TRANSLATING TEXT FROM THE JORDANIAN DIALECT TO MODERN STANDARD ARABIC USING LARGE LANGUAGE MODELS

By

Moath Rabeh Khaleel

Supervisor

Dr. Gheith Abandah, Prof.

This Thesis was Submitted in Partial Fulfillment of the Requirements for the Master's Degree in Artificial Intelligence and Robotics

School of Graduate Studies
The University of Jordan

August, 2024

COMMITTEE DECISION

This Thesis/Dissertation (Translating Text from The Jordanian Dialect To Modern Standard Arabic Using Large Language Models) 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 Abdel-Majeed, (Member) Assoc. Prof. of Computer Engineering	----------
Dr. Esam alQaralleh, (Member) Prof. of Computer Engineering Princess Sumaya University for Technology	----------

DEDICATION

I dedicate this thesis to my parents, Rabeh and Buthaina, to whom I owe every success in my life. If it weren't for their belief in me, investment in me, and their love and dedication, I wouldn't be the person I am today. Whatever I do, it will always be for them, and because of them.

To the rock in my life, my brother, Mohamad, who has always been a role model and an example. His resilience and character have always helped me get through hardships.

To my number one supporter, my grandfather, who has been a second father to me. His belief in me, his encouragement, and his presence in my life have always guided me.

To my cousin Zaid, whom I hope I will be able to help achieve the great things he is destined to achieve.

To my family and friends.

AKNOWLEDGEMENT

My gratitude goes to my supervisor, Professor Gheith Abandah, for supporting me through this thesis. Prof. Abandah's follow ups, prompt responses, and feedback have helped bring this thesis to life. I learnt a lot from Prof. Abandah, on both an academic and personal level. Observing Prof. Abandah's enthusiasm for science has been inspiring.

A thank you is also due to the faculty of the Computer Science department at the University of Jordan. Throughout the past two years, they did not spare a chance to teach us every way they could.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

BiLSTM	Bidirectional Long-Short Term Memory
BLEU	Bi-Lingual Evaluation Understudy
ByT5	Byte-to-Byte T5
C4	Colossal Clean Crawled Corpus
CER	Character Error Rate
CNN	Convolutional Neural Network
CPU	Central Processing Unit
FFNN	Feed-Forward Neural Network
GCP	Google Cloud Platform
LSTM	Long-Short Term Memory
MSA	Modern Standard Arabic
mT5	Multilingual T5
NLP	Natural Language Processing
RBMT	Rule Based Machine Translation
RNN	Recurrent Neural Network
SMT	Statistical Machine Translation
T5	Text-to-Text Transfer Transformer
TPU	Tensor Processing Unit
TSMTS	Tashkeela Spelling Mistakes Test Set

TRANSLATING TEXT FROM THE JORDANIAN DIALECT TO MODERN STANDARD ARABIC USING LARGE LANGUAGE MODELS

By
Moath Rabeh Khaleel

Supervisor
Dr. Gheith Abandah, Prof.

ABSTRACT

Translating texts from the Jordanian dialect to Modern Standard Arabic (MSA) requires a significant amount of manual effort. Rule-based systems are extremely challenging to develop, especially with a language like Arabic, which is rich in grammar and morphology. Large language models offer practical solutions for this task. Byte-to-byte models, particularly the ByT5 model, emerge as a promising solution for automating the translation of Jordanian dialect texts to MSA.

This thesis aims to address the challenges faced in this low-resource field. We use the JODA dataset for the first time for this problem, experiment with fine-tuning a baseline model, and attempt to find well-tuned hyperparameters that perform efficiently on the dataset. We also explore using additional datasets in training to tackle correcting general and directed spelling mistakes. We leverage the model's byte-level processing and auto-tokenization features, along with synthetic data generation, to overcome the scarcity of training data.

The study begins with training a baseline model on the JODA dataset. We then adjust various hyperparameters in pursuit of achieving state-of-the-art BLEU scores on the JODA test set. Subsequently, we experiment with training the model on additional datasets, including the Clean-50 and Clean-400 datasets, to extend the model's performance to be able to correct direct and general spelling errors.

The ByT5 model variants, small and base, along with the optimizers Adam and Adafactor, are investigated to enhance the model's performance. Results show that the base

model with the Adam optimizer performs best for translating the Jordanian dialect to MSA. The proposed solution achieves a BLEU score of 57.77 on the JODA dataset. Additionally, we present a model that performs slightly less on the JODA test set with a BLEU score of 57.39 but achieves competitive scores on the Test200 and TSMTS test sets at character error rates of 4.64% and 1.65%, respectively, demonstrating the model's ability to correct direct and general spelling mistakes.

Chapter One

Introduction

Arabic is the official language of 22 sovereign states stretching from Mauritania in the West to Iraq in the East. Estimates suggest a figure of around 274 million Arabic speakers worldwide (Ethnologue, 2023). Arabic is also spoken in the neighboring parts of Southern Turkey, Afghanistan, Iran, and Uzbekistan. In addition, political and economic conditions in the Arab region caused various migration movements which led to large Arabic speaking communities in the United States, Europe, and South America.

The evolution of societies all over the world led to the emergence of different languages and dialects that differ because of geographical location. This is applicable in both Arab and Western countries alike. Arabic dialects have evolved because of two kinds of movements: a gradual movement in terms of lifestyle, resulting in a shift from a historically tribal society to a settled society; and a population movement within and outside the Arabian Peninsula (Watson *et al*., 2011). People from different Arab regions used to, and continue to be, brought together by trade caravans, religious pilgrimages, and migratory work. This movement contributed to the emergence of different Arabic dialects.

This thesis aims to bridge the gap between the different dialects spoken by Arabic speakers, by providing a mean for translating local dialects to Modern Standard Arabic (MSA).

1.1 Arabic Dialects

The vernacular languages of Arabic countries are informal dialects that differ in varying degrees from Modern Standard Arabic (MSA). Weakness in speaking and writing in MSA among Arabic speakers leads to abandoning it and resorting to informal dialects, or in some cases, foreign languages. This issue increased with the wide spread of social media, which led to widespread communication between different groups of society, with wide variation in their written and spoken Arabic proficiency.

There is an urgent need for practical solutions capable of processing informal Arabic sentences (Al-Ibrahim *et al*., 2020). Such solutions can bridge the gap between speakers of different dialects. Solutions that rely on traditional methods require a complex set of rules due to the great diversity in vernacular dialects at the level of one country, l*et al*one, at the level of the entire Arab region. There is a need to utilize technological advancements to translate different Arabic dialects, facilitating their understanding and handling by people from other communities.

Modern methods for natural languages processing using deep machine learning techniques emerge as strong candidates for translating informal Arabic dialects to MSA. Deep machine learning techniques have proven to be efficient in addressing Arabic processing tasks, such as correcting Arabic spelling mistakes (Abandah *et al*., 2022a) and diacritizing Arabic poetry (Abandah *et al*., 2022b).

1.2 Thesis Problem

The vernacular languages of Arab countries are informal Arabic dialects that differ from MSA in varying degrees. Weakness in the formal Arabic language leads to writing weak Arabic texts and resorting to foreign languages. This problem increased with the wide-spread of social media and wide-spread communication between groups of people with varying levels of proficiency in written and spoken Arabic. This raises the need for a solution to bridge the gap.

The research problem is encapsulated in finding a way to automate the process of translating local Jordanian dialect to MSA. Traditional methods require a comprehensive and complicated set of rules that can only be applied to specific pre-computed conditions. Once the text deviates from the pre-computed conditions, traditional methods fail, hence the need for an autonomous method rises. This thesis investigates using large language models to translate texts in the Jordanian dialect to MSA.

1.3 Thesis Importance

Building a solution using deep neural networks will impact the level of Arabic language used in different communication platforms. The model can facilitate integrating the solution with automated response systems (Chatbots). The proposed solution can be used in various customer service, governmental, and economic sectors. Furthermore, this research can pave the way for further implementation with other Arabic dialects and can be expanded on a regional level.

1.4 Thesis Objective

The research objectives of this thesis are summarized as follows:

- Provide a comprehensive overview of the different techniques and architectures previously employed for sequence-to-sequence natural language processing.
- Employ transfer learning along with large language models to translate Arabic text from the Jordanian dialect to MSA.
- Examine the performance of a token-free model such as ByT5 (Xue et al., 2022) on a limited resource language like the Jordanian dialect.
- Explore enhancing the dataset by using synthetic samples in hopes of developing a model able to correct general and directed errors.
- Determine the effect of dataset quality, scaling, and fine-tuning techniques on the quality of results obtained.

1.5 Thesis Questions and Assumptions

The main questions of this research are:

- Can transfer learning be used to fine tune a pre-trained model on a downstream task like translating Jordanian dialect to MSA?
- Can a token free model like ByT5 (Xue *et al*., 2022) perform well with a low resource language like the Jordanian dialect?
- What effect does the dataset quality have on the performance of the model?
- Does model size matter in translating Jordanian dialect to MSA?
- Can training the model on additional datasets help in developing a model able to correct general and directed errors?

