96.4% of the real test set with a CER of 1.28%.

Laaroussi *et al.* (2022) proposed a method that is based on two systems, the Viterbi algorithm, and a probabilistic model, where they combined both a distance and an n-gram language model. Using an Arabic multipurpose corpus, the probability model learned. This method has achieved a correction accuracy of 93.6% and they justify that percentage by the error that can be caused because of the strong links between each word's actual meaning and what it should be depending on the context.

For Arabic documents' detection and correcting, C. Zribi (2023) has proposed a combination of three embedding techniques, FastText, SkipGram, and BERT, -and named it "Easy" meta-embedding- these three techniques built a system that checks the word's semantic affinity depending on both the immediate context and the closest context of the studied sentence. He has trained the system on Arabic FastText, AraVec v 3.0, and AraBert v.1. This new combination has resulted in a precision percentage of 91.2%, recall of 85.6%, and F-measure of 88.3%, which is noticeably better performance than each individual embedding technique.

In the context of correcting Arabic spelling mistakes correction, (Hasan and Abandah, 2023) have revealed a model that corrects seven different spelling mistakes. This model is based on Transformers model and is implemented on two different datasets, the wiki-40b/ar dataset, where they injected errors synthetically, and according to the weights from this dataset results, they've implemented it on their personally collected dataset which they called REALMS (uploaded to GitHub) for both training and testing methods. The model they have proposed achieved the result of 99.12% for accuracy, a CER of 1.14% on the wiki-40b/ar, and an accuracy rate of 98.85% on REALMS dataset.

2.1.2 General Approaches

Shaalan *et al.* (2010) proposed a mechanism that implements a rule-based edit distance algorithm. They first address the common error patterns that Arabic learners fall into while typing and propose a two-layer spell-checking approach. Then, they suggested an error detection mechanism that they would apply along with Buckwalter's Arabic morphological analyzer, all in favor of creating something to detect the errors of spelling mistakes. As said, the suggested approach

is a rule-based approach to see the patterns of mistakes they fall into and use a multi-filter mechanism to give the best results. Their system has achieved over 80% in recall metrics and over 90% in precision metrics.

For the automatic correction of Arabic text shared task, Mubarak and Darwish (2014) have proposed two models, correction models and punctuation recovery models For the correction model, they have used a character-level model, which is assisted by a language model (LM) to handle the letters' insertion, substitution, deletion and word merges and a case-specific model where LM also handles the linguistic transformations specific types such as word as word splits and word substitution, results of models shown below in TABLE 1.

Table 1. Mubarak and Darwish (2014) Metrics Results

The Method	Precision	Recall	F-measure
QCRI-1	0.717	0.5686	0.6343
QCRI-2	0.6286	0.6032	0.6157
QCRI-3	0.6066	0.5928	0.5996

It is worth mentioning that QCRI-1 is character-level correction, case-based correction, QCRI-2 is for case-based correction and statical punctuation recovery, and the last raw which is QCRI-3 stands for case-based correction along with statical punctuation recovery based on statical punctuation recovery.

For the punctuation recovery models, they used a statical word-based system approach and an approach by (Lafferty *et al.*, 2001) called conditional random fields (CRF) sequence labeler. By implementing their models on the QALB dataset (Zaghouani *et al.*, 2014), they came up with the following results: For the correction models, they found out that by combining the character-level model and case-specific correction, they got the best results of 0.615 F-measure. In the

punctuation models, the results showed that the Stat model had an F-measure result of 0.204, the same result the CRF model held.

The approach that AlAmri and Teahan (2019) proposed was to fix Dyslexic Error by using the Prediction by Partial Matching (PPM) scheme for targeted text compression, where the system predicts from the context what can be the word we could replace instead of dyslexic one. The operations used in this system were insertion, deletion, transposition, and substitution, also known as edit operations. Compared with previous tools, their system achieved good results in terms of these metrics, recall, precision, F1, and accuracy. They have achieved 43%, 89%, 58%, and 81%, respectively.

2.2 Spelling Correction in Other Languages

A combination of the sequence-to-sequence model and the Bahdanau Attention mechanism by Trandafili *et al.* (2019) was suggested to solve the spelling mistakes in the Albanian language. The dataset with its three portions includes several sentences as follows: The train set includes 718,812, the Test set has 143,717 sentences, and the Wikipedia dataset includes 95,812 sentences which sum to 958,116 in total. Where Wikipedia's sentences are collected from numerous legal documents, electronic books, and articles within Wikipedia. The measurement method used for this model is accuracy and the accuracy percentage was 11.3%, and they have mentioned that it is because of the random error injection and an enhancement in the used Albanian dataset.

Ahmadzade and Malekzadeh (2021) proposed a sequence-to-sequence model along with an attention mechanism for correcting Azerbaijani spelling mistakes. The attention mechanism solves the problem of the final encoder's state

because since the input is usually a long sentence in this proposed work, the encoder can construct valuable information. This proposed model was trained on the 1200-sentence dataset, which had both right and wrong words, and tested on 1000 misspelled words; after the testing, the scores were 75% F1 score, 0.90% for distance one, and distance two scores were 96%.

Wang and Xie (2022) suggested an adversarial multi-task learning method that uses Semantic detection to correct Chinese text. It focuses on improving the modeling and detection capability of the character polysemy of the sentence context, by applying both masked language and scoring language models as adversarial learning tasks. As for the detection and correction technique, they've applied a policy network along with a Monte Carlo tree search strategy, and their experiment was performed on three datasets which, are Chinese Language Understanding Evaluation Benchmark (CLUE), Chinese Grammatical Error Diagnosis (CGED-2018), and their own dataset, Xuexi dataset Table 2. Shows the results of the ablation results for both CLUE's and Xuexi's.

Table 2. Ablation and BLUE results (Wang & Xie, 2022).

Dataset	Accuracy	Precision	Recall	F1 Score	BELU
CLUE	62.9	45.0	44.9	43.3	0.629
CGED-2018	-	-	-	-	0.643
Xuexi	61.1	44.3	43.1	41.9	0.738

Irani *et al.* (2022) introduced an automatic tool using a supervised deep learning approach that gains from a conditional random field (CRF); this approach has been used for both the Persian Language and Arabic Languages to help with the detection and correction of spelling mistakes. The dataset they used contained 220,000 sentences in both Arabic and Persian languages, where Persian letters are

the same as those in Arabic. While error injection, research have decided to follow two methods, first was injecting 4 errors percent, and the second was one error injection per sentence, the accuracy of implementing their approach was 60.26% for 4 errors per sentence and 83.46% for 1 error per sentence.

Below in Table 3 is a simple comparison between the previous article's results.

Table 3. Summary of the Literature Review.

Author(s)	Language	Method	Score
Shaalan *et al.* (2010)	Arabic	Rule-based Edit Distance Algorithm	Recall = 80%
Mubarak and Darwish (2014)	Arabic	Statical Word-Based System	Recall = 56% F_1 = 63%
AlAmri and Teahan (2019)	Arabic	Prediction by Partial Matching	Recall = 43% F_1 = 81%
Trandafili *et al.* (2019)	Albanian	Sequence-to-sequence Model and Bahdanau Attention	D1 = 90% D2 = 96% Accuracy = 11.3%
Azmi *et al.* (2019)	Arabic	*n*-gram & ML	Accuracy = 98%
Alkatib *et al.* (2020)	Arabic	BiLSTM	Accuracy = 93.89%
Al-Qaraghuli *et al.* (2021)	Arabic	Transformers	Acc: 97.37% CER: 0.86%
Ahmadzade & Malekzadeh (2021)	Azerbaijani	Sequence-to-sequence Attention Mechanism	F_1 = 75%
Abandah *et al.* (2022)	Arabic	BiLSTM 40% EIR	F_1 = 90.7% Accuracy = 96.4% CER = 1.28%
Laaroussi *et al.* (2022)	Arabic	Levenshtein Distance	Accuracy = 93.6%
Wang & Xie (2022)	Chinese	BiLSTM	F_1 = 93.89%
Irani *et al.* (2022)	Persian	BiLSTM 4% & 1% EIR	Accuracy = 60%
C. Zribi (2023)	Arabic	BERT	Recall = 85.6% F_1 = 88.3%

2.3 Arabic Spelling Mistakes Datasets

There was a huge lack of datasets that contained diverse Arabic Errors for researchers and developers to use until (Aichaoui *et al*., 2022) gathered a corpus of almost 1 million words where they gathered the errors from both old and new newspapers and libraries. These errors fall into multiple categories: the addition or multiplication of a character, the error of deleting a character, the replacement of a character, the wrong order of characters in a word, the phonetic errors, the wrong recognition of a character when handwriting, the error people falls into while typing on a keyboard and mistakenly swap the meant character with a neighbor one, the error of spacing between characters while typing on a keyboard, and the errors of morular changing certain characters depending on their culture.

Chapter Three: Datasets

This chapter presents the datasets used to train and test our model. We explain how we downloaded/collected them and prepared them. Eventually, we analyzed them.

3.1 Introduction

Our thesis used two datasets, wiki-40b/ar (a part of wiki-40b) and TestSets200 (TestAud200 & TestExg200), detailed below.

The wiki40b is a big collection of data collected in more than 40 languages and 40 billion characters, where it is free of errors; as the name indicates, it is from Wikipedia's articles (Al-Rfou *et al.,* 2020).

TestSets200 is the most authentic way to collect real-life data by getting it directly from the source, so the method used in our thesis is collecting data from some internet platforms, such as Facebook, Twitter, and LinkedIn. This data set has 200 examples: 200 for the auditory errors and 200 for the exaggeration errors.

The letters focused on in auditory errors are eight, shown in

Table 4. With their Unicode Names and the correct letter opposite to it.

Table 4. Letters of Auditory Errors

The Letter in Arabic	Unicode Name	The Error	Unicode name
ث	*heh*	ت	*teh*
ذ	*thal*	ز	*zain*
ض	*dad*	د	*dal*
ظ	*zah*	ض	*dad*
ط	*tah*	ت	*teh*
ق	*qaf*	أ	*alef with hamza above*
ص	*sad*	س	*seen*
ة	*the marbuta*	هـ	*heh*

3.2 wiki40b

The version used in our model is the Arabic version of Wiki-40b, wiki-40b/ar split into three parts: a train set with 220,885 examples, a test set with 12,271 examples, and a validation set with 12,271 examples.

3.2.1 Collection

The wiki-40b was imported directly online data on TensorFlow, in the table below Table 5 are some examples.

Table 5. Examples from wiki-40b/ar.

Index	Text	Set
1	يفضل توعية الطفل لتجنب استخدام اللعب التي تحدث أصواتاً عالية وعدم استخدامها بالقرب من أذنه.	Train Set
2	دراسة وتصحيح وفهرست المخطوطات الإسلامية وعلم الأنساب، وغيرها.	Train Set
3	يبلغ عدد سكان مدينة ميافارقين (سيلڤان) اليوم حوالي 90 ألف نسمة	Validation Set
4	معركة أم الجماجم هي أهم معركة شهدتها قرية التويثير بالأحساء.	Validation Set
5	استغرقت الرحلة ثمانية أشهر ونصف، قطع المسبار خلالها مسافة 570 مليون كيلومتر.	Test Set
6	أنشئت عليه ذاكرة لتخزين البيانات بحجم 8 جيجابايت، يمكنها تخزين 4000 صورة.	Test Set

3.2.2 Preparation

A. In this step, we are performing a set of steps to prepare the data after the collection process, the steps followed are:

- Clean the set off any impurities, such as empty rows.
- Edit the wiki40b/ar to a more convenient dataset for my model.
- Inject errors, either the auditory or the exaggeration errors into the clean text in wiki40b/ar dataset.

B. The cleaning process differs from one dataset to another. Still, in our case, when we downloaded the wiki40b/ar, we found that in each paragraph, there were empty lines, which wasn't very efficient since we were trying to reduce the massive size of wiki40 as much as we could. That's why we removed the line using a Python code on google.colab environment. From the validation folder, only 155,300 lines were removed and became 71,447 lines (an almost 54% decrease).

C. wiki40b/ar offers the dataset already divided into three parts, the training portion, the evaluation portion, and the test portion, where each one will be used in the model to accomplish the needed process according to its name.

D. The cleaning process differs from one dataset to another, but in our case, when we downloaded the wiki40b/ar we found that in each paragraph, there were empty lines, which wasn't very efficient since we were trying to reduce the massive size of wiki40 as much as we could, that's why we have removed the line using a certain code, where from the validation folder only there was 155,300 lines and became 71,447 lines (almost 54% decrease). As for the dataset we have gathered, while gathering it, we tried our best to take the cleanest version of it from the beginning.

E. The test set was cropped to 5,000 lines, and then, for the exaggeration line of code, the set was edited to have ten words per line for a more efficient evaluation.