The main assumptions of this research are:

- There will be a complete dataset containing sentences in the Jordanian dialect and their counterparts in MSA from a previous funded project led by Prof. Gheith Abandah.
- The model to be used is an open source one and can be readily used without further licensing.
- The topic of translating Jordanian dialect to MSA is one largely unexplored.
- The ByT5 model is better at translating the Jordanian dialect than the T5 model as it operates at a byte level. This eliminates the need for words to be part of a fixed vocabulary, and hence can be more robust with a low-resource scarce language like the Jordanian dialect.

1.6 Thesis Contribution

As far as we are aware, we present the first of its kind research that focuses on translating the Jordanian dialect to MSA using the ByT5 model and working at the byte level in this manuscript. Our research is different as it uses byte-level processing to capture and translate a low resource dialect like the Jordanian dialect.

We aim to pave the way for other research in the field, in particular one that focuses on translating local Arabic dialects to MSA. This will push the boundaries of linguistics for the Arabic language and will bridge the gap between local communities speaking different dialects.

1.7 Thesis Methodology

The research methodology of this thesis is summarized as follows:

- Conduct a thorough literature review on the modern natural language processing techniques used.
- Acquire and prepare a complete dataset with sentences in both the Jordanian dialect and Modern Standard Arabic.
- Employ transfer learning along with large language models to translate the Jordanian dialect to MSA.
- Assess the performance of a token-free model like ByT5 when finetuned for the task in hand.
- Examine the different factors influencing the performance of the model, such as: dataset quality and size, batch size, learning rate, optimizer used, and model size.
- Assess training the model on a complimentary dataset in hopes of obtaining a model able to correct general and directed errors in Arabic.
- Explore pathways for future research.

1.8 Thesis Outline

The remaining part of this thesis is organized in the following manner:

- Chapter 2 reviews previous research on the thesis topic and natural language processing techniques used.
- Chapter 3 provides a comprehensive explanation of the proposed model and explores the details for several key concepts in machine learning.
- Chapter 4 covers dataset preparation and proposed baseline model.
- Chapter 5 reports the experiments conducted to find the best performing model for our thesis objective.
- Chapter 6 outlines a summary of the conclusions and paves the way for future work.

Chapter Two

Literature Review

In this chapter we survey past work that focused on translation tasks using non-machine learning techniques, work that utilized modern natural languages processing (NLP) techniques for translation, and work that focused on translating different Arabic dialects. In addition, we survey past work that explored correcting spelling mistakes, as we will use similar approaches to generate complimentary synthetic datasets.

2.1 Translation Using Traditional Machine Translation Approaches

Prior to the evolution of deep neural networks, traditional machine translation techniques were used for translation tasks. Rule-based machine translation (RBMT) is one of the traditional methods used for machine translation. RBMT requires morphological, semantic, and syntactic rules about the source and target languages.

RBMT systems work based on the specification of rules for morphology, syntax, and lexical selection. RBMT is divided into transfer method and direct method. Transfer method analyzes of the syntactic structure of source languages and transfers the learnt abstract representation of sentences to target sentences. Direct methods apply word-by-word translation directly. In a paper published by Rajan *et al*. (2009), RBMT was used to translate English to Malayalam.

Statistical Machine Translation (SMT) is a machine translation approach that uses statistical methods for translation. There are three models in SMT, phrase based, syntax based, and hierarchical phrase-based system. SMT techniques use parameters derived from parallel corpus analysis along with probability, to determine the correct translation. In a paper published by Sreelekha *et al*. (2017), SMT was used in translation tasks on the Indian language.

2.2 Translation Using Neural Networks

Convolutional Neural Networks (CNNs) were used for machine translation tasks. In a paper published by Gehring *et al*., (2016) the authors used a fully convolutional model for sentence to sequence learning in a large dataset. The approach proved efficient, and authors were able to compare their results to those reported using recurrent networks. The authors worked on a translation task from English to Romanian.

In the past couple of decades, we have witnessed a great amount of progress in the machine translation field. Deep learning in particular has caught a lot of attention as it enables the automatic extraction of features without the need to manually extract them. Deep learning has been employed in several fields and machine translation is no different (Singh *et al*., 2017).

While encoder-decoder architectures are common in translation tasks, working with low-resource languages might not give the best results. Weng *et al*. (2017) suggested using a word predictor mechanism. The model uses words in a sentence where they can be considered a natural annotation to reinforce initial states generated by the encoder and control hidden states of the decoder. The authors were able to use this model to translate German to English and Chinese to English. The experiments reported by the authors showed promising and good results.

Wang *et al*. (2019) used a deep transformer model along with normalization to complete a translation task. The authors proposed an approach based on a dynamic linear combination of layers and successfully trained a 30-layer transformer system. The thin-but-deep encoder was able to achieve remarkable scores on the WMT'16 English-German, NIST OpenMT'12 Chinese-English, and larger WMT'18 Chinese-English tasks. The model proposed by the authors is also smaller in size and faster in computation time than its counterpart classical transformer model.

In a paper published by Sutskever *et al*. (2020), deep neural networks were employed to translate sentences from English to French. The authors used a multi-layered Long Short-Term Memory (LSTM) architecture to convert input sequences (in English) to a higher dimensionality vector and then used another deep LSTM to decode the target sequences (in French). The results reported by the authors show an advancement from the previous results reported using SMT.

2.3 Translation of Jordanian Dialect to Modern Standard Arabic

Translating Jordanian dialect to Modern Standard Arabic is a topic largely unexplored. In a paper published by Al-Ibrahim *et al*., (2020) a recurrent neural network encoder-decoder model was used to perform the translation task. LSTM units were used in the model's layers. The authors used two datasets in their training. The first dataset consisted from 24,200 words collected from different Jordanian dialects along with their counterparts in MSA. The second dataset contained 500 sentences collected from different Jordanian dialects along with their counterparts in MSA.

The authors applied different filtering techniques and trained the model to produce the probability of a Jordanian word/phrase corresponding to a word/phrase in MSA. The authors were able to achieve an accuracy of 91.3% and a loss value of 0.8 on a word-to-word level, and an accuracy of 63.2% level and a loss value of 3.33 on a sentence-to-sentence.

2.4 Correction of Arabic Spelling Mistakes

Arabic spelling correction is a field that has been gaining momentum in recent years. In a paper published by Mubarak *et al*. (2014), the authors introduced a character-level correction model to correct Arabic spelling mistakes. The language model focused on errors difficult for detection or correction. The model functioned by searching the lexicon for similar words to those that contain errors. The authors were able to achieve an F1 score of 63.43% on the QALB dataset.

In a paper published by AlShenaifi *et al*. (2015), the authors introduced a system called Arib that uses rule-based corrector to insert space after the prepositions (على ، الى) and letters (ـة ، ة) when merged with the next word, converts English punctuation to Arabic punctuation, and removes letters that occur sequentially three times or more in the same word. The authors also used the Bayes probability algorithm along with a large dictionary constructed from various sets to correct errors that span across multiple words. The authors were able to achieve an F1 score of 57.8% on the QALB dataset.

Attia *et al*. (2016) introduced a system that detects and corrects Arabic spelling mistakes. The spelling mistakes were detected if found in a list containing the counterpart correct words, or were detected using two-character level language models. The two models consisted of one trained on correct words and another trained misspelled words. The model was able to achieve a remarkable accuracy of 93.64%.

In a paper published by Abdellah *et al*. (2020), a system was introduced that is capable of correcting space omitted mistakes in Arabic text based on the Levenshtein distance. The system inserts a space character between two words merged together due to a space character being wrongfully omitted. The system was tested with a test set comprising from 1 thousand words and was able to achieve a correction rate of 99.14%.

In a paper published by Abandah *et al*. (2022), authors focused on correcting stochastically injected soft spelling mistakes. The authors trained a Bi-directional long-short term memory (BiLSTM) model to correct artificial errors injected by randomly replacing target letters pertaining to soft Arabic spelling mistakes. The authors experimented with different error injection rates, where a letter belonging to one of the four groups shown in Table 1 was randomly replaced by one of the letters in its group. The best model proposed by the authors was able to correct 96.4% of the errors injected and was able to achieve a low character rate of 1.28% on a real test set containing soft spelling mistakes.

Table 1. Letters replaced through stochastic injection (Abandah *et al*., 2022)

Intermediate Forms	Mapped To
[ا ، ئ ، إ ، ؤ ، آ ، أ ، ء and J (اء)]	ء
[ة , T (ت) , H (ه)]	ة
[O (و) and A (وا)]	O (و)
[ا and ى]	ى

2.5 Character Level Models

Pretrained character-level and byte-level language models have shown competitive performance across a range of Natural Language Processing (NLP) tasks. In a paper published by Jentoft (2023), the author utilized the ByT5 model for grammatical error correction tasks in Norwegian and English. For Norwegian, the author utilized the ASK dataset consisting from 50,000 sentences written by non-native language learners with two different levels of Norwegian fluency. For English, the author utilized the FCE dataset, containing responses written by candidates who took the Cambridge ESOL First Certificate English Exam. The model achieved an F0.5 score of 0.51 for English and 0.581 for Norwegian.

Kirov *et al*. (2024) utilized the ByT5 byte-level sequence-to-sequence model to transliterate full sentences from informal romanization prevalent in South Asia to their native

scripts. The study covered 12 languages within the Dakshina dataset. The authors reported a remarkable 3.3% absolute mean word-error rate reduction corresponding to 18.6% relative rate.

In a paper published by Edman *et al*. (2024), the authors explore the effectiveness of character-level models when used in translation tasks. The authors focus their study on the ByT5 and mT5 models in particular. The authors chose to conduct multiple translation tasks. In particular, they fine-tuned the models to translate from German and Russian to English and from Portuguese and English to Spanish. The languages were picked to account for varying degrees of language similarity. The authors considered several translation quality metrics and eventually opted for chrF++, which is a combination of character level and word level F-scores.

The authors conclude that character level models can produce competitive, or even superior in some cases, results when compared to subword-level models. The authors prove that character models are particularly efficient when fine-tuning is done with scarce data. Character models can implicitly account for the appropriate input granularity given the context, resulting in better accuracy when translating orthographically similar and rare words. The authors also report an important tradeoff in computation speed, as they found that character models are at least 4 times slower in training and inference, making them suboptimal for many real-world situations.

Chapter Three

Machine Transcription Models

In chapter Three, we outline the evolution of natural language processing models and techniques leading to the ByT5 model. In the first stage of this study, we experiment with different hyperparameters to determine the best performing ones in translating Jordanian dialect to MSA using the JODA dataset. We also explore training our model on synthetic datasets that target general and directed errors, in hopes of finding a model that can extend its performance beyond the JODA dataset in hand.

3.1 Introduction

In our experiments, we use the ByT5 model, a byte level model built using the transformer architecture. To understand the model and its advantage over other models, we must understand the advancements in the natural language processing field that lead to the ByT5 model. This chapter focuses on the evolution of deep neural networks, including RNN and LSTM networks, as well as the transformer architecture.

3.2 Recurrent Neural Networks

Recurrent neural networks (RNNs) are a form of networks that perform time-dependent processing using feedback from previous time steps. RNNs are also known as memory cells, they have an ability to capture sequential patterns and temporal dependencies. RNNs gained popularity in the NLP field due to their ability to capture context and dependencies between input tokens by retaining context from previous inputs (Rumelhart *et al*., 1986).

Figure 1 shows a representation of what a typical RNN cell looks like. As shown in the figure, input at each time step is used along with context from the previous time step to make a prediction for the current time step.

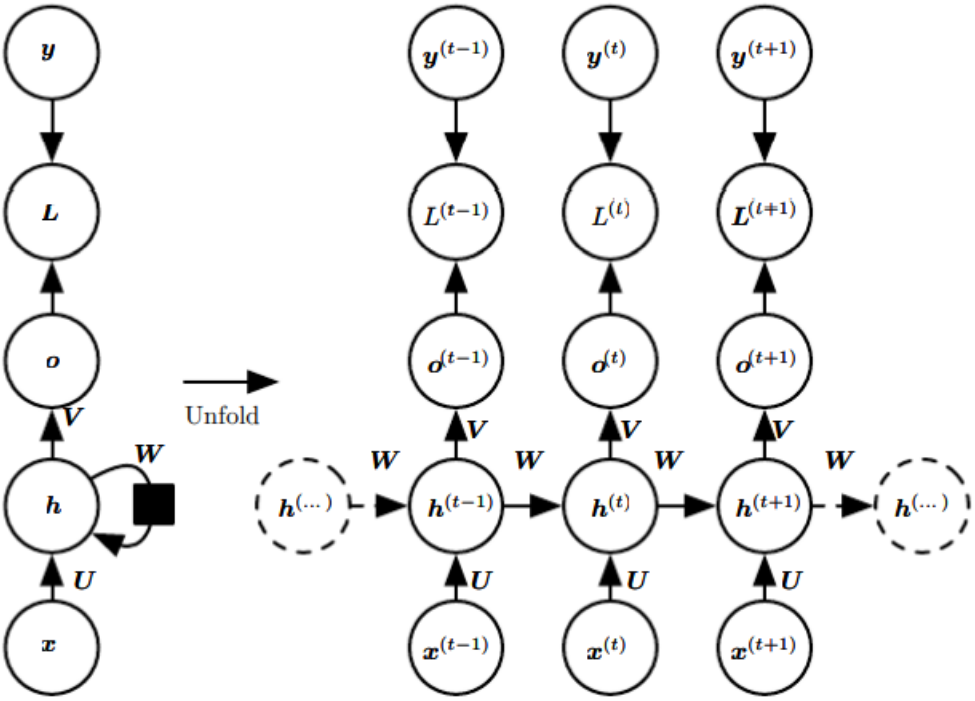

Figure 1. Visual representation of a RNN cell (Analytics, 2021)

The following equations are used to represent the system:

$$a_t = b + Wh_{t-1} + Ux_t \quad (1)$$

$$h_t = \tanh(a_t) \quad (2)$$

$$o_t = c + Vh_t \quad (3)$$

$$y_t = softmax(o_t) \quad (4)$$

RNNs map an input sequence, x_t, to a corresponding output sequence, o_t. A loss, L, measures the difference between the actual output y_t and the predicted output o_t. The variables b and c are the bias vectors. U, V, and W, are weight matrices which are updated using the gradient of the loss function with respect to each weight matrix. The hidden state for each time step is computed as shown in equation 2.

While recurrent neural networks offer many advantages, they suffer from certain challenges. RNNs tend to exhibit diminishing performance with longer sequences, as RNNs suffer to capture long-term dependencies between inputs. By the time RNNs receive later parts

of a sequence, they tend to forget earlier parts of the same sequence. RNNs also suffer from vanishing and exploding gradients. To address those fallbacks, the LSTM architecture was introduced.

The LSTM architecture is a form of RNNs that addresses the vanishing gradient problem. LSTMs use memory cells to capture hidden states from earlier stages, enabling the model to process and analyze long-range sequences (Sherstinsky, 2020).

While LSTMs are successful in capturing long-term dependencies based on past context, the BiLSTM architecture was introduced, allowing for predictions to be made based on past and future context, allowing for better capturing of temporal patterns in sequential data. The BiLSTM architecture processes sequences in both directions simultaneously; from the past to the future and from the future to the past. Figure 2 shows a visual representation of what the BiLSTM architecture looks like (Li *et al*., 2020).

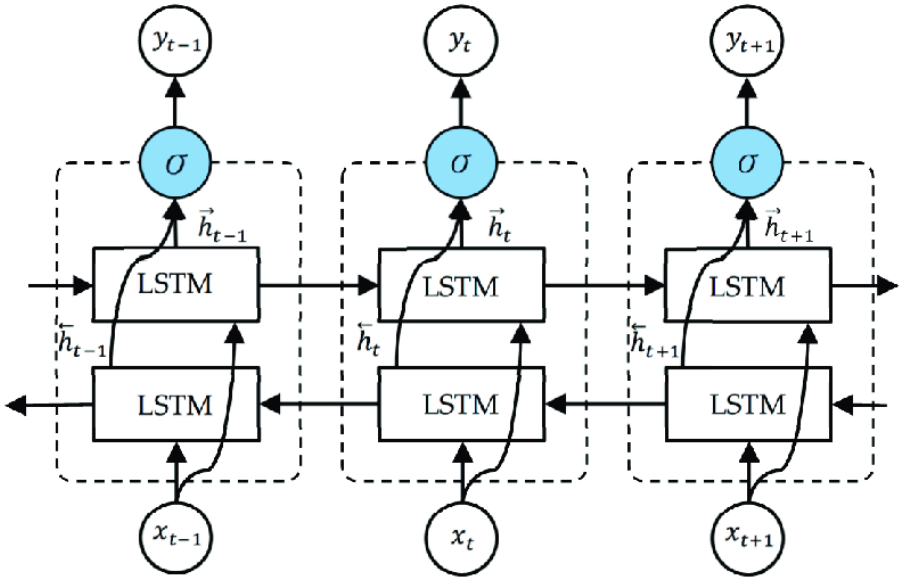

Figure 2. BiLSTM Architecture (Li *et al.*, 2020)

Deep RNNs are formed by arranging multiple RNN hidden layers one after the other, with the input sequence for each layer being the output from the previous layer. The hidden vectors are generated repeatedly for all layers from $n = 1$ to $n = N$ and from $t = 1$ to $t = T$ time step, assuming a constant hidden layer function across all N levels in the stack.

Applications that depended on LSTM and RNN as foundational models suffered from lengthy training periods and huge computational demand. This led to the evolution of the transformer, which utilizes self-attention to eliminate fallbacks found in RNNs in general.

Recurrent models perform computation along the positions of the input and output sequences, where positions are aligned to steps and sequences of hidden states are generated. The sequential nature of recurrent models becomes critical at longer sequence lengths, as batching is limited by memory constraints. Reducing sequential computation led to the emergence of convolutional neural networks that enable parallel computation of hidden representations for all input and output positions. Nevertheless, the number of operations required to relate signals from two arbitrary positions grew drastically, which made it difficult to learn dependencies between distant positions (Gehring *et al*., 2017).