3.2.3 Error Injection Rate

This process is one of the key steps in our thesis, where we must perform stochastic error injection with different error injection rates and see which one gives us a better result comparing to previous papers published with the same spelling

mistakes or similar ones, depending on certain measuring metrices. We used two lines of code, one for the letters people usually replace according to their dialect and the second for the letter duplication for most of the Arabian letters caused by exaggerated tones. In the first line of code, which is the auditory error correction, we are following the error injection rates 25% error injection rate (EIR), 50% EIR, 75% EIR, and 100% EIR for the auditory errors, but as for the exaggeration errors, these percentages are not suitable because the duplicated letters are frequently used and impossible to be used with high percentages, hence the EIRs are lower, where the rates injected are 5%, 10%, 15,% and 20%.

3.3 Test Sets

The set was collected manually with real-life errors from social media platforms, such as Facebook and Twitter, aka X nowadays, and LinkedIn and Mawdoo3 websites.

3.3.1 Introduction

Since correcting auditory and exaggerated spelling mistakes is our research problem, a dataset containing these spelling errors must exist.

The most authentic way to collect real-life data is getting the data directly from the source, so the method used in our thesis is collecting data from some social media platforms, Facebook, and Twitter, to be precise. Another way was from Linked In, just to prove that the rate of spelling mistakes in social media platforms is significantly higher than in official internet platforms.

3.3.2 Collection

Data was collected from different account fields. The data was in the form of tweets, which are sentences or short paragraphs—not more because Twitter's character count is 280 per tweet—written by different public account owners.

The collection process took seven days, due to the aim of having a diverse error for both spelling mistakes type. The quantity of this data was small, but the main aim was to test the code on real-life examples, as shown in

Table 6 and

Table 7.

Table 6. Examples from TestAud200.

Example	Error	Corrected	Channel
قد تنسى من شاركك الظحك.	الظحك	الضحك	Twitter
الأم شمعة تظيء ليل الحياة.	تظيء	تضيء	Facebook
موت الأب هو حسرة وكسرة ضهر للأبد.	ضهر	ظهر	Facebook
هل تسدق أنها فكرة تسويقية رائعة	تسدق	تصدق	Facebook
رسالتين تلات شبيهات بموضوعك.	تلات	ثلاث	Facebook

Table 7. Examples from TestExg200.

Example	MSA	Error	Corrected	Channel
السؤال جداااا عام وغير دقيق	السؤال جدا عام وغير دقيق	السؤال	السؤال	Facebook
وانا توترت كثيراً مليووون وقد رجعنا لنأتي بهم	وانا توترت كثيراً مليون وقد رجعنا لنأتي بهم	مليووون	مليون	Facebook
يا شعوب العالم هيا نقم مظاهرات نضغط على الحكام هيا نوقف حياتنا وشغلنا لأجل غزة ارجوكم اصحووووواا ناس تموت اخوتنا سوف نتحاسب عليهم	يا شعوب العالم هيا نقم مظاهرات نضغط على الحكام هيا نوقف حياتنا وشغلنا لأجل غزة ارجوكم اصحوا ناس تموت اخوتنا سوف نتحاسب عليهم	اصحووووواا ا	اصحوا	Instagram
المباراة مبيووعة	المباراة مبيوعة	مبيووعة	مبيوعة	Facebook
فقط ٥ دناااااانير	فقط ٥ دنانير	دناااااانير	دنانير	Instagram

3.4 Datasets Analysis

As for wiki-40b/ar datasets, below is Table 8, are some characteristics.

Table 8. Comparison between used Datasets

Criterion	wiki-40b/ar		
	Training	**Validation**	**Test**
Size	278MB	~40MB	39MB
Sequence count	514,799	71,447	71,092
Word count	27,405,965	3,889,671	3,854,820
Word Per Sequence	53.24	54.44	54.22
Letters per Word	4.89	4.90	4.91
Sequence ≤ 300 chars	372,875	51,425	51,399
Character count	134M	19M	70K
Maximum Sequence Length	47,638	20,914	30,887

In Table 9 and Table 10 , a new column labeled "Train for Evaluation (Eval) Subset" has been introduced, derived from the Train Set for the purpose of evaluating metrics. This subset was created due to the large size of the Train Set, ensuring compatibility with our TensorFlow evaluation model.

Table 9. Auditory and Exaggeration Model Datasets Comparison

Criterion	wiki-40b/ar (Edited to Suit My Model)				TestAud200	TestExg200
	Train Set	**Train for Eval Subset**	**Validation Set**	**Test Set**		
Size	20 MB	2.07 MB	2.05 MB	1.03 MB	10 KB	10 KB
Sequence count	80,000	4,000	4,000	2000	200	200
Word count	1,978,246	207,115	202,846	100,459	1,258	1,300
Word Per Sequence	50.75	51.77	50.7	50.2	6.26	6.5
Letters per Word	4.84	4.79	4.84	4.90	4.36	4.42
Character count	11,556,221	12M	1,185,494	592,834	6564	6648
Max Sequence length	300	300	300	300	200	200

As for the errors we are fixing, here are some statistics about the characters in Table 10.

Table 10. Dataset characteristics.

The Character	Character in data		Percentage in data	
	Train set	**Test set**	**Train set**	**Test set**
The Auditory Letters				
ث	135,302	6,669	0.68%	0.71%
ذ	114,115	5,464	0.58%	0.58%
ض	119,809	5,744	0.61%	0.61%
ظ	40,801	2,033	0.21%	0.22%
ط	189,430	8,433	0.96%	0.89%
ق	394,620	18,264	1.99%	1.94%
ص	174,805	8,247	0.88%	0.88%
ة	676,029	31,828	3.42%	3.38%
Total	1,844,911	86,682	9.32%	9.20%
The Exaggeration Letters				
ا	2,879,548	136,296	14.55%	14.46%
و	1,102,655	53,379	5.57%	5.66%
ي	15,721,99	75,776	7.95%	8.04%
Total	5,554,402	265,451	28.07%	28.17%
All Characters	45,249,246	2,159,629	-	
The count without spaces	19,787,029	942,321	-	

Chapter Four: Machine Transcription Model

In this chapter, we will be presenting the chain of AI's NLP and how it developed until the scientist reached ByT5.

4.1 Machine Transcription

In NLP, a lot of models are proposed to solve various language issues; one of the models is Machine Transcription (MT), which is used to translate the slang language to MSA. As known, slang has no specific dictionary because of the diverse ways a human can either pronounce a word then write it accordingly, hence, this process of correcting slang to MSA is considered challenging. That's why researchers have been proposing many solutions in ML to train models where it can correct the slang to MSA with the highest rate of accuracy.

Despite these challenges, MT systems can still be trained to translate slang to MSA by incorporating a large amount of training data that includes slang expressions and their corresponding MSA translations. Additionally, domain-specific models and customized training data can be used to improve the accuracy of the translations for specific applications, such as translating social media posts or chat messages. However, it's important to note that even with the best MT systems, there may still be errors or inaccuracies in the translations, particularly when dealing with informal language such as slang.

4.2RNN and LSTM

Recurrent neural network (RNN) is a simple system and is a Neural Network (NN)'s general class that includes Long-Short Term Memory (LSTM), (Hamza, 2019).

4.3.1 RNN

The concept of RNN is shown in Figure 1 below. It can process sequential data well and create hidden states (h_t) for each inserted sequence (x_t), where the (h_t) is again used at the step after to create an output ($\hat{y}_t$), the following equation will clarify the process.

$$h_t = g(W_{hx}\, x_t + W_{hh}\, h_{t-1} + b_h)$$

$$\hat{y}_t = g(W_{yh}\, h_t + b_y)$$

h_t is the hidden state at time t

x_t is the input at time t

$\hat{y}_t$ is the output at time t

h_{t-1} is the hidden state at time $t-1$

b is bias parameter.

W is the weight parameter.

g is the activation function (Rumelhart *et al*., 1986).

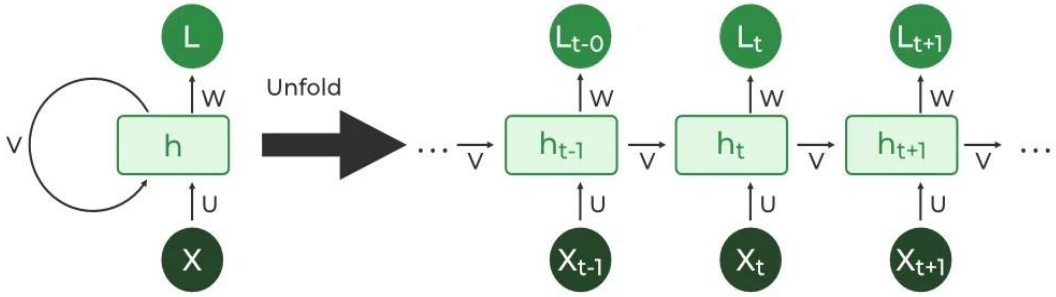

Figure 1. RNN's Structure (Srinivasan, 2023).

4.3.2 LSTM

As for the LSTM network, it's one type of RNN, and it is created to address the problem of vanishing gradients, where it is designed to maintain the hidden states for the exact previous steps -etc. epoch- (Hamza, 2019). An updated version of LSTM is Bidirectional LSTM (BiLSTM), from the name itself, it processes the sequence from both directions, forward and backward (Schuster and Paliwal, 1997), leading to an improved performance shown below in Figure 2.

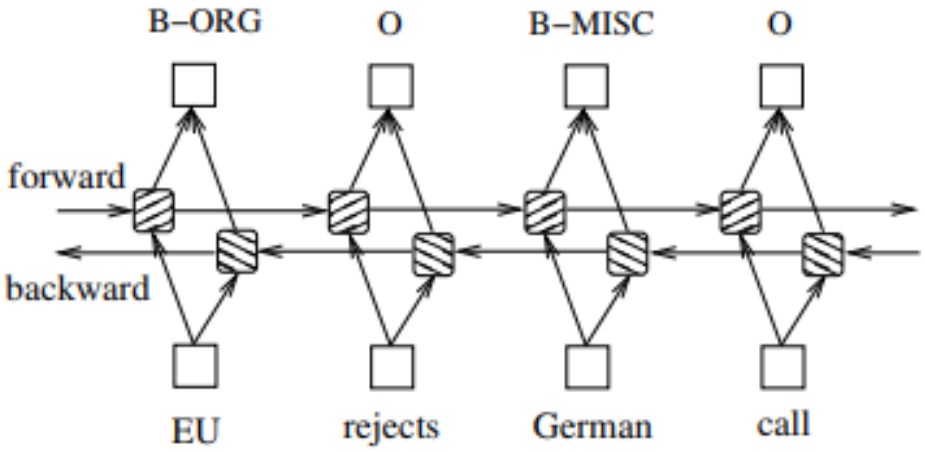

Figure 2. BiLSTM network (Huang *et al.*, 2015)

4.3Transformers

Vaswani *et al*. (2017), have introduced Transformers, shown in Figure 3, to solve the issues of language models and MT, where they called their paper "Attention Is All You Need" in preference to the attention mechanism that relates to multiple positions of one single sequence to calculate a representation of the sequence. The self-attention method has proved its efficiency in various tasks, which is why they have reused it to create transformers, but in their proposition, they are the first to rely entirely on it.

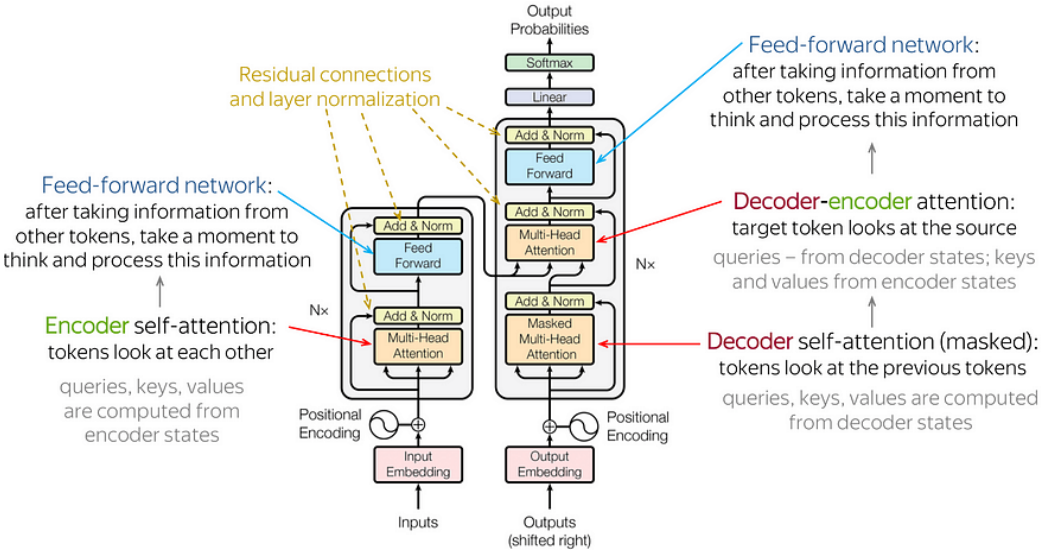

Figure 3. An overview of Transformers Architecture (Chew, 2023).

4.3.1 Embedding

In transformers, the embedding layer is where the input words are first processed, for the raw words get converted into vectors of size d. In NLP, it's a common process to embed the word from a discrete version to a continuous space of real numbers, which in a normal case of language translation model, every single embedded word corresponds to a certain recurrent neural network (RNN)/ long-short term memory (LSTM)/ gated recurrent unit (GRU) cell, here is Figure 4, to a simple example of text embedding.