3.3 Transformers

The shortcomings of RNNs lead to the development of transformers which leverage self-attention mechanisms to process and understand text. Introduced by Vaswani *et al*. (2017), transformers eliminate the need for recurrent or convolutional layers by relying on self-attention to capture relationships between words in sentences, regardless of the distance between words. This approach enables parallel processing and improves the model's ability to handle long-range dependencies, making transformers highly effective for a wide range of NLP tasks.

3.3.1 Transformer Architecture

The introduction of the attention mechanisms allowed modeling of dependencies without regard to the distance between them. In most cases, attention mechanisms were used in conjunction with a recurrent network. In the "Attention Is All You Need" paper (Vaswani *et al*., 2017), the Transformer was proposed, where recurrence was left out altogether and instead

self-attention mechanisms were relied on to draw dependencies between the input and output sequences. The Transformer is the first transduction model relying on self-attention without using sequence aligned Recurrent Neural Networks or convolution.

The Transformer consists of an encoder-decoder structure. The encoder takes in an input sequence, maps it to a higher dimensionality representation, and a decoder is then used to generate the output sequence one element at a time. The model is auto regressive, taking in previously generated symbols as extra input while generating the next element. The transformer uses positional encoding, stacked self-attention layers, point-wise fully connected layers, and multi-head layers running in parallel. The Transformer, along with self-attention, reduces computational complexity per layer, enable parallelized computation, and allow learning long-distance dependencies (Khan *et al.,* 2022).

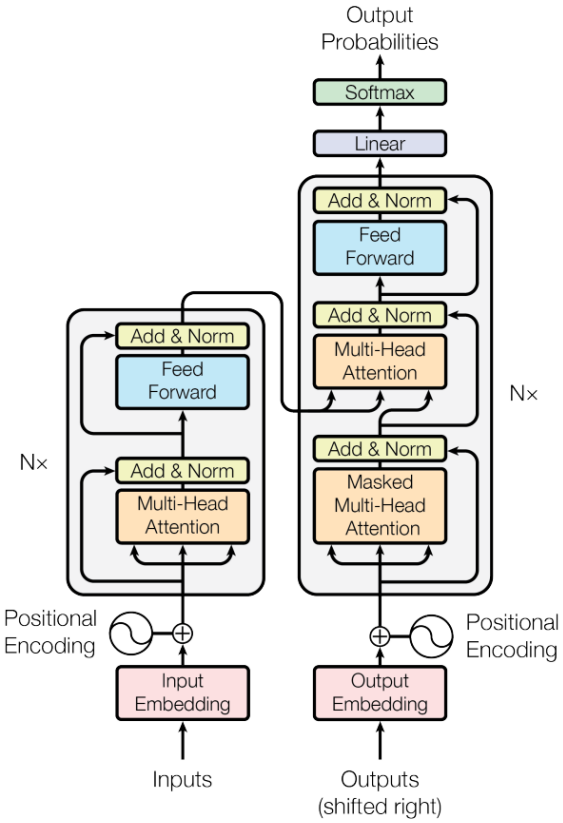

Figure 3. Transformers Architecture (Vaswani *et al.*, 2017)

As shown in figure 3, the transformer architecture consists of several key components; embeddings, the encoder and the decoder, multi-head attention, and feed-forward neural network (FNN).

3.3.2 Word Embeddings

In natural language processing, texts are not dealt with in their raw formats. Text must first be converted to vectors of numbers in order for a model to process that piece of text. Word embedding is used in NLP to convert pieces of text to dense vectors of real numbers in a higher dimensionality space.

In this higher dimensionality space, words that are similar form vectors that are closer together, while words that are not similar form vectors further apart from each other. This allows the model to understand and build relationships between sequences of text.

While dedicated models have been pre-trained to generate word embeddings, the transformer employs an embedded word layer integrated into the model's architecture. When the model ingests an input sequence, whether through the encoder or through the decoder, the transformer converts the sequence to a vector as it goes through the embedding layer.

3.3.3 Encoder and Decoder

The encoder and decoder are two main building blocks in the transformer architecture. The encoder is responsible for ingesting the input sequence and transforming it to a higher dimension vector that can be sent over to the decoder. The encoder receives the sequence after it gets processed through embedding and positional encoding.

The original transformer first released by Vaswani *et al*. (2017) consisted from 6 identical encoders stacked on top of each other. Each encoder layer consists of a multi-head self-attention component and a fully connected FNN component. The positional encoded and embedded input sequence undergoes a process called "residual connection", which allows for

summation and normalization of the input sequence, and helps in preserving information throughout the network (He *et al*., 2016).

Similar to the encoder, the original transformer consisted from 6 identical decoders stacked on top of each other. The decoder's setup is similar to the encoder, besides the addition of a masked multi-head attention block that attends over the output of the decoder itself. Masking is used to prevent the decoder from seeing future outputs.

3.3.4 Attention Mechanism

The attention mechanism is the key component that allows the model to understand how different pieces of the input sequence relate to each other. This kind of knowledge allows the transformer to output coherent sequences while preserving inner-relationships between sub-sequences. This is crucial in tasks that involve text understanding, translation, and text generation.

Assuming Q, K, V, d_k, and d_v are the matrices of queries, keys, values, a scaling dimension of the key, and a scaling dimension of the value respectively. The attention is computed using:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \qquad (5)$$

Multi-head attention incorporates self-attention. To consider different features of the input data, multi-head attention parallelly computes multiple transformations of the query, key, and value matrices using trainable weights. The scores for individual heads are computed, concatenated, and passed through a final layer to produce output vectors that are passed to the position-wise FNN.

3.3.5 Feed Forward Network

The position-wise FNN is a neural network consisting of a two-layer multi-layer perceptron. The input is multiplied by a weighted matrix W_1, a bias term is added, the output is then multiplied by a second weight matrix W_2, and a second bias term is added.

$$FFN(x) = \max(0, (\ input\ \times\ W_1) +\ b_1)W_2 + b_2 \qquad (6)$$

3.4 Text-to-Text Transformer Model

The Transformer architecture became the primary building block for a lot of research in the NLP domain. In a paper published by Raffel *et al*., (2020) the Transformer architecture was used as a bedrock for introducing a unified framework that converts all text-based language problems into a text-to-text format. In their systematic study, the authors compare pre-training objectives, transfer approaches, architecture scaling, and several other factors, on multiple language understanding tasks.

Along the way, authors of the text-to-text transformer model (T5) framework introduce the "Colossal Clean Crawled Corpus" C4 dataset, consisting of hundreds of gigabytes of filtered and clean English text crawled from the web. The dataset is used to develop general-purpose knowledge that allows the model to understand text. Within their systematic study, the authors were able to push the performance on popular NLP benchmarks (Liu *et al*., 2023).

3.4.1 Text-to-Text Transformer Model Architecture

The T5 architecture is like the original transformer's architecture with some key enhancements. For starters, the T5 architecture incorporates normalization layers to stabilize training and improve convergence across different layers. In addition, the T5 architecture adds positional embeddings into the input embeddings, which allows for encoding word positions within the embeddings. This helps in enhancing the model's contextual comprehension and understanding of text structure.

The number of encoders and decoders used in the T5 model is similar to the BERT-base model with both the encoder and decoder consisting of 12 blocks each (Devlin *et al*., 2018). Similar to the original transformer, each building block consists of self-attention, a feed-forward network, and a multi-head masked attention for the decoder.

The output of the dense layer has 3072 dimensions (Raffel *et al*., 2020), followed by a ReLU activation layer and another dense layer. The attention mechanisms are composed from 64 and 12 heads. Output embeddings from all other sub-layers are set to 768 dimensions. This brings up the total number of parameters for the model to around 220 million parameters. Dropout is applied throughout the model with a probability of 0.1.

3.4.2 Colossal Clean Crawled Corpus

Along the way, Liu *et al*. (2023), original authors of the T5 model introduce the C4 dataset, consisting of around 750 gigabytes of filtered and clean English text crawled from the web. The dataset is used to develop general-purpose knowledge that allows the model to understand text. Within their systematic study, the authors were able to use the dataset to train the model and push the performance on popular NLP benchmarks. The corpus is publicly available through TensorFlow Datasets.

3.4.3 Text-to-Text Transformer Model Evaluation Benchmarks

Raffel *et al*. (2020) introduced the T5 model as a text-to-text framework able to handle multiple text processing tasks. The model is designed to be trained with a consistent objective, namely teacher-forced maximum likelihood, and can handle various tasks out of the box, including text classification, question answering, translation, and text summarization. The T5 model was able to achieve exceptional scores on benchmarks such as SuperGlue and GLUE for text categorization, SquAD for question answering, and CNN/Daily Mail for text summarization.

While the T5 model is able to handle tasks in English, as well as other tasks in French, German, and Romanian, the model is not very capable of handling tasks in other languages. For this reason, a variant of the T5 model was introduced called the mT5 model.