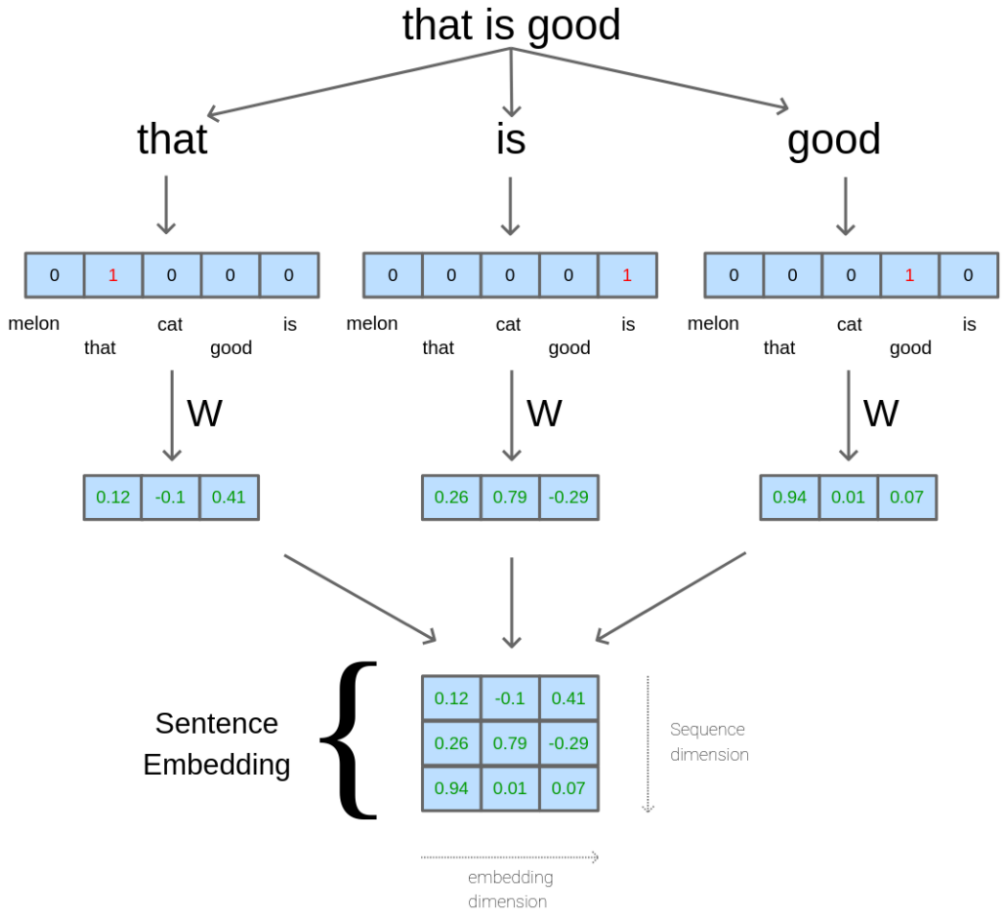

Figure 4. Flow-graph of Sentence Embedding (Phung, 2021).

4.3.2 Encoder/Decoder

As for the model of Transformers, as shown in Figure 3, it includes encoder and decoder stacks, attention, position-wise feed-forward networks, embedding and SoftMax, and positional encoding. In the following part we are explaining the model by detailing the previous components. Let us first mention TensorFlow, which is an open-source software made by google as a deep learning (DL) open-source library that uses data flow graphs as the computational procedures. In some cases, when TensorFlow can't process a specific code, for multiple reasons such as weak development processing certain applications, colab.google diverts the process to be resolved by Keras, which takes the computational graph in a simpler way compared to other frameworks. Both the encoder and decoder are simply shown in Figure 5 below, where the hidden state are all the Neural Network processes are made.

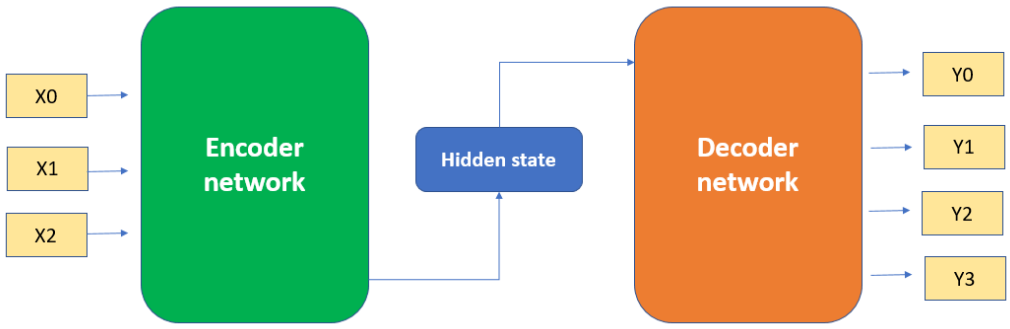

Figure 5. Transformers Encoder/Decoder (Fiagbor, 2023).

4.3.2.1 Encoder

The encoder in the transformers model is created using the concept of a layer, with six identical layers, each containing two sublayers. The first sublayer contains the multi-head self-attention mechanism, and the second sublayer is a position-wise connected feed-forward network.

4.3.2.2 Decoder

The decoder also contains 6 identical layers, each having the same as the encoder's sublayers. Still, in addition to the first two, there is a third sublayer that executes the multi-head attention over the output of the decoder stack of layers.

4.4.3 Self-Attention

As for the self-attention operation, it's well known that the architecture of Transformers is based on learning the correlations between the input segments utilizing the dot product. When $\{x_i\}_{i=1}^{n}, x \in \mathcal{R}^d$ where n is a set of words in a sequence, and i represents vector x_i 's position, the word's position in the sentence's sequence. Here comes the definition self-attention, which is the weighted dot product of vectors x_i with one another.

Self-attention is divided into two processes, where step one calculates a normalized dot product between all vector's pairs in the given input sequence, and step two is finding a new representation of x_i called z_i,which holds the summed weight of all $\{x_j\}_{i=1}^{n}$ input segments, where j $\leq n$. So, the result of vector z_i is like the input x_j with the highest attention weight, w_{ij}. Below is Figure 6, that shows the self-attentions more clearly.

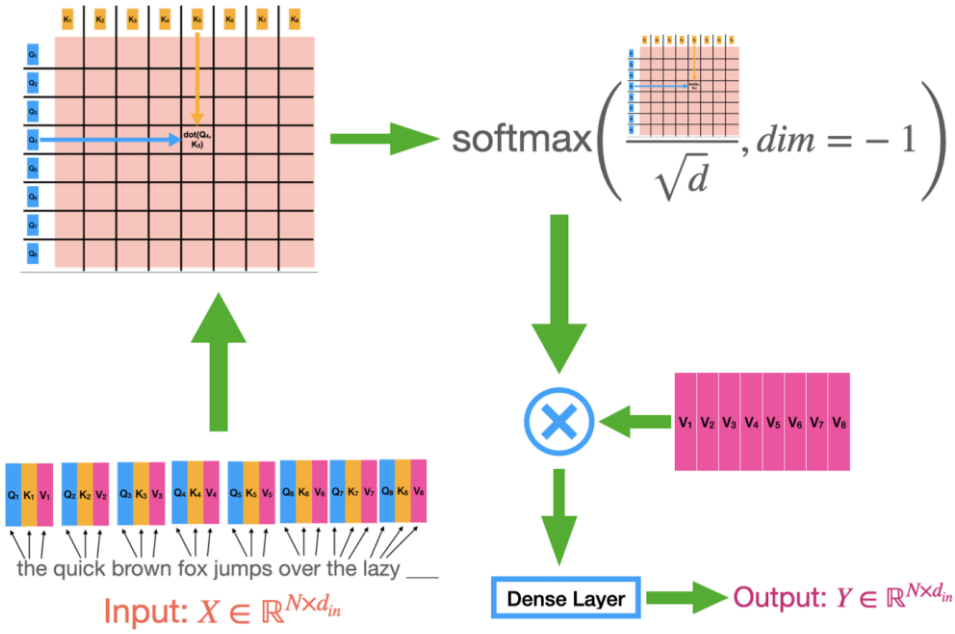

Figure 6. Self-Attention Architecture in Transformers (Sharma, 2023).

4.4.4 Feed Forward.

Feedforward (FF) layer is a layer connected directly with the residual connections and the layers of normalization within the encoder and decoder blocks. This layer includes two linear layers, which also contains an activation function called Rectified Linear Unit (ReLU), ReLU's function is to output the positive input and set the negative one as a zero output, below shown Figure 7, for a clear concept of FF.

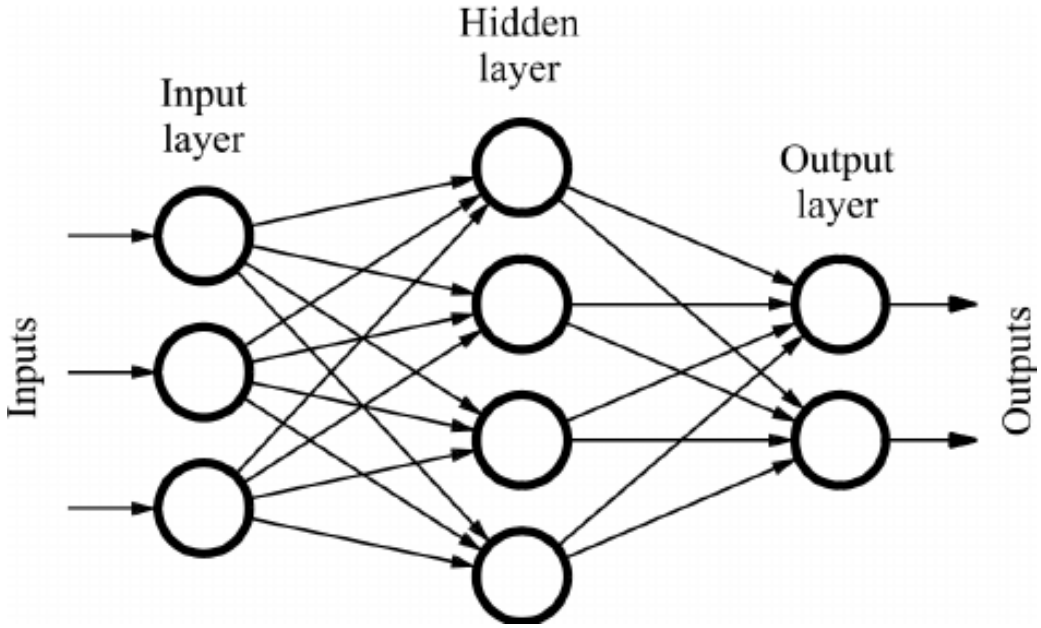

Figure 7. Feedforward's basic concept (Vaswani *et al.*, 2017).

4.4.5 SoftMax

SoftMax is an activation function that holds the highest relative probability score for a certain classification, calculated in the following formula:

$$S(y_i) = \left(\frac{e^{y^i}}{\sum_{i=1}^{N} e^{y^i}} \right)$$

Since it is extremely important to incorporate the order of the sequence in sequential data processing, transformers implement the concept of positional encoding to add the information of the input's position, then process the modified input parallelly, this way it dodges the challenges of the data being processed sequentially. This technique uses the calculation of $n\ PE\ vectors$ ($p \in \mathbb{R}^d$), adding the previous vectors to the input $\{x_i\}_{i=1}^{n}$.

4.5 Pre-Trained Transformers Models

Pre-trained transformers models are the core of modern NLP, where it enhances the transformers architecture and is very effective in various tasks such as summarization, translation etc. Here are some models of pre-trained transformers:

4.5.1 Bidirectional Encoder Representative BERT

Bidirectional Encoder Representative from Transformers (BERT) is a recent state-of-the-art NLP method, which is a stack of multiple encoders of transformers. It was considered the best in all various NLP tasks such as questions answering (QA), natural language understanding, sentiment analysis, and language inference (Ghojogh *et al*. 2020). A version of BERT called AraBERT was created for the Arabic language and has been used for the previous NLP tasks too as shown below Figure 8, explaining an example of where AraBERT is located when using sentiment analysis in NLP.

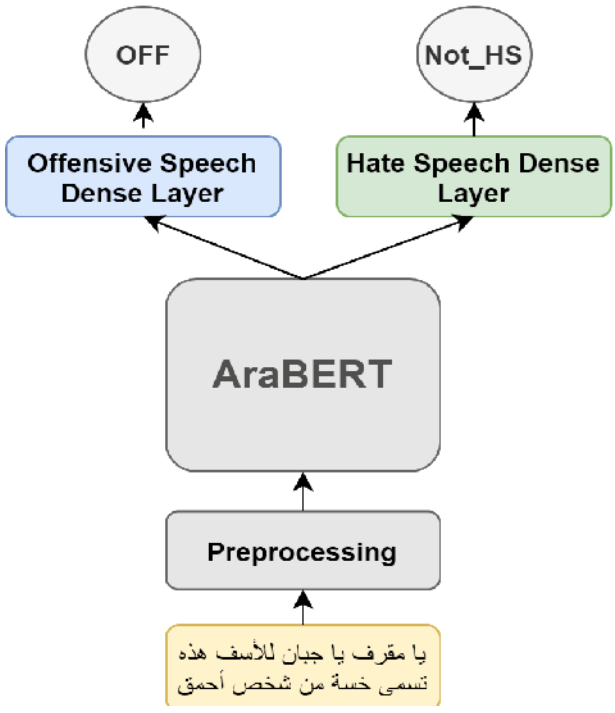

Figure 8. Sentiment Analysis using AraBERT (Djandji *et al.*, 2020).

4.5.2 Text-to-Text Transformer (T5)

T5 is an architecture of language model developed by google AI languages (Roberts *et al.*, 2020), which is built on the transformer architecture, and it masters the NLP tasks processing. In our thesis we are using the NLP task, machine translation. Machine Translation is usually defined as the process of translating from one language to another, but having a text influenced by dialects and correcting it to MSA falls into this category too.

As the architecture's name indicates, in Text-to-Text, the input is a text, and the output is produced as a new text, as shown in Figure 9, which is inspired by NLP tasks, but in T5, they have developed it to be easier to implement where for instance the fine-tuning can be implemented individually on each task. While developing Transformer and creating T5, they were considered implementing models that directly process the input with an encoder and the output with a different decoder and chose transfer learning instead of zero-shot learning. They have also adopted generative tasks like MT and abstractive summarization.

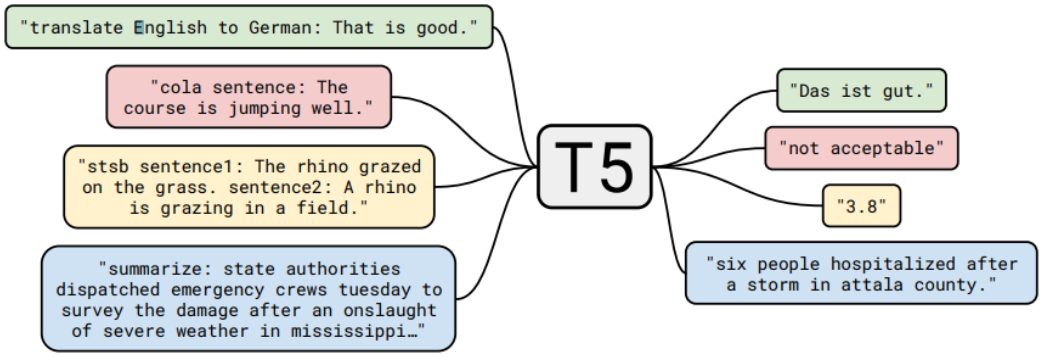

Figure 9. Diagram of Text-to-Text Framework (Raffel *et al.*, 2020).

4.5.3 mT5

mT5 is a multilingual variant of Text-to-Text Transformers (T5) pre-trained on a data set covering 101 languages. mT5 inherits many of T5's benefits, such as its scale, design based on visions from a large-scale tentative study, and general-purpose text-to-text format. mT5 was trained on a dataset called mC4, which compares text drawn from a public web scrape called Common Crawl. mT5 is based on the T5 recipe "T5.1.1", in which the T5 version is improved using GeGLU nonlinearities. To validate how this model is efficient, researchers evaluated the model on the following tasks: XNLI, which covers 14 languages; XQuAD, which covers ten languages; MLQA, which covers seven languages; and TyDi and QA, which covers 11 languages. The mT5 model was pre-trained on 1 million steps on batches of 1024 input sequences, corresponding to a total of about 1 trillion input tokens.

4.6 ByT5

ByT5 was created as a branch of T5 and it's competitive with a sub-word-level baseline. The architecture of ByT5 is very close to mT5 with some differences, such as token-free models, and embeddings. The following Figure shows an example of concepts for mT5 and ByT5 in Figure 10.

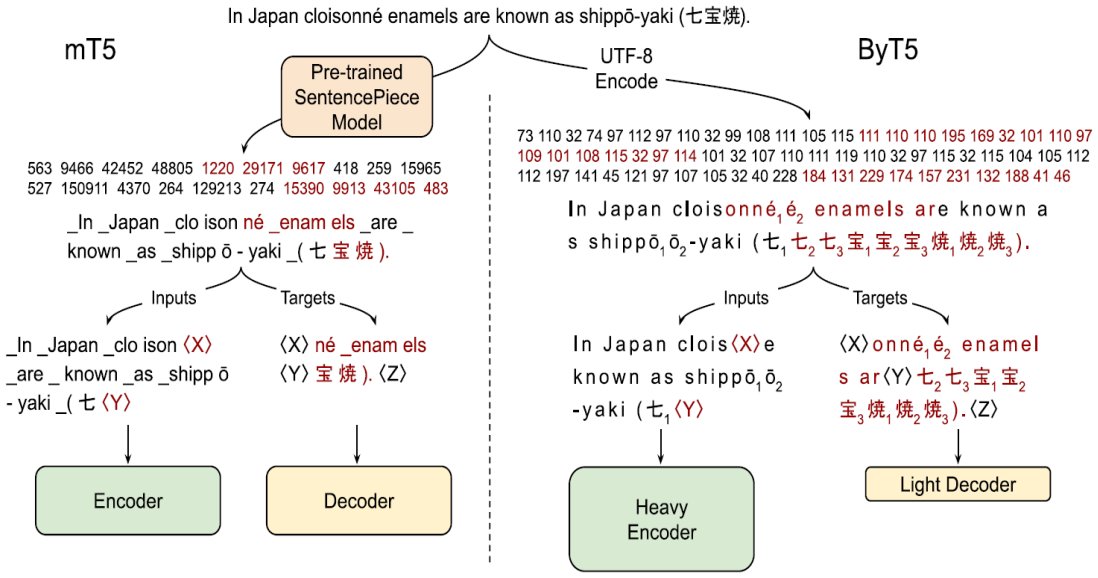

Figure 10. Pretrained Example on mT5 and ByT5's Model Concept (Raffel *et al.*, 2020).

Token-free models that operate directly on the raw text, such as bytes or characters, have different benefits; where they can process languages considered "out of the box," they minimize the technical weight by removing complexity and the error-prone text processing pipelines. They are also very robust to noise. That's why researchers took a model from Transformers and modified it as minimally as possible to process byte sequences. Then, they showed that the byte-level models are more robust to noise than token-level models and perform way better on spelling and pronunciation-sensitive tasks.

Models with fixed vocabulary usually complicate the adaptation to new terminology and new languages, but token-free models, by definition, can process any text sequence. There are a few drawbacks in byte-level models, but the main one is that byte sequence tends to be way longer than token sequences. Still, character-byte-level models have learned ways to process long sequences using

convolutions with pooling or adaptive computation time. As for the embeddings, byte-level models require 256 embeddings.

Below, specifying the architecture of ByT5 for each of the sizes: small, base, large, XL and XXL in Table 11.

Table 11. Comparison Between Byt5 Sizes (Xue *et al.*, 2022).

Size	Vocab	d_{model}/d_{ff}	Encoder/Decoder
Small	0.3%	1472/3584	12/4
Base	0.1%	1536/3968	18/12
Large	0.06%	1536/3840	36/12
XL	0.04%	2560/6720	36/12
XXL	0.02%	4672/12352	36/12

4.6.1 Byt5 and mT5 comparison

Even though byT5 is a better and more enhanced version of T5 (an architecture designed to handle the NLP tasks), both byT5 and mT5 are encoder-decoder-based transformer models, which google AI models generate. Below is Table 12, that compares between the byT5 and mT5 models.

Table 12. Comparison between byT5 and mT5.

Characteristics	mT5	byT5
Tokenization	SentencePiece subword	UTF-8
Pre-Training Objective	Group of tokens masking	Masking on byte-level
Architecture layout	Equal depth of Enc./Dec.	Arch.: Encoder-heavy
Data Efficiency	More pre-trained data that byT5	Less pre-trained data, same performance as mT5
Noise Robustness	-	Great tolerance to inputs that are noisy or corrupted.
Strengths	Large tasks: High acc.	Small size: More efficient
Weaknesses	Low efficiency @ small	Expensive on large seq.
Use Cases	Perfect for large-scaled	Ideal for limited training

From this comparison we are assured that byT5 is more suitable to our model especially that our training data is limited.

Chapter Five: Experimental Setup and Results

This chapter details the environment used to run our code, presents the results we have achieved from testing on collected datasets, and discusses the results compared to previous ones.

5.1 Experimental Environment

Our entire model ran on colab.google hosts the Jupyter Notebook service with no required setup. They are using the runtime specified to colab.google, TPU, which is quicker to work than GPUs, the model works smoothly (Bisong, 2019), specifications mentioned in Table 13.

As for the storage of the data, for the wiki40b/ar, we have done nine folders, three for training, three for validation, and three for testing, with sizes that vary from eighty thousand kilobytes to one and a half gigabytes. As for the small dataset we have gathered manually, it is twenty-seven kilobytes. The storage of the data was on both Google Drive and Google Cloud bucket (smadi2456) where we have created an account there for more storage, TPU's, Google Compute Engine Virtual Machine (GCE VM).

Table 13. Experimental Platform Specifications.

Aspects	Specifications
Environment	Google Colab Pro.
TPU	Designed by Google, TPU version 3 that contains two systolic arrays (128*128 ALUs), a single processor, and a single board computer
Memory	35.24 GB
OS	Linux 5.15.120+
Libraries	Eager 0.0.1, Jaxlibrary 0.4.20, keras 2.12.0,

5.2 Evaluation Metrics

To assess how well our model performed, evaluate our model's outputs, and see how good the predicted output is, we use metrics. These metrics are Accuracy, and CER.

5.2.1 Accuracy

Since spelling correction is considered a classification task, there is a sure way to produce predictions by this model, and there are four: true positive (TP), true negative (TN), false positive (FP), and false negative (FN). When the case is that a letter x is predicted to be as letter x, this is classified as TP prediction, but when a letter is not x and expected not to be a letter x, that is FP prediction; when a letter x is mistakenly predicted to a letter x, this is FP, and when a letter x is expected to something other than the letter x, then this is classified as a FN prediction.

The calculation of accuracy is calculated using the following equation:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

5.2.2 CER

Character Error Rate (CER) calculates the ratio of the wrong characters in the text. Using the following equation, the ratio is extracted:

$$CER = \frac{S + D + I}{N}$$

Putting into consideration that S is the number of substitutions,

D is for the number of deletions.

I is for the number of insertions,

N is for the number of characters in the target sequence.

The lower the results of CER, the better the performance of model is.

5.3 ByT5 Model Used

The size used in our model is small; we chose it not only because the datasets used are relatively small but also because it has been shown that the smaller the model of ByT5 is, the better the results are (Xue *et al.,* 2022). Below are Byt5/small characteristics in Table 14.

Table 14. Byt5/Small Characteristics.

Size	Byt5/Small
Vocab	0.3%
d_{model}	1472
d_{ff}	3584
Encoder/Decoder	12/4
CER	9.8

5.4 Experiments

Our model's experiment was conducted with different variables that have changed through the entire process, such as:

- Change of Error Injection Rate.
- Change of Finetune (Training) number of steps, we choose how many steps we want the system to train our model on. And according to the results we see what the best number of steps is to have the best results. It is not conditional that the more steps we train on, the better the result is. In fact, the model can be over trained and might affect the result negatively.
- Change of Testing Datasets Used.

5.5 Hyperparameters

Our model's primary hyperparameters include:

- **Batch Size:** This parameter denotes the number of samples processed before weight updates take place. Although values of 256 or more are common, larger batches might show decreasing performance enhancement benefits. We decided that the suitable batch size is **256** for our model, which is the standard for the small-byT5.
- **Maximum Sequence Length:** sets the boundary for the number of tokens allowed in a sample. This restriction is influenced by two main factors: the time complexity of the transformer model and the byte-level granularity of our model, which necessitates more tokens to represent text compared to models operating at the word level. As a result, the ByT5 model imposes a limit of 1024 tokens for the maximum sequence length. The sequence per line was set to **300** in our model.
- **Learning Rate:** The learning rate set for this model is **0.003**; this LR is set according to the researcher's (Al-Rfooh *et al.,* 2022), where the paper shows that this learning rate produces the best results for CER.
- **Optimizer: Adafactor** serves as the primary optimizer utilized in pre-trained models, employing the Adafactor algorithm. This optimizer offers a method to automatically adjust learning rates according to parameter size and demonstrates effective performance across a range of models.

5.6 Results

This is the step where we evaluate our model and show in the figures below, the models' performance according to our metrics.

5.6.1 The Auditory Errors Correction Model

This model corrected the auditory errors. The best accuracy result is 79.78% on the EIR 75%. The best CER result on the test set of Wiki-40b is 0.07%, and for the TestAud200 set, it is 1.13%.

We changed the EIRs four times with the EIRs mentioned previously: 25%, 50%, 75% and 100%.

- The Results of the Auditory errors correction model to fix 25% EIR:

This part is to set the weights that we are evaluating the test sets on based on accuracy metrics.