3.5 Multilingual Text-to-Text Transformer Model

The T5 paper introduced a unified text-to-text framework that achieved state-of-the-art results on a wide range of English related NLP tasks. In a paper published by Xue *et al*. (2021), the multilingual text-to-text transformer (mT5) was introduced as a multilingual variant of T5 trained on a new dataset covering 101 languages. mT5 inherits all the benefits of T5, including general-purpose text-to-text format, empirical study design, and scale.

3.5.1 Model Overview

The model is trained on multiple languages, ranging from low-resource language to high-resource languages, which requires careful sampling to avoid underfitting and overfitting (Xue *et al*., 2021). The approach followed by the authors was to boost lower-resource languages by conducting sampling based on the probability $P(L) \propto |L|^{a}$, where P(L) represents the probability of selecting text from a specific language during pre-training and L denotes the number of samples in that language. The hyperparameter, a, manages the extent of boosting on low-resource languages. Authors of the paper found the optimal value to be 0.3.

3.5.2 Multilingual Colossal Clean Crawled Corpus

Along the way, authors of the mT5 framework introduce a multilingual variant of the C4 dataset called the multilingual colossal clean crawled corpus (mC4). The dataset is crawled from more than 100 languages. The dataset is significantly larger than the C4 dataset, amounting to 6.6 billion pages in total. In a paper where the mT5 and mC4 were introduced, authors tackle cross-lingual zero-shot transfer (models fine-tuned on English data only), translate-train (models fine-tuned on English data plus translations in all target languages), and in-language multitask (models fine-tuned on gold data in all target languages). The authors

were able to achieve state-of-the-art performance on many multilingual benchmarks by pre-training the model using the mC4 dataset (Liu *et al*., 2023).

3.6 Token-Based to Token-Free Models

An important consideration when designing NLP models is the way text is represented. The previously introduced literature operates on sequences of tokens corresponding to word or sub-word units. Pieces of text are "tokenized" and fed into the model for processing. This requires using a fixed vocabulary of words. If a piece of text contains an out-of-vocabulary word, there is no obvious way to process that piece of text.

3.7 Byte-to-Byte Text-to-Text Transformer Model

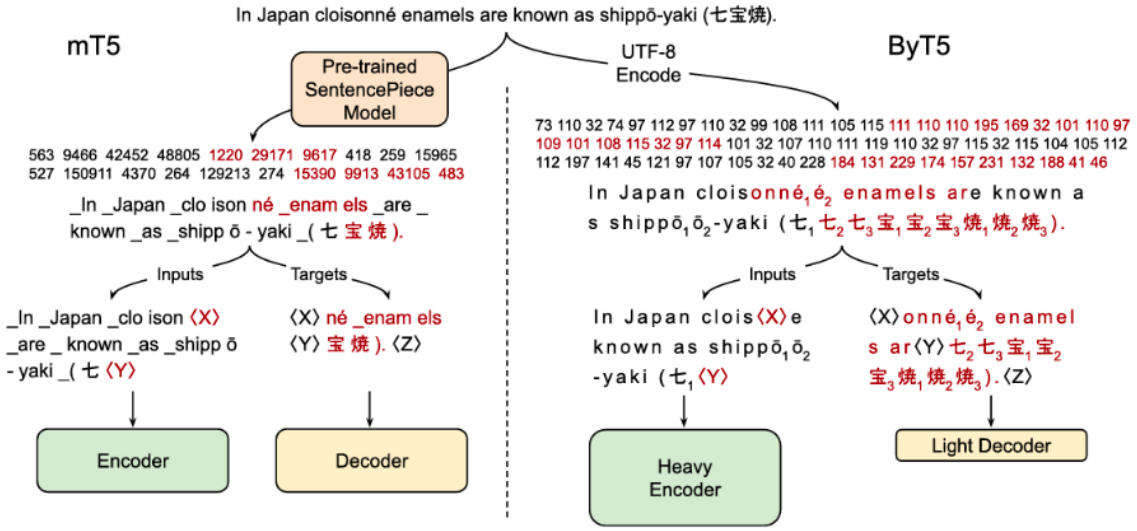

Figure 4. Comparison between the mT5 model and ByT5 model (Xue *et al*., 2022)

In a paper published by Xue *et al*. (2022), the Byte-to-Byte text-to-text transformer (ByT5) model was introduced. The model is based off of the mT5 model, with the key distinction of operating on raw text (bytes) directly without any text pre-processing.

3.7.1 Model Overview

By operating on bytes directly, the ByT5 model becomes more robust to noise, allows processing any language out of the box, and reduces the number of parameters allocated outside of the encoder-decoder architecture. To put matters in perspective, dispensing the words vocabulary frees up about 66% of the total parameters count used in the mt5-Base model, allowing allocating those parameters elsewhere in the model. ByT5 was proven competitive to parameter-matched mT5 models in some downstream tasks, while outperforming mT5 in other downstream tasks (Liu *et al*., 2023).

3.7.2 Byte-to-Byte Text-to-Text Transformer Model Architecture

The ByT5 model's architecture is similar in nature to that of the T5 and mT5 models except for a few key differences:

- Each character in the input text is converted directly into its corresponding byte representation (typically using UTF-8 encoding). The ByT5 model has a fixed vocabulary of 256 tokens, corresponding to the 256 possible byte values (0-255). This vocabulary covers all possible byte values, making it language-agnostic and capable of handling any text, regardless of the language or script. Each byte token is passed through an embedding layer, which maps the byte value to a high-dimensional vector. This is similar to how other models map word pieces or subwords to vectors, but in ByT5, it's done at the byte level. The model learns the embeddings for these byte tokens during training, just as it would for any other token type in a conventional model.
- Pre-training technique for the ByT5 model is slightly different than the mT5 model. The ByT5 model uses a mean mask span length of 20 bytes while the mT5 model uses an average span length of 3 sub-word tokens.

- The ByT5 model complies with UTF-8 standards. If an illegal byte (out of scope) is part of the input, the byte is excluded from the model's output.
- The ByT5 model's architecture uses a more in-depth encoder than a decoder. This is like the BERT model.

While exhibiting the above differences, the ByT5 model preserves a sequences length of 1024 (Bytes in this case) and pre-training is done for 1 million steps over batches consisting from 220 tokens.

Instead of tokenizing text into words or subwords, ByT5 tokenizes text into bytes. This byte-level approach allows the model to handle text in any language or script without needing specific tokenization schemes for different languages. The model uses an embedding layer that maps byte values (ranging from 0 to 255) to continuous-valued vectors. This layer transforms each byte into a dense representation.

Just like other transformer models, ByT5 uses positional encodings to provide information about the position of each token in the sequence. Since ByT5 operates on bytes rather than words or subwords, positional encodings are crucial for the model to understand the order of bytes in the text.

3.7.3 Byte-to-Byte Text-to-Text Transformer Model with Arabic Tasks

ByT5 has been leveraged to perform downstream tasks in Arabic. In a paper published by Al-Rfooh *et al*. (2023), ByT5 was fine-tuned to predict and insert missing diacritics in Arabic text. Diacritics are essential to disambiguate the meaning of an Arabic text. Arabic diacritization is a complex task that requires understanding the sentence semantics and morphological structure of tokens. Authors were able to adopt the ByT5 as a foundational model and achieve state-of-the-art results including a reduction in Word Error Rate by 40%.

3.8 Transfer Learning

Transfer learning is a technique used to leverage a model trained on one task in solving a different, yet related, task. As the model is pre-trained on an initial task, the model's weights get updated. This allows for conducting downstream tasks with a model having updated weights instead of using one with random weights. Pre-trained models, such as the T5 and ByT5 models, are trained on large datasets for a specific task. Those pre-trained models can be leveraged by finetuning them on a new task.

Transformer models are made up of encoders only, decoders only, or encoder-decoder architectures. The pre-trained ByT5 model used by this thesis is an encoder-decoder based transformer, like the one originally proposed by Vaswani *et al*. (2017).

Chapter Four

Datasets and Experimental Setup

During the experimentation phase, this thesis utilized multiple datasets for training purposes. In addition, several test sets were used to evaluate the performance of different model configurations in pursuit of finding the best-performing model configuration.

This thesis utilized the JODA dataset, which is introduced for the first time, as well as the Tashkeela dataset. The Tashkeela dataset is a collection of Arabic texts primarily derived from books crawled from the web. It is a corpus of vocalized Arabic text that covers both Classical Arabic and Modern Standard Arabic, with classical sources constituting most of the corpus (Zitouni *et al*. 2009).

This chapter introduces the datasets used, outlines the training methodology, and covers the ByT5 model hyperparameters, experimental setup, and performance evaluation metrics. The aim of this chapter is to provide a thorough understanding of the dataset preparation process and experimentation setup used.

4.1 Training Datasets

Within this thesis, multiple training datasets are used for different purposes. In this section, we outline the different training datasets used within the experiments conducted.

4.1.1 Clean-50

Fadel *et al*. (2019) filtered 50,000 sequences from the Tashkeela dataset based on a diacritic-to-character rate of at least 80%. Several filtering heuristics were used to post-process the dataset, including correcting diacritics, removing English letters and whitespaces, and isolating numbers from words.