After comparing between the results on checkpoints upto 40k as shown in Figure 11, we noticed that after 24k checkpoints, the results stabilized, and there was no enhancement in accuracy for both the Train Set and Validation Set. Hence, the following EIR's maximum checkpoints will be at the 24th checkpoint step.

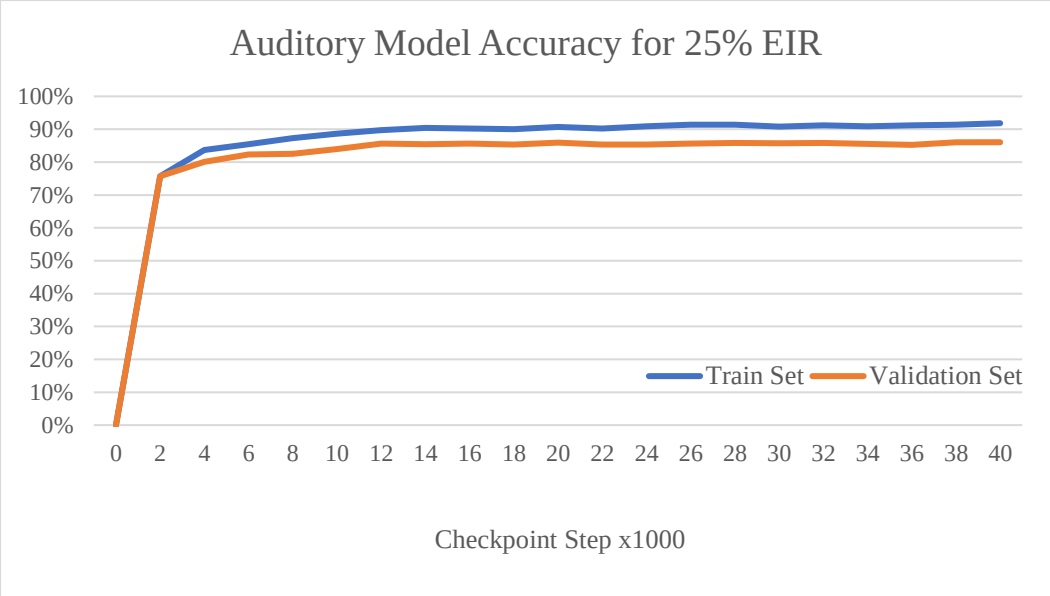

Figure 11. Accuracy Results for Auditory Model on 25% EIR.

Below is the value of CER according to the model's evaluation on wiki-40b/ar Test Set and our personal TestAud200:

Test Set evaluation on the 24th checkpoint step: CER: 0.07% and on the TestAud200 evaluation on the 24th checkpoint step: CER: 1.91%.

- The Results of the Auditory errors correction model to fix 50% EIR shown in Figure 12:

Auditory model to fix 50% EIR on wiki40b/ar edited datasets.

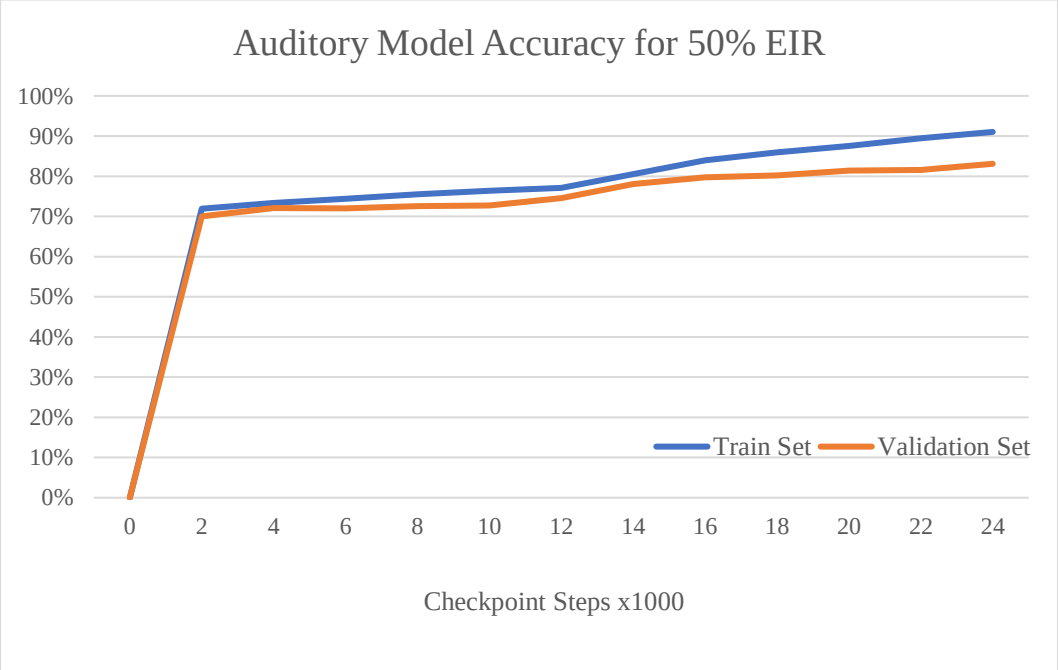

Figure 12. Accuracy Results for Auditory Model on 50% EIR.

According to the accuracy metric results, we noticed that this model's best accuracy was at the 24th checkpoint step. So below is the value of CER according to the model's evaluation on wiki-40b/ar Test Set and our personal TestAud200.

Test Set evaluation on the 24th checkpoint step: CER: 0.098% and on the TestAud200 evaluation on the 24th checkpoint step: CER: 1.44%.

- The Results of the Auditory errors correction model to fix 75% EIR shown in Figure 13:

Auditory model to fix 75% EIR on wiki40b/ar edited datasets.

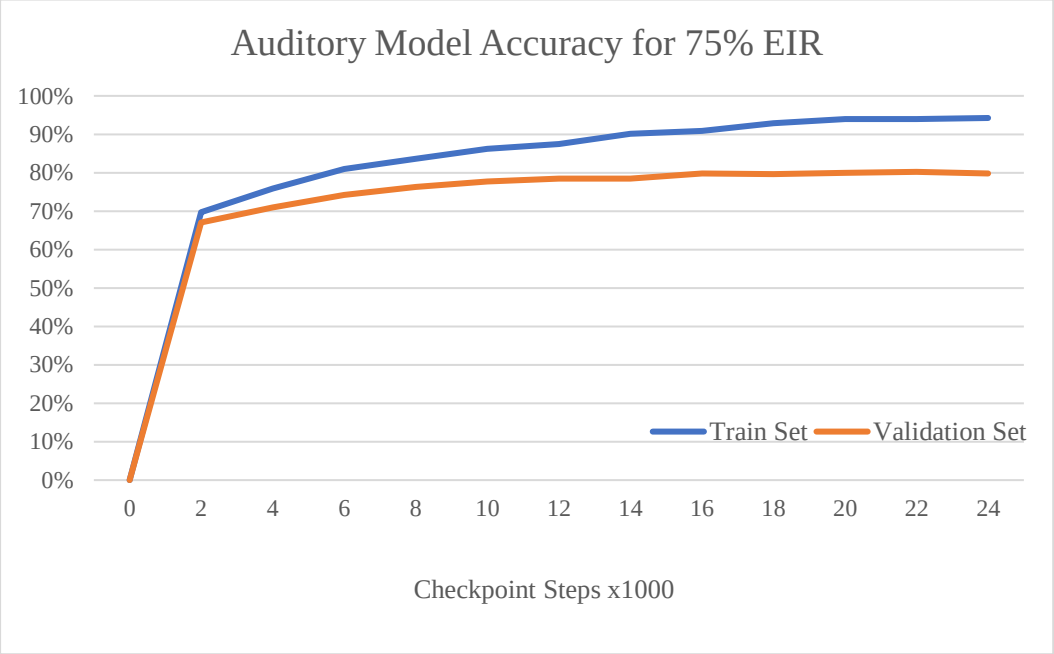

Figure 13. Accuracy Results for Auditory Model on 75% EIR.

According to the results of the accuracy metric, we have noticed that the 24th checkpoint step had the best accuracy this model can get, so below is the value of CER according to the model's evaluation on wiki-40b/ar Test Set and our personal TestAud200.

Test Set evaluation on the 24th checkpoint step: CER: 0.12% and on the TestAud200 evaluation on the 24th checkpoint step: CER: 1.13%.

- The Results of the Auditory errors correction model to fix 100% EIR shown in Figure 14:

Auditory model to fix 100% EIR on wiki40b/ar edited datasets.

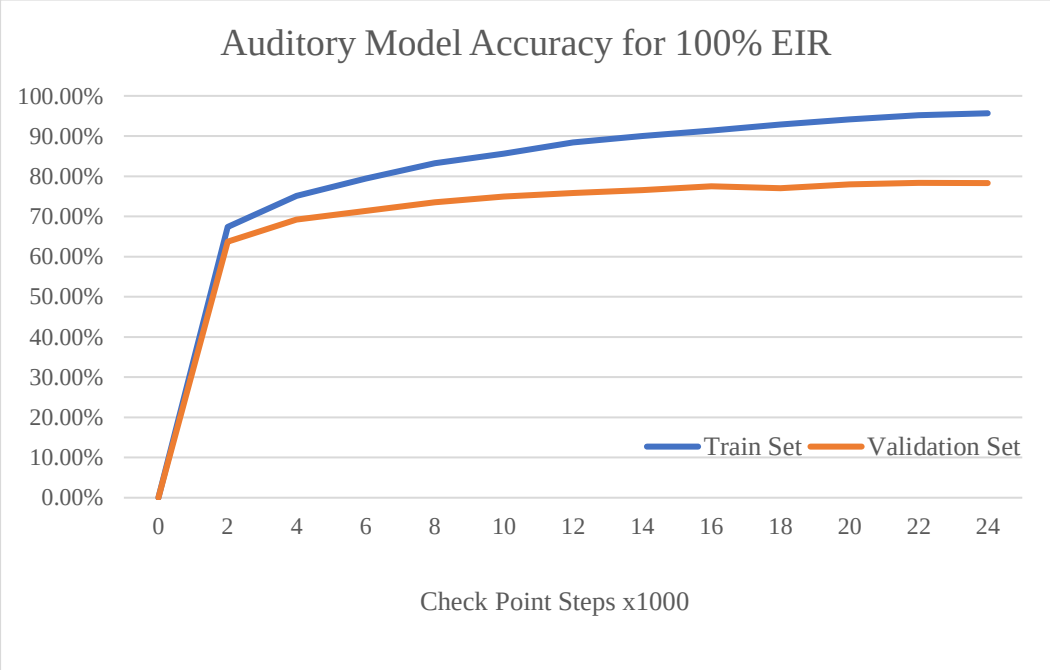

Figure 14. Accuracy Results for Auditory Model on 100% EIR.

According to the results of the accuracy metric, we have noticed that the 24th checkpoint step had the best accuracy this model can get, so below is the value of CER according to the model's evaluation on wiki-40b/ar Test Set and our personal TestAud200.

Test Set evaluation on the 24th checkpoint step: CER: 0.14% and on the TestAud200 evaluation on the 24th checkpoint step: CER: 1.25%.

- The results of Test Set wiki on the Auditory errors correction model in Figure 15 and Figure 16 below:

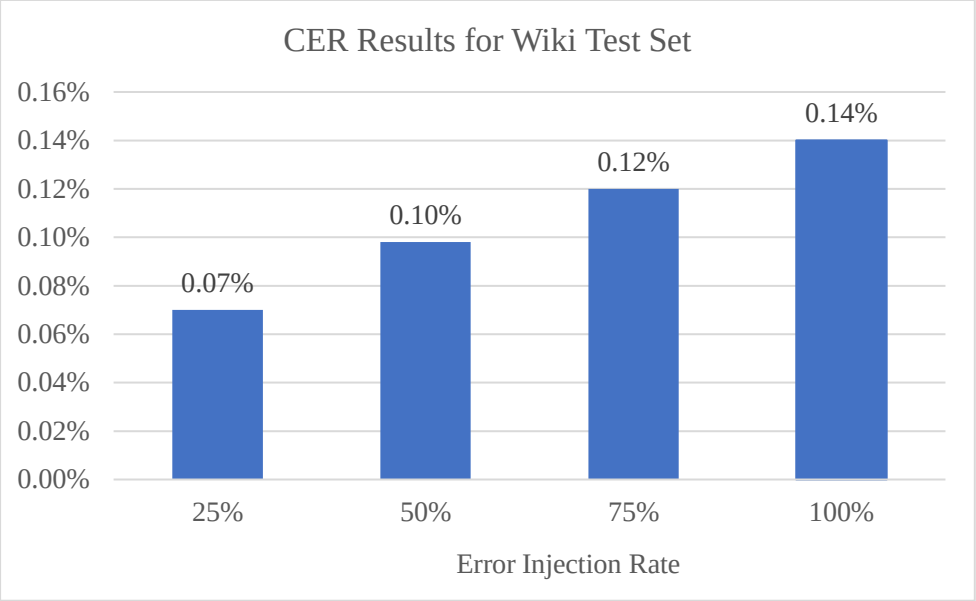

Figure 15. CER Results for Auditory Model on wiki40b/ar Test Set.

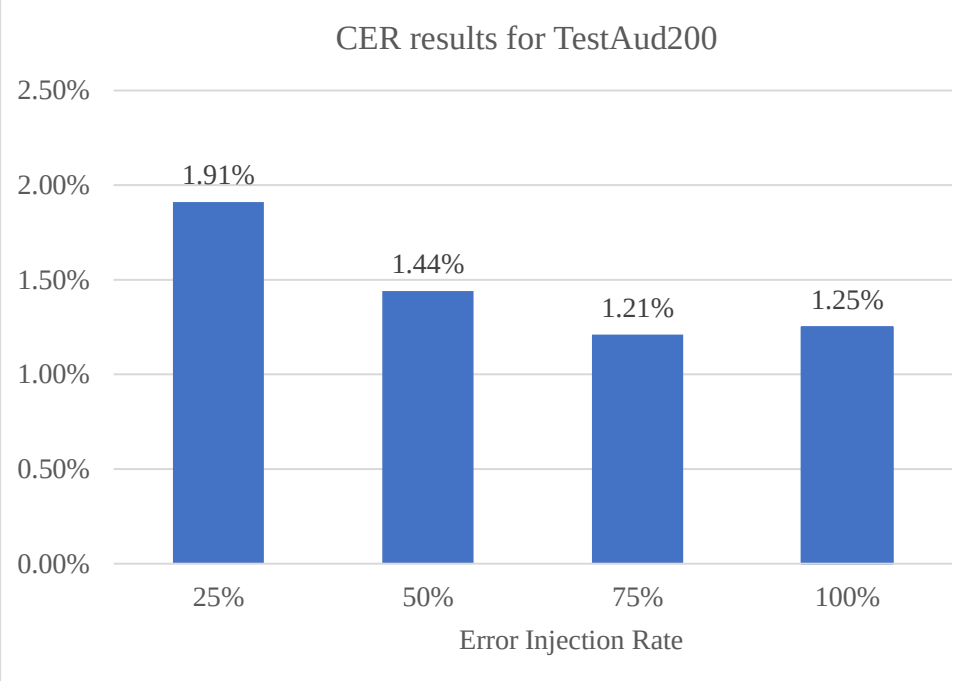

Figure 16. CER Results for Auditory Model on TestAud200.

5.5.2 Exaggeration Errors Model Results

This model corrected the exaggeration errors. The accuracy result is 99.39% for the EIR 15% and got the best CER score which is 0.38% .These results were conducted on a sequence length of 810 since the errors injection is done by duplicating letters here for the input, and 575 for the output.

We have changed the EIRs four times, with the EIR's mentioned previously: 5%, 10%, 15%, and 20%.

As previously set in the auditory errors correction model, the learning rate set for this model is 0.003, this LR is set according to the researcher's in (Al-Rfooh *et al*., 2022) where the paper shows that this learning rate produces the best results for CER.

- The Results of the Exaggeration errors correction model to fix 5% EIR:

As for the exaggeration errors correction model, setting the model to run on 52,000 checkpoint steps (ckps). The results as shown in Figure 17. On a train batch size of 256.

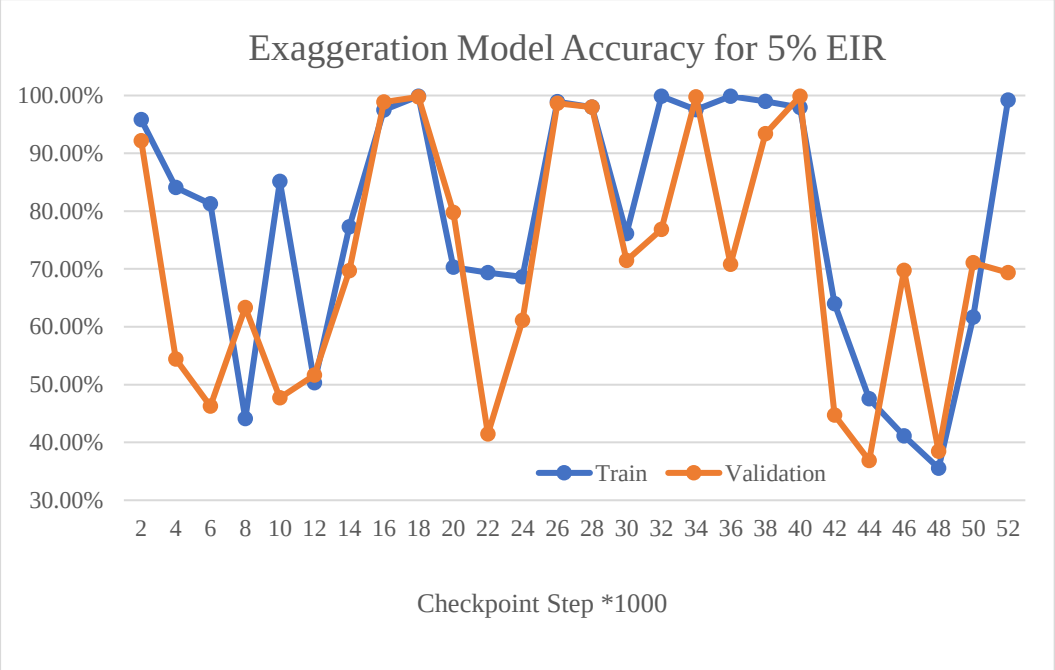

Figure 17. Accuracy Results for Exaggeration Model on 5% EIR.

Test Set evaluation on the 18k checkpoint step: CER: 0.19%, on the TestExg200 evaluation on the 18k checkpoint step: CER: 0.98% and on the TestExg200 evaluation on the 40k checkpoint step: CER: 0.52%.

After observing that the percentage of accuracy does not increase higher than the 18th k step, we have decided that in the upcoming EIRs, we will be conducting fine tuning until 24th k steps.

- The Results of the Exaggeration errors correction model to fix 10% EIR:

As for the exaggeration errors correction model, setting the model to run on 24,000 ckps. The results as shown in Figure 18.

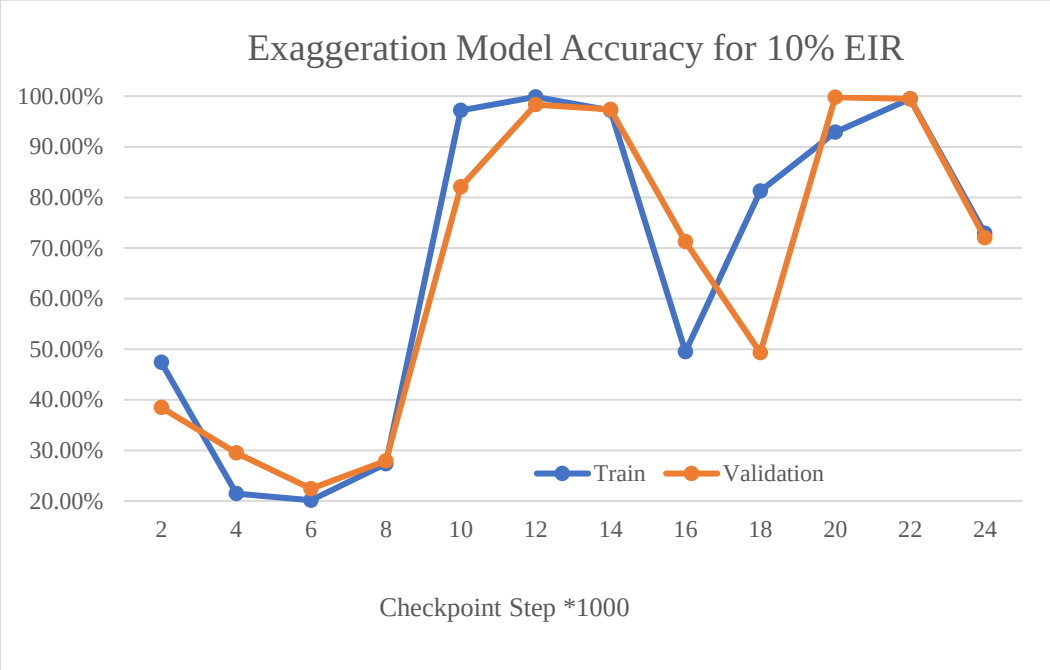

Figure 18. Accuracy Results for Exaggeration Model on 10% EIR.

Test Set evaluation on the 20k checkpoint step: CER: 0.013% and on the TestExg200 evaluation on the 20k checkpoint step: CER: 0.38%.

- The Results of Exaggeration errors correction model to fix 15% EIR:

As for exaggeration errors correction model, setting the model to run on 24,000 ckps. The results as shown in Figure 19.

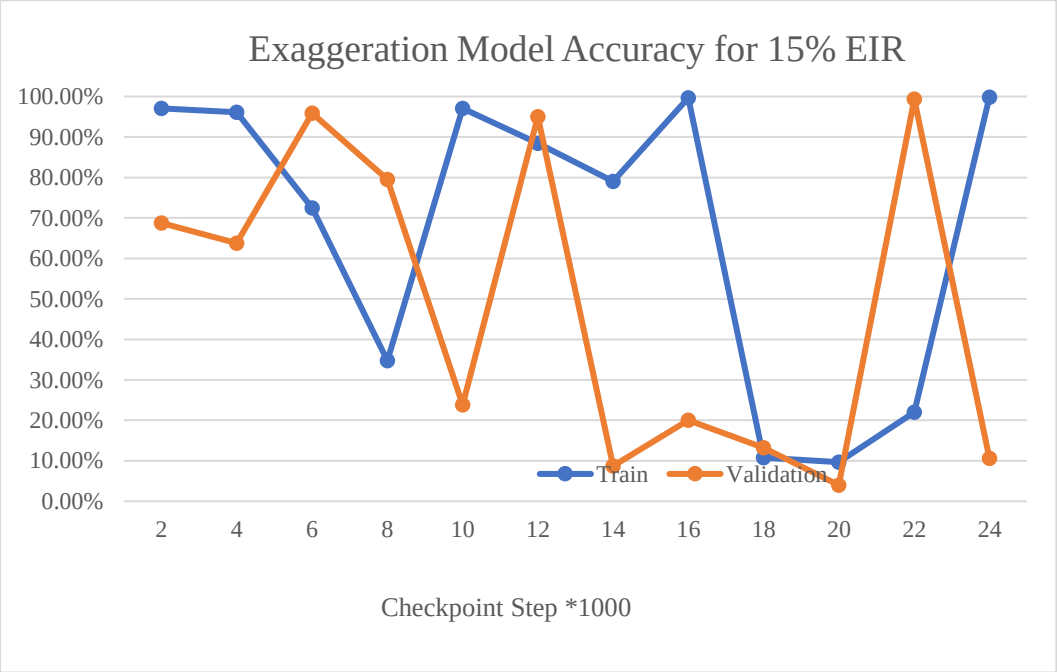

Figure 19. Accuracy Results for Exaggeration Model on 15% EIR.

Test Set evaluation on the 22ndk checkpoint step: CER: 0.75% and on the TestExg200 evaluation on the 22ndk checkpoint step: CER: 6.95%.

- The Results of the Exaggeration errors correction model to fix 20% EIR:

As for the exaggeration errors correction model, setting the model to run on 24,000 ckps. The results as shown in Figure 20.

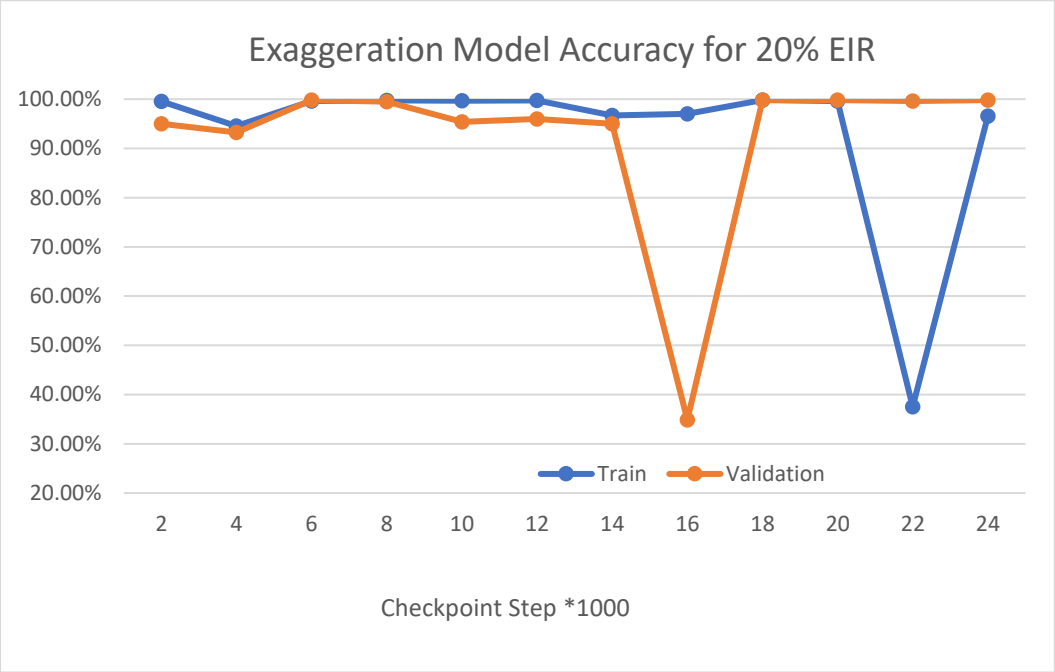

Figure 20. Accuracy Results for Exaggeration Model on 20% EIR.