For this thesis, the Clean-50 dataset was wrapped to a maximum sequence length of 512 bytes. This ensured consistency with the JODA dataset (which will be discussed in subsequent

sections) sequence length used. The wrapping technique involved evaluating each sequence and truncating it at the last period, comma, or whitespace found before the 512-byte limit.

Table 2 illustrates various characteristics of the Clean-50 dataset used.

Table 2. Clean-50 Dataset Characteristics

Criteria	Split		
	Train	**Validation**	**Test**
Size (MB)	12.80	0.63	0.65
Number of sequences	50,000	2,500	2,500
Word count ($\times 10^3$)	1,619.89	79.46	81.48
Character count ($\times 10^3$)	7,338.44	359.55	370.01
Average number of characters per word	4.53	4.52	4.54
Average number of words per sequence	32.40	31.78	32.59

4.1.2 Clean-400

Abdel Karim and Abandah (2021) employed Fadel *et al*. (2019) filtering and cleaning heuristics to extract a larger dataset from the Tashkeela corpus. The authors ensured a diacritic-to-character ratio of 80% and provided a larger training dataset that we utilize in this thesis.

For the purpose of this thesis, the Clean-400 dataset was wrapped to a maximum sequence length of 512 bytes. This ensured consistency with the other datasets used. The wrapping technique involved evaluating each sequence and truncating it at the last period, comma, or whitespace found before the 512-byte limit.

Table 3 illustrates various characteristics of the Clean-400 dataset used.

Table 3. Clean-400 Dataset Characteristics

Criteria	Split		
	Train	**Validation**	**Test**
Size (MB)	102.50	0.63	0.65
Number of sequences	400,000	2,500	2,500
Word count ($\times 10^6$)	12.95	0.08	0.08
Character count ($\times 10^6$)	58.67	0.36	0.37
Average number of characters per word	4.53	4.52	4.54
Average number of words per sequence	32.36	31.78	32.59

4.1.3 JODA

JODA is the first dataset of its kind to offer sentences in the Jordanian dialect alongside their counterparts in MSA. The JODA dataset was collected from various sources, including social media platforms such as Facebook, YouTube, Instagram, and Twitter, as well as other online datasets like the Shami-Corpora and the DART-Dataset. The dataset primarily contains Jordanian dialect sequences but also includes MSA sequences with minor mistakes.

Stratification was used when dividing the JODA dataset into training, validation, and test subsets. This ensured equal sampling from different sources and dialect types (Jordanian dialect and MSA). Table 4, as well as figures 5 and 6, show various characteristics of the JODA dataset.

Table 4. JODA Dataset Characteristics

Criteria	Value
Size (MB)	5.52
Number of sequences	59,135
Word count ($\times\ 10^6$)	0.57
Character count ($\times\ 10^6$)	3.02
Average number of characters per sequence	51.1
Average number of words per sequence	9.7

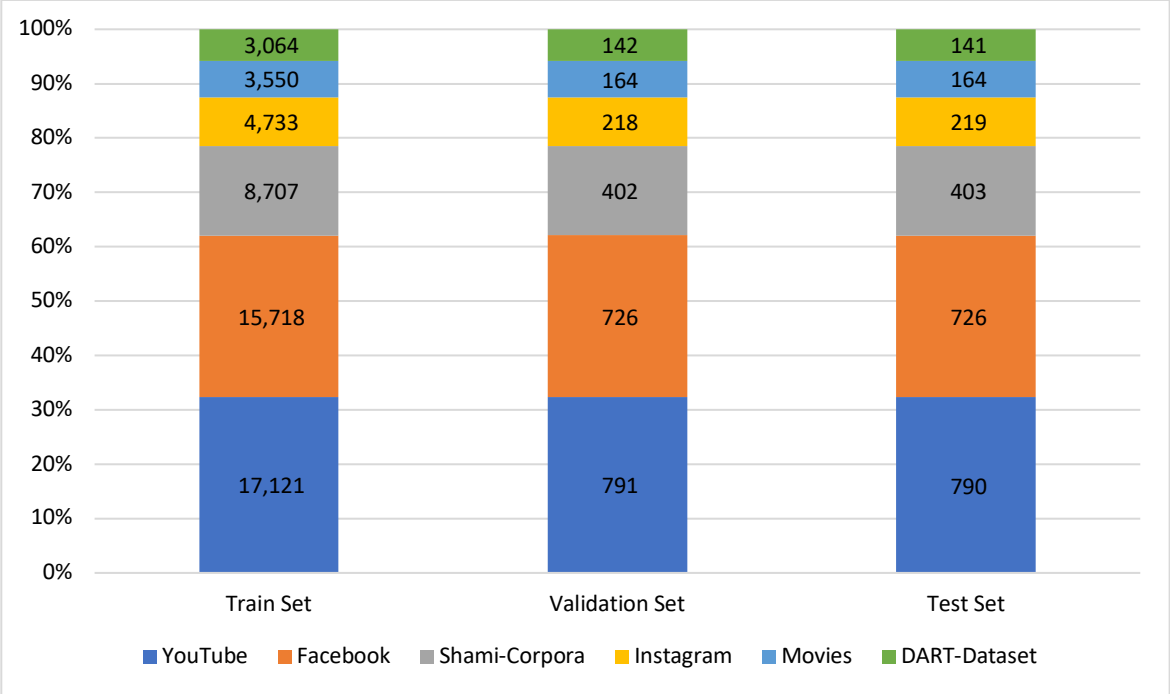

Figure 5. JODA Dataset Sample Sources

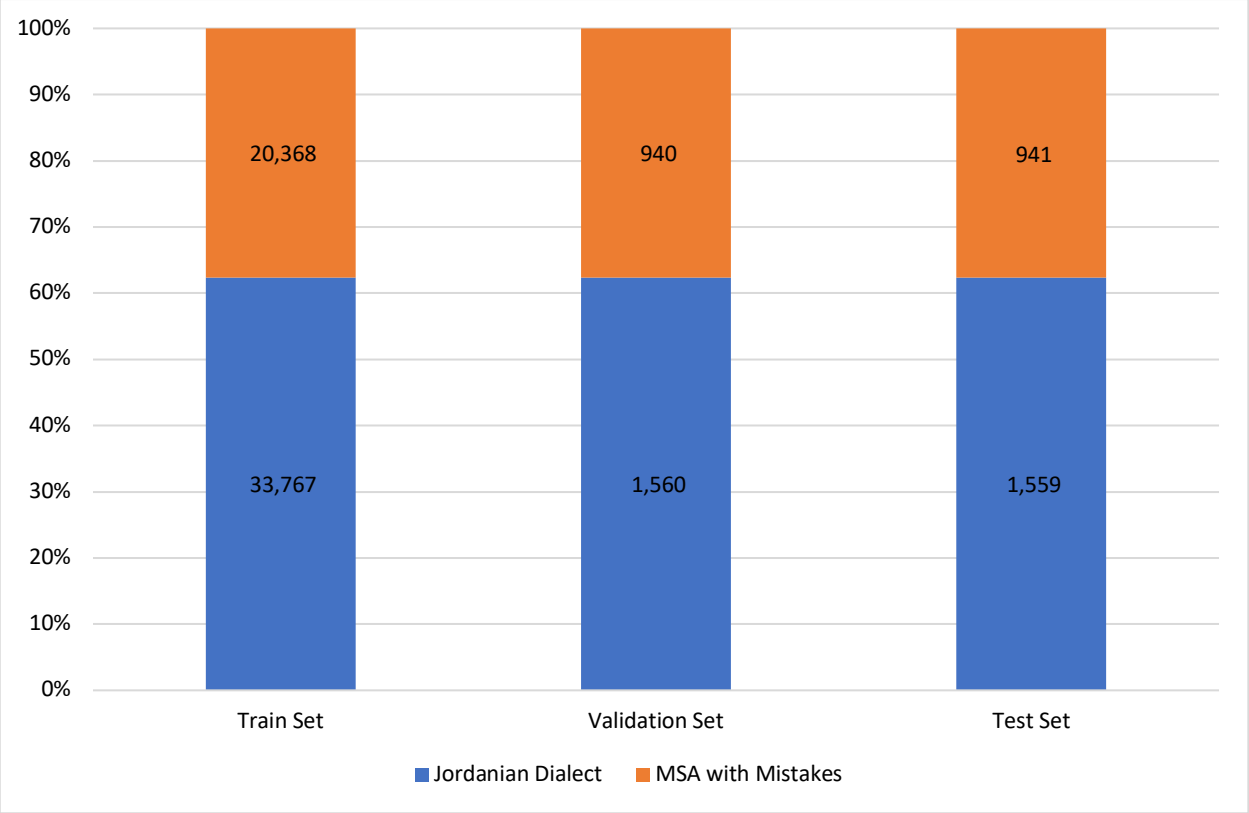

Figure 6. JODA Dataset Sample Type

4.2 Error Injection Techniques

We hypothesize that the JODA dataset can be complemented by other data sources to achieve better results. Therefore, we include the Clean-50 and Clean-400 datasets in our set of experiments. While the JODA dataset contains input and target sequence pairs, the Tashkeela dataset does not. Both the Clean-50 and Clean-400 datasets consist of Arabic texts. These datasets need to be prepared in such a way that input-target pairs are extracted.

In preparing the Tashkeela dataset, we utilize stochastic error injection with different error injection rates and techniques. By doing so, we aim to generate a synthetic dataset that will hopefully help our model correct various error types found in the JODA dataset. We also examine the best error injection rate and technique for our use case.

4.2.1 Directed Error Injection

Soft spelling mistakes are widespread among native Arabic speakers and foreign learners alike. These mistakes occur due to the complex rules that dictate the correct usage of orthographic variations of some Arabic letters. Given the identical phonetic sounds, people often confuse such letters.

In injecting directed error mistakes that reflect soft spelling mistakes often committed by Arabic speakers, we resort to the method proposed by Abandah *et al*. (2022). The authors propose an effective technique that converts one-letter and two-letter combinations within an Arabic word that are prone to spelling mistakes. This includes the combination of alef-hamza (اء) and the letters at word ends (ه، وا، و، ت).

The conversion used by the authors is positional in nature. For instance, the combination of waw-alef (وا) is only processed when it appears at word endings where the associated error frequently occurs. If the same combination appears in the middle of a sentence, no conversion is performed. On the other hand, the combination of alef-hamza (اء) is converted wherever it occurs in the word due to its susceptibility to soft mistakes both in the middle and at the end of a word.

This approach is meant to train the model to correct artificial errors randomly injected into the input sequences. Errors are injected by replacing the target letters with soft Arabic spelling mistakes. With an error injection rate, a letter belonging to one of the target letters in table 5 is randomly replaced by one of the choices in the corresponding replacement group.

This thesis investigates three error injection rates: 2.5%, 10%, and 40%, similar to the rates proposed by Abandah *et al*. (2022).

Table 5. Directed Error Targets and Replacement Groups

Target Letters	Replacement Group
ء ، آ ، أ ، ؤ ، إ ، ئ	اء ، ء ، آ ، أ ، ؤ ، إ ، ئ ، ا
ا	ى ، ء ، آ ، أ ، ؤ ، إ ، ئ ، ا
ة	ة ، ت ، ه
ى	ى ، ا
اء (end of sentence)	اء ، ء ، آ ، أ ، ؤ ، إ ، ئ ، ا
و ، وا (end of sentence)	وا ، و
ه (end of sentence)	ة ، ه
ت (end of sentence)	ة ، ت

4.2.2 General Error Injection

This thesis advances Arabic stochastic error injection by proposing a more general technique that is not limited to soft-spelling mistakes. Our general error injection method targets some of the most common error patterns found in texts written by Arabic users, including letter exaggeration (by a random number between 2 and 4 letters), letter deletion, letter insertion, letter swapping, and letter replacement.

In preparing the dataset, we examine the sequences and perform letter exaggeration, deletion, insertion, swapping, or replacement if the letter is an Arabic letter, using three error injection rates: 2.5%, 10%, and 40%, similar to the rates used by Abandah *et al*. (2022). To determine which error injection pattern to apply (exaggeration, deletion, insertion, swapping, or replacement), we rely on a pre-set probability distribution.

This approach is designed to train the model to correct artificial errors randomly injected into the input sequences. Table 6 highlights the different error injection patterns and the probability of each pattern occurring, given that a general error is injected for a given letter.

Table 6. General Error Injection Patterns and Examples

Pattern	Probability	Example	
		Original word	Modified word
Exaggeration	10%	سنعود	سنعووود
Deletion	10%	ذهبت	ذبت
Insertion	10%	اليوم	اليونم
Swapping	10%	الجامعة	الجماعة
Replacement	60%	الوطن	الوتن

4.3 Test Sets

Due to the various training tasks attempted, and to appropriately evaluate the model's performance, multiple test sets are used. The performance on these different test sets guides our choice of model configuration and training dataset.

4.3.1 Test200

The Test200 is an Arabic test set that contains real samples with soft spelling mistakes. The test set consists of 200 sequences and has previously been used in literature (Abandah *et al.*, 2022). The samples present a challenging collection of soft spelling mistakes, with an average of 6.5 mistakes per sequence and a total character error rate of around 5%.

The Test200 set provides a good source for evaluating a model's performance in correcting directed spelling mistakes. This test set will be used to determine hyperparameter fine-tuning when selecting a complementary training dataset that contains soft spelling mistakes.

Table 7. Test200 Set Characteristics

Criterion	Count
Number of Sequences	200
Word Count	2,443
Character Count	24,002
Number of Mistakes	1,306
Mistakes per Sequence	6.5

4.3.2 JODA Test Set

The JODA test set is a subset of the JODA dataset. When extracting the subset, stratification was used to ensure similar sampling to the training dataset from different sources: Facebook, YouTube, Instagram, Twitter, and other online datasets such as the Shami-Corpora and DART-Dataset. The test set also has equal representation of the Jordanian dialect and MSA with mistake sequences, similar to the training dataset.

The JODA test set is a good source for comparing different model configurations. Results reported on the subset will define the hyperparameter fine-tuning approach.

Table 8. JODA Test Set Characteristics

	Category	Number of Sequences
Source	YouTube	790
	Facebook	726
	Sami-Corpora	403
	Instagram	219
	Movies	164
	DART-Dataset	141
	Twitter	57
Type	Jordanian Dialect	1,559
	MSA with Mistakes	941

4.3.3 TSMTS

To evaluate the different general error injection techniques experimented with, we introduce a new test set that contains synthetic general errors. The Tashkeela Spelling Mistakes Test Set (TSMTS) aims to provide a means for evaluating different general error injection rates, similar to how the Test200 set is used to evaluate directed error injection. Like the Test200 set, we ensured a character error rate of around 5% for consistency.

The TSMTS offers a good source for evaluating a model's performance in correcting general spelling mistakes. This test set will be used to determine hyperparameter fine-tuning when selecting a complementary training dataset that contains general spelling mistakes. TSMTS consists from 2,500 sequences. Following table shows the different errors injected into the set.

Table 9. TSMTS Characteristics

Error Type	Number of Occurrences
Exaggeration	1.5 k
Deletion	1.6 k
Insertion	1.6 k
Swapping	1.3 k
Replacement	10.3 k
No Error Injected	352 k

4.4 ByT5 Model Selection

In addition to testing with the ByT5 small model, we tested with the ByT5 base model to evaluate its performance on the tasks examined. As reported by Edman *et al*. (2024), larger model sizes tend to perform better in translation tasks. Due to limitations in computational resources, we were not able to test models larger than the base model. In table 10, we highlight the specifications for the small and base ByT5 models.

Table 10. ByT5 Model Parameters

Criteria	Small	Base
Parameters	300M	582M
Encoder/Decoder Layers	12/4	18/6
Feed Forward Dimension (d_{ff})	3584	3968
Model Dimension (d_{model})	1472	1536

4.5 ByT5 Model Hyperparameters

Hyperparameter fine-tuning is crucial for adjusting the weights of artificial intelligence networks. Throughout the experimentation phase, hyperparameter fine-tuning was conducted to find the best-performing model configuration. The process included modifying the following:

- **Batch Size:** As part of the fine-tuning process, the batch size was adjusted. Using smaller batch sizes can help escape local minima and potentially lead to better generalization. Conversely, using larger batch sizes can provide more accurate and stable gradient estimates, but may also result in diminishing returns in terms of performance improvement. The different batch size values were based on experiments conducted by (Al-Rfooh *et al*. 2023).
- **Learning Rate:** Modifying the learning rate allows the model to improve its convergence. Smaller learning rates lead to slower adjustments in the model's parameters during the training process, ensuring training stability and promoting gradual convergence. In contrast, higher learning rates lead to larger weight updates, which speeds up the training process but increases the likelihood of weight oscillations, leading to training instability. Several learning rates were experimented with to find the optimal one for our use case. The learning rate values used were based on those experimented with by (Al-Rfooh *et al*., 2023).
- **Optimizer Type:** Two optimizers were tested to assess their effectiveness in training the model. The AdaFactor optimizer was experimented with, as it is the default optimizer used by the ByT5 pretrained models and is based on the AdaFactor algorithm, providing a way to automatically set learning rates based on parameter size. Additionally, the Adam Weight Decay Optimizer was used, which

is built on the Adam algorithm and reduces L2 weight based on a certain function to reduce overfitting.

- **Maximum Sequence Length:** The maximum sequence length refers to the maximum number of bytes for a single sequence. Since the ByT5 model operates at a byte level, specifying this hyperparameter has a crucial impact on the model's performance. For our experiments, we used a maximum sequence length of 512 bytes, as that is the maximum length observed within the JODA dataset. Other datasets used were adjusted to accommodate the same sequence length. Throughout our experiments, the maximum sequence length was not altered, as the JODA dataset sequences were manually audited while maintaining logical and coherent sentences.
- **Model Size:** The model size was altered to observe its impact on performance. As reported by Edman *et al*. (2024), larger ByT5 models tend to perform better in translation tasks. Two model sizes were experimented with: small and base.
- **Dataset:** As we aim to find a model able to generalize its performance and not be limited to the JODA dataset, we utilized the Clean-50 and Clean-400 datasets as complementary datasets during the training process. We compared the effectiveness of these datasets in enhancing the model's performance.
- **Training Steps:** The number of training steps was tweaked based on the model's performance. Once overfitting was detected, training was stopped. Hence, the number of training steps is not the same for different experiments.