Test Set evaluation on the 6thk checkpoint step: CER: 0.6% and on the TestExg200 evaluation on the 6thk checkpoint step: CER: 0.58%.

- The results of Test Set wiki on the Exaggeration errors correction model:

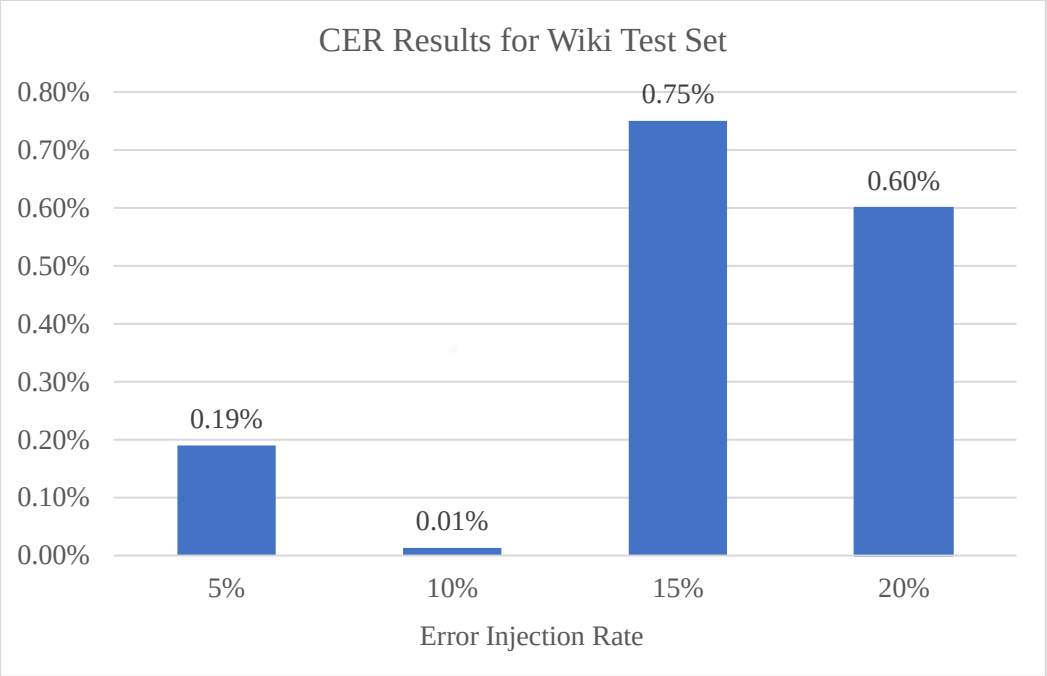

Figure 21. CER Results for Exaggeration Model on wiki40b/ar Test Set.

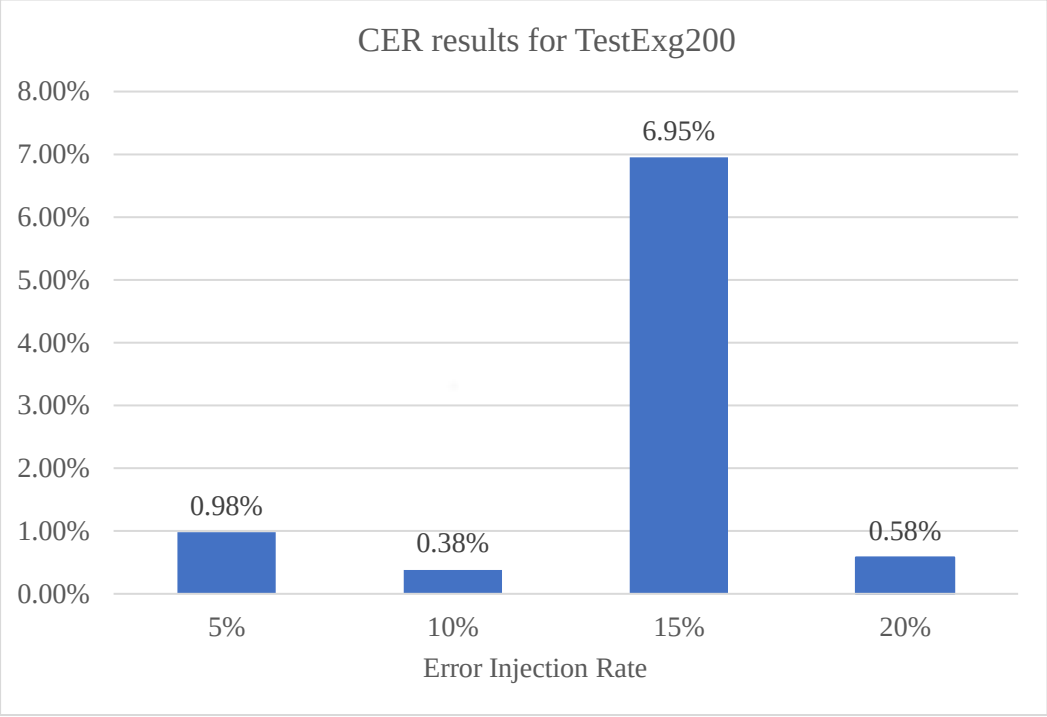

Figure 22. CER Results for Exaggeration Model on TestExg200.

5.7 Model's Fine-Tune and Prediction Timing

In our created model, we saved a checkpoint for the fine-tuning process every 1000 steps. Each checkpoint took around 36 minutes for the model that corrects auditory mistakes and 51 minutes for the model that corrects exaggeration mistakes.

As for the prediction, for the auditory and exaggeration errors correction model, checking the prediction process on the test sets takes 66 seconds, so for each sequence, it takes 0.33 seconds.

5.8 Discussion

The model trained to correct auditory errors performed its best on the EIR of 75%, with an accuracy score of 79.78% on the validation set and a CER of 1.13% on TestAud200. After evaluating four different EIRs, we noticed that there was not that big of a difference between the CER results, whether the EIR was low or high. For the highest CER, the result was 1.91% for the 25% EIR. We think the explanation for that is that byT5 is pre-trained well, but these mistakes are new to it, and the more mistakes it has, the better it learns from them.

The model trained to correct exaggeration errors on the other hand, scored an accuracy of 99.78% on the validation set and a CER of 0.38% on TestExg200 for an EIR of 10%.

After conducting the previous results in 5.5.2 Exaggeration Errors Model Results, we have adjusted the sequence length of the model from 810 to 900 hoping to have a less noisy and better results and achieved the following results in Figure 23:

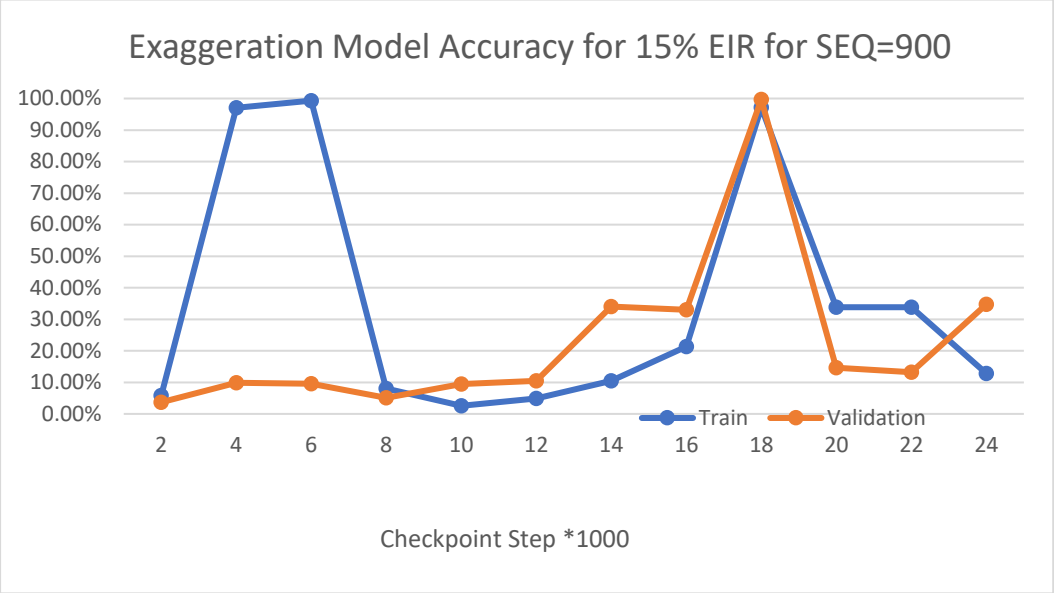

Figure 23. Accuracy Results for Exaggeration Model on 15% EIR on Sequence Length of 900.

Where the results of the test sets were as following:

Test Set evaluation on the 18thk checkpoint step: CER: 34.44% and the TestExg200 evaluation on the 18thk checkpoint step: CER: 2.69%.

Considering previous results, this is insufficient. The rise in sequence length for the exaggeration errors correction model will only result in higher CER scores, which are not preferred in our model.

After conducting the previous results in Accuracy Results for Auditory Model on 75% EIR., we have changed the sequence count from 80,000 to 200,000 in the train dataset wiki-40b/ar in the fine-tune step hoping to have a better accuracy, results in Figure 24.

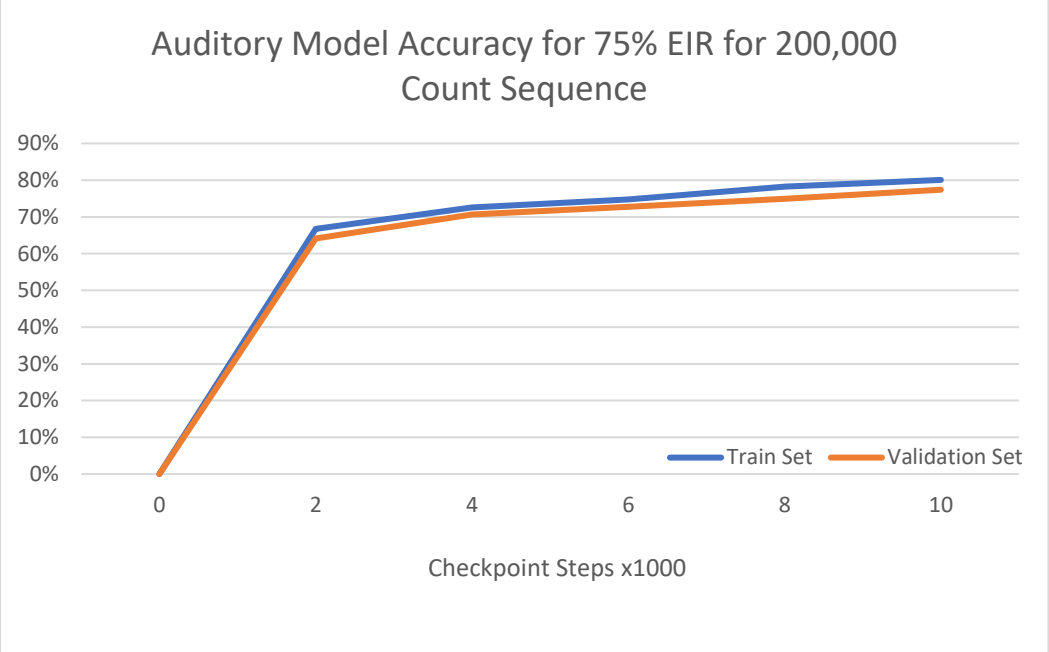

Figure 24. Auditory Model Accuracy for 75% EIR for 200,000 Count Sequence.

Comparing results of the 80,000-sequence count of the train dataset wiki-40b/ar, we've noticed that it is less in accuracy from 2-3%.

5.9Related Work Comparison

Although we cannot directly compare our model to any previous model because the models correct different types of spelling mistakes, data below in

Table 15, compared our model to some Arabic mistake correction models.

Al-Qaraghuli *et al.* (2021) tried fixing four types of soft spelling mistakes in the Arabic Language. Their paper suggested using Transformers and on a

percentage of 30% on the wiki-40b dataset and using Tashkeela dataset, they achieved an accuracy of 97.37% achieved a CER of 0.86%. Using a TPU on Kaggle, they needed 100 epochs, each epoch took 48 minutes, so for their model to come up with these results, it needed 80 hours.

Abandah *et al.* (2022) using BiLSTM network, they corrected the errors of spelling at the character level. They corrected 96.4% of the real test set with a low character error rate of 1.28% on an EIR 40%. To reach these results, the model needed 30 epochs, that needed 25 hours, where each epoch took 50 minutes.

Hasan and Abandah (2023) proposed model is to correct seven different spelling mistakes. Based on Transformers model and using the wiki-40b/ar dataset, and REALMS (their own new dataset uploaded to GitHub) they achieved the result of 99.12% for accuracy and a CER of 1.14% on the wiki-40b/ar, and an accuracy rate of 98.85% on REALMS dataset. These results were obtained after a training of 106 epoch, each epoch needed 19 minutes, so the system needed approximately 34 hours to come up with these results.

Our model studied the possibility of using Byt5 to correct two mistakes in the Arabic language, auditory and exaggeration. Our auditory mistakes correction model had several EIRs, but the best results was on the percentage 75 and achieved an accuracy of 79.78%, and even though it not the best accuracy among the other three papers, but the CER result of 1.13% overcomes Abandah *et al.* (2022) and Hasan and Abandah (2023). As for our exaggeration mistakes correction model, the best result was achieved on the EIR of 10%, where the accuracy reached 99.78% and the CER was 0.38%, which is better than Al-Qaraghuli *et al.* (2021), Abandah *et al.* (2022) and Hasan and Abandah (2023).

As for the model timing, both our models overcame Al-Qaraghuli *et al.* (2021), Abandah *et al.* (2022) and Hasan and Abandah (2023). Where our model that fixes auditory errors took 14 hours, 44% better than least timing of these papers is Abandah *et al.* (2022), for 25 hours. As for our model that fixes exaggeration errors, it took 17 hours which is 32% better than Abandah *et al.* (2022).

Table 15. A Comparison Between our Model and Multiple Published Research.

Author(s) & Method	**Data Set Used**	**EIR**	**Results**
Al-Qaraghuli *et al.* (2021), Transformers	wiki-40b & Tashkeela	30%	Acc: 97.37% CER: 0.86%
Abandah *et al.* (2022), BiLSTM	Tashkeela & Arabic Treebank Part3	40%	Acc: 96.4% CER: 1.28%
Hasan & Abandah (2023), Transformer	REALMS	30%	Acc: 99.12% CER: 1.124%
Our Model That Fixes Auditory Errors	**wiki-40b**	**75%**	**Acc: 79.78% CER: 1.13%**
Our Model That Fixes Exaggeration Errors	**wiki-40b**	**10%**	**Acc: 99.78% CER: 0.38%**

Chapter Six: Conclusion and Future Work

This final chapter is to write the thesis summary along with what we think can be added to our model to enhance its results.

6.1 Conclusion

The power of transformers models has been proven especially in the NLP field in the last couple of years. While the world is taking more interest in correcting spelling mistakes, which is a component of NLP, transformers have improved significantly over LSTM models and other models in the correction process.

In this master's thesis, we have studied the ML approach and chose Byt5, introduced by google's research team, to correct two spelling mistakes: the auditory mistakes -the replacing technique of a letter with another for the reason of an accent influence- and the exaggeration mistakes -which is the duplication technique of a letter due the influence of a certain feeling like excitement- by converting the text to a correct MSA.

Previously, no one used the model we have used which is based on the byte-level models and are more robust to noise than token-level models and perform way better on spelling and pronunciation-sensitive tasks.

In our model we used wiki-40b/ar for both finetuning and evaluation and our personal TestSets200 for the testing process to register our results. These datasets were used for auditory and exaggeration models.

Even though it is not possible to compare our model with previous models because of different error types and error injection rates, we have compared our model to some Arabic mistake correction models.

In conclusion, our auditory errors correction model can fix artificial errors with a 79.78% accuracy, 1.13% CER and our exaggeration errors model can fix with an accuracy of 99.78%, 0.38% CER.

6.2 Challenges

The challenges we faced during our thesis process are:

- Lack of knowledge of AI and NLP. When we first started working on our thesis, we had zero knowledge of AI and its types, so it took us time to educate ourselves and be able to work with it.
- Our knowledge of Python and TensorFlow was also poor.
- The environment we worked in (google.colab) offered a costly TPU (we paid monthly from 10-80).
- The storage environment was also online and owned by google, and it expired every 90 days. So, more experiments meant more time to create new google Cloud Consoles and buckets.

6.3 Future Work

Researchers can try using the emerging newer large language models such as GPT and LLaMA. They can insert in the training process more datasets collected from social media on a wider range.

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).

Ahmadzade, A., & Malekzadeh, S. (2021), Spell Correction for Azerbaijani Language Using Deep Neural Networks. **arXiv Preprint arXiv:***2102.03218*.

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).

Alamri, M. M., & Teahan, W. J. (2019), Automatic Correction of Arabic Dyslexic Text. **Computers**, *8*(1).

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

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

Al-Rfooh, B., Abandah, G., & Al-Rfou, R. (2023, May). Fine-Tashkeel: Fine-Tuning Byte-Level Models for Accurate Arabic Text Diacritization. In **2023 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT)** pp. 199-204.

Azmi, A. M., Almutery, M. N., & 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),** pp. 1308-1320.

Bisong, E. (2019), google Colaboratory. Building Machine Learning and Deep Learning Models on google Cloud Platform: A Comprehensive Guide for Beginners. **Springer,** pp. 59-64.

CherifI, D., Kaddari, R., Hamza, Z. A. I. R., & Amine, N. A. (2019). Infrared Face Recognition Using Neural Networks and HOG-SVM. In **2019 IEEE 3rd International Conference on Bioengineering for Smart Technologies (BioSMART)** pp. 1-5.

Chew, A. (2023), Transformers Architecture Explained in 5. In **AI Mind,** https://pub.aimind.so/summary-of-transformer-achitecture-c2cef6dcaca6, Last visited April 2023.

Djandji, M., Baly, F., Antoun, W., & Hajj, H.M. (2020), Multi-Task Learning Using AraBert for Offensive Language Detection. **OSACT**.

Fauntleroy, C. (2021), 5 Reasons Learning Arabic Should Be Your New Year's Resolution. **MEI,** https://www.mei.edu/education/blog/5-reasons-to-learn-arabic Last visited March 2024.

Fiagbor, H. (2023), Why Decoder-only Transformer Models are Dominating Now? **LinkedIn,** https://www.linkedin.com/pulse/why-decoder-only-transformer-models-dominating-now-harriet-

fiagbor/?trackingId=i2GNF2TMTqO4Ko3nr%2B%2FiwQ%3D%3D, Last visited February 2023.

Ghojogh, B., & Ghodsi, A. (2020), Attention Mechanism, Transformers, BERT, and GPT: Tutorial and Survey. **Papers With Code,** https://paperswithcode.com/paper/attention-mechanism-transformers-bert-and-gpt Last visited March 2024.

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

Hasan, R. D., & Abandah, G. A. (2023). Correcting a Wide-Range of Arabic Spelling Mistakes Using Machine Learning and Transformers. In **2023 IEEE International Conference on Information Technology (ICIT),** pp. 405-410.

Hochreiter, S., and Schmidhuber, J. (1997), Long Short-term Memory. **Neural Computation,** 9(8), pp. 1735-1780.

Huang, Z., Xu, W., & Yu, K. (2015), Bidirectional LSTM-CRF Models for Sequence Tagging. **arXiv Preprint arXiv:***1508.01991*.

Irani, M., Elahimanesh, M. H., Ghafouri, A., & Bidgoli, B. M. (2022), A Supervised Deep Learning-based Approach for Bilingual Arabic and Persian Spell Correction. In **2022 IEEE 8th Iranian Conference on Signal Processing and Intelligent Systems (ICSPIS***)* pp. 1-7.

Laaroussi, S., Yousf, A., Aouragh, S. L., & Alaoui, S. O. E. (2023), Global Spelling Correction in Context using Language Models: Application to the Arabic

Language. **International Journal of Computing and Digital Systems,** *13*(1), pp. 361-370.

Lafferty, J., McCallum, A., & Pereira, F. (2001), Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In **Proceedings of the 18th International Conference on Machine Learning (ICML),** pp. 282-289.

McCarthy, J., Minsky, M. L., Rochester, N., & Shannon, C. E. (2006), A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence. August 31, 1955, **AI magazine,** *27*(4), pp. 12-12.

Mohit, B., Rozovskaya, A., Habash, N., Zaghouani, W., & Obeid, O. (2014), The First QALB Shared Task on Automatic Text Correction for Arabic. In **Proceedings of EMNLP Workshop on Arabic Natural Language Processing,** Doha, Qatar.

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

Necva, B. and Burcu, C. (2019), "Context Based Automatic Spelling Correction for Turkish". **Scientific Meeting on Electrical-Electronics & Biomedical Engineering and Computer Science (EBBT),** pp. 1-4, doi: 10.1109/EBBT.2019.8742067.

Post, M. (2021), Scientific Dissemination via Comic strip: A case study with SacreBLEU. In **Beyond static papers: Rethinking how we share scientific understanding in ML-ICLR workshop.**

Roberts, A., & Raffel, C. (2020), Exploring Transfer Learning with T5: The Text-to-Text Transfer Transformer, **google AI Blog**.

Shaalan, K., Aref, R., & Fahmy, A. (2010), An Approach for Analyzing and Correcting Spelling Errors for Non-native Arabic Learners. In **2010 The 7th international conference on informatics and systems (INFOS),** pp. 1-7.

Sharma, J. (2023), Understanding Attention Mechanism in Transformer Neural Networks. **LearnOpenCV**. https://learnopencv.com/attention-mechanism-in-transformer-neural-networks/ Last visited May 2023.

Sherstinsky, A. (2020), Fundamentals of Recurrent Neural Network (RNN) and Long Short-term Memory (LSTM) Network. **Physica D: Nonlinear Phenomena**, *404*, 132306.

Srinivasan, A. (2023), Introduction to Recurrent Neural Network. **GeeksforGeeks**. https://www.geeksforgeeks.org/introduction-to-recurrent-neural-network/ Last visited December 2022.

Trandafili, E., Paci, H., & Karaj, E. (2019). A Novel Document Summarization System for Albanian Language. In **Proceedings of the 20th International Conference on Computer Systems and Technologies** pp. 273-277.

Beysolow, T. (2018), Applied Natural Language Processing with Python. **Springer,** USA: San Francisco, California.

Phung, T. (2023), The Transformer Neural Network Architecture. **Tung M Phung's Blog**. https://tungmphung.com/the-transformer-neural-network-architecture/ Last visited April 2023.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., & Polosukhin, I. (2017), Attention is All You Need. **Advances in Neural Information processing Systems**, 30.

Wang, F., & Xie, Z. (2022), An Adversarial Multi-Task Learning Method for Chinese Text Correction with Semantic Detection. In **Artificial Neural Networks and Machine Learning–ICANN 2022: 31st International Conference on Artificial Neural Networks**, Bristol, UK, 2022, Proceedings, Part II pp. 159-173. Cham: Springer Nature Switzerland.

Xue, L., Barua, A., Constant, N., AlRfou, R., Narang, S., Kale, M., Roberts, A. Raffel, C. (2022). Byt5: Towards a Token-Free Future with Pre-Trained Byte-to-Byte Models. **Transactions of The Association for Computational Linguistics**, 10, pp. 291-306.

Zribi, C. (2023), “Easy” Meta-embedding for Detecting and Correcting Semantic Errors in Arabic Documents. **Multimedia Tools and Applications,** pp. 1-15.

تصحيح الأخطاء الإملائية السماعية وأسلوب المبالغة في اللهجة الأردنية باستخدام تقنيات التعلم الآلي

إعداد

ملاك أحمد يوسف الصمادي

المشرف

الأستاذ الدكتور غيث علي عبندة

ملخص

تختلف اللهجات من منطقة لأخرى، بعض الكتابات تحتوي على أخطاء إملائية وهو شيء شائع أيضًا في اللغة العربية. استغل الباحثون تقنيات متعددة لتصحيح هذه الأخطاء مثل تقنية نماذج اللغات، مقاييس الارتباك، والشبكات العصبونية. طورت مؤخراً تقنية وهي المحولة وقُدمت كنوع من الشبكة العصبونية ومنذ ذلك الوقت وهي تستعمل لحل العديد من المسائل في معالجة اللغات الطبيعية.

تظهر هذه الرسالة كيفية استخدام إحدى تقنيات المحولة لتصحيح نوعين من الأخطاء الإملائية وهما الأخطاء السماعية وأخطاء أسلوب المبالغة، وذلك باستخدام الأخطاء الإملائية المصطنعة للتدريب. صُنعت هذه الأخطاء بتقنية أسمها تلقيح الأخطاء العشوائي حيث قمنا بتطبيقها على مجموعة البيانات (wiki-40b/ar).

أما بالنسبة للنتائج، فقد أعطى نموذج تصليح الأخطاء السماعية أفضل نتائج عند تدريبه مع نسبة 75% لتلقيح الأخطاء، حيث وصلت الدقة الى 79.78% على مجموعة البيانات المراد بها تصديق النتائج (Validation Set) ومعدل خطأ الأحرف وصل الى 1.13% على مجموعة البيانات التي تم تجميعها من قبلنا (TestAud200). أما نموذج تصليح أخطاء أسلوب المبالغة، أعطى تلقيح الأخطاء بنسبة 10% أفضل نتيجة بدقة 99.78% على مجموعة البيانات المراد بها تصديق النتائج (Validation Set) ومعدل خطأ الأحرف هو 0.38% على مجموعة البيانات التي تم تجميعها من قبلنا (TestExg200).