4.6 Experimental Frameworks and Platforms

The experiments were conducted using Google's Colab Pro Plus platform. TPU v2 units were used within the training environment to facilitate the training process. TPU v2 units are part of a set of interconnected units via a high-speed 2D mesh network, supported by host machines running on central processing units (CPUs). Table 11 highlights the different specifications for the training environment:

Table 11. Training Environment Specifications

Hardware	Specification
CPU	Intel® Xeon® CPU @ 2.20GHz, 4 cores, 56 MB cache
TPU	V2
GPU	Nvidia-Tesla V100-SXM2 @ 1.32GHz, 16 GB Nvidia-Tesla P100-PCIE @ 1.32 GHz, 16 GB
Memory	50 GB
Libraries	Python 3.7.13, TensorFlow 2.12.0, Tensorboard 2.12.0, Tensor2tensor 1.15.7, Jax 0.4.25, NumPy 1.23.5, Pandas 1.5.3, Datasets 2.5.0

Parallel training was facilitated by using the Mesh TensorFlow and SeqIO libraries.

4.7 ByT5 Training

In our training we adapt the methodology outlined by Phakmongkol *et al*. (2021), the original authors of the paper that introduced the T5 model. In their study, the authors use a baseline configuration and alter one hyperparameter at a time. While this "coordinate ascent" approach might miss second-order effects (for example, a different learning rate might work better with a larger model size), performing a combinatorial exploration of all of the factors would be prohibitively expensive.

Similar to the authors' approach, we will systematically change one hyper-parameter at a time and will compare results to the baseline model. Once the best individual hyperparameters are found, we will combine all of them under one model configuration and train the model accordingly. Table 12 shows the baseline model's hyperparameters:

Table 12. Baseline Model Hyperparameters

Hyperparameter	Value
Maximum Sequence Length	512 Bytes
Learning Rate	0.003
Batch Size	256
Optimizer	AdaFactor Optimizer

The main task this thesis addresses is translating Jordanian dialect to MSA. We will therefore train and test the model using the JODA dataset and optimize on its performance by finetuning the different hyperparameters. Once a satisfactory performance is found, we will attempt to complement our training process with the Clean-50 and Clean-400 datasets to try to enhance our model's performance beyond the JODA dataset.

ByT5's training process is a step-by-step process, where model parameters are updated based on batches of data. The model's weights and parameters are strategically saved at defined intervals called checkpoints. This allows for model training and evaluation at different intervals.

4.8 Performance Evaluation

The performance of the model is evaluated using the following metrics:

- **BLEU Score:** The BLEU (Bilingual Evaluation Understudy) score is a metric used to evaluate the quality of translations. It measures the overlap of n-grams between the LLM-generated text and the reference text. The score ranges from 0 to 1, where

higher values indicate closer matches to the reference translation. The BLEU score is used to evaluate the quality of translation from the Jordanian dialect to MSA.

- **Character Error Rate (CER):** The CER is the percentage of incorrectly predicted characters compared to the target characters. It is measured using the overall edit distance, which indicates the number of changes needed to convert the input text to the target text. The CER is used to measure the quality of restoring general and directed errors.

In evaluating translation tasks, the BLEU score is used. In contrast, the CER is used to evaluate the quality of restoring general and directed errors.

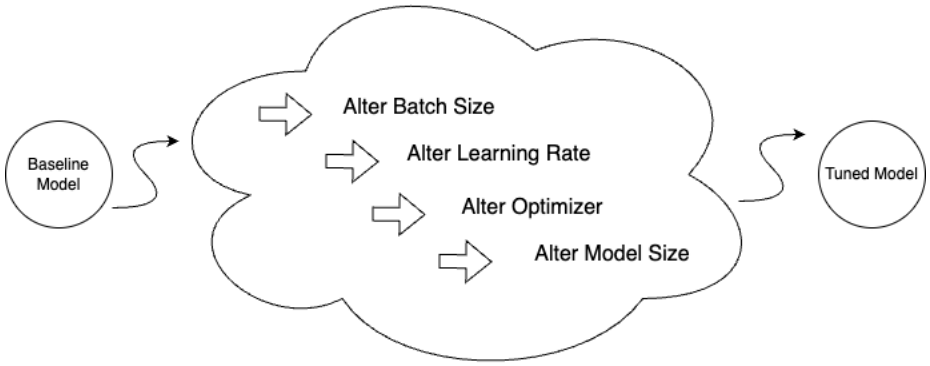

Figure 7: Visual Representation of Experiments

Chapter Five

Experiments and Results

This chapter outlines the different experiments conducted, discusses the results, and evaluates the model's performance using the metrics outlined in Chapter 4.

5.1 Baseline Model

Table 13. Baseline Model Hyperparameters

Hyperparameter	Value
Maximum Sequence Length	512 Bytes
Learning Rate	0.003
Batch Size	256
Optimizer	AdaFactor Optimizer

As outlined in Chapter 4, our approach is to use a baseline model and change different hyperparameters sequentially, in hopes of finding the best hyperparameter combination for our task. For our base-line model, we used the pretrained ByT5 small model, with the AdaFactor optimizer, a maximum sequence length of 512 bytes, a learning rate of 0.003, and a batch size of 256. The best BLEU score for the baseline model as reported on the test set will be our benchmark moving forward.

After training the model on the JODA training set, the model's performance as reported on the validation set stabilizes after 3,000 steps of training with no further improvements. As overfitting is evident after 3,000 steps, training is stopped. Figure 8 shows the BLEU score as reported on the JODA train and validation sets for the baseline model configuration.

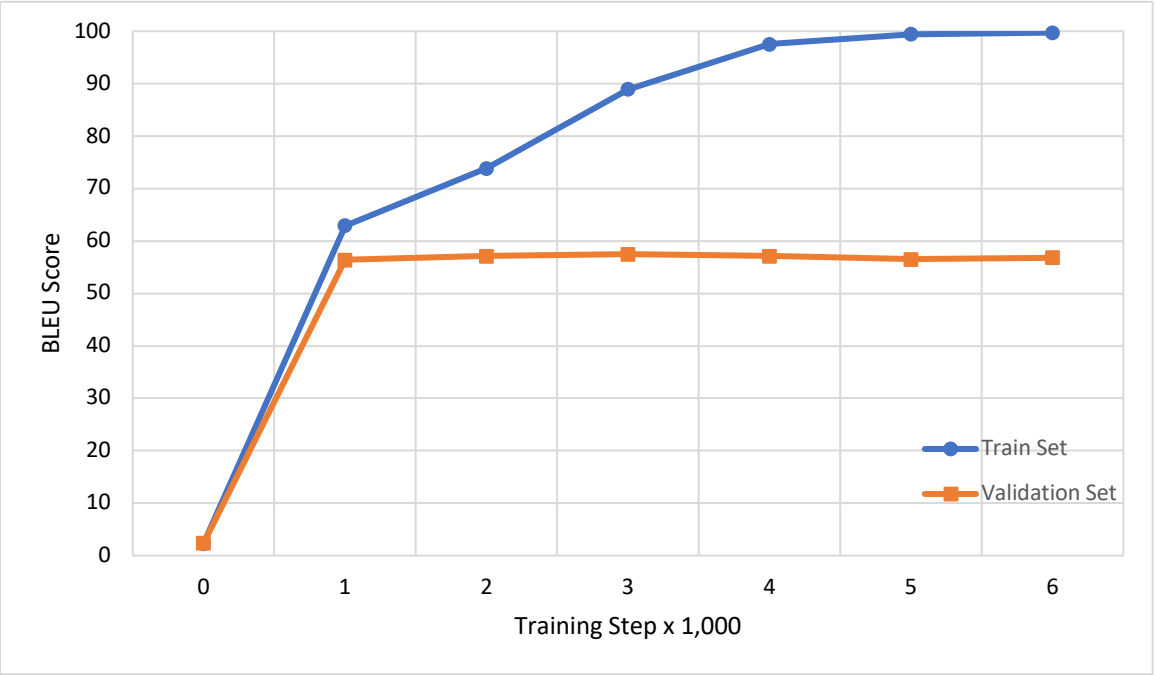

Figure 8. Training Curve for Baseline Model

- The best BLEU score on the validation set is reported at step 3,000 with a score of 57.49.
- The corresponding BLEU score reported on the test set at step 3,000 is 56.07.

5.2 Effect of Batch Size

In our investigation, we try different batch sizes while training the model. While our baseline model was trained with a batch size of 256, we experiment training our model with batch sizes of 128 and 512. The different batch size values we chose were based on experiments conducted by Al-Rfooh *et al*. (2023).

5.2.1 Batch Size of 128

After training the model on the JODA training set, the model's performance as reported on the validation set peaks at training step 4,000. Figure 9 shows the BLEU score as reported on the JODA train and validation sets for the model when trained with a batch size of 128.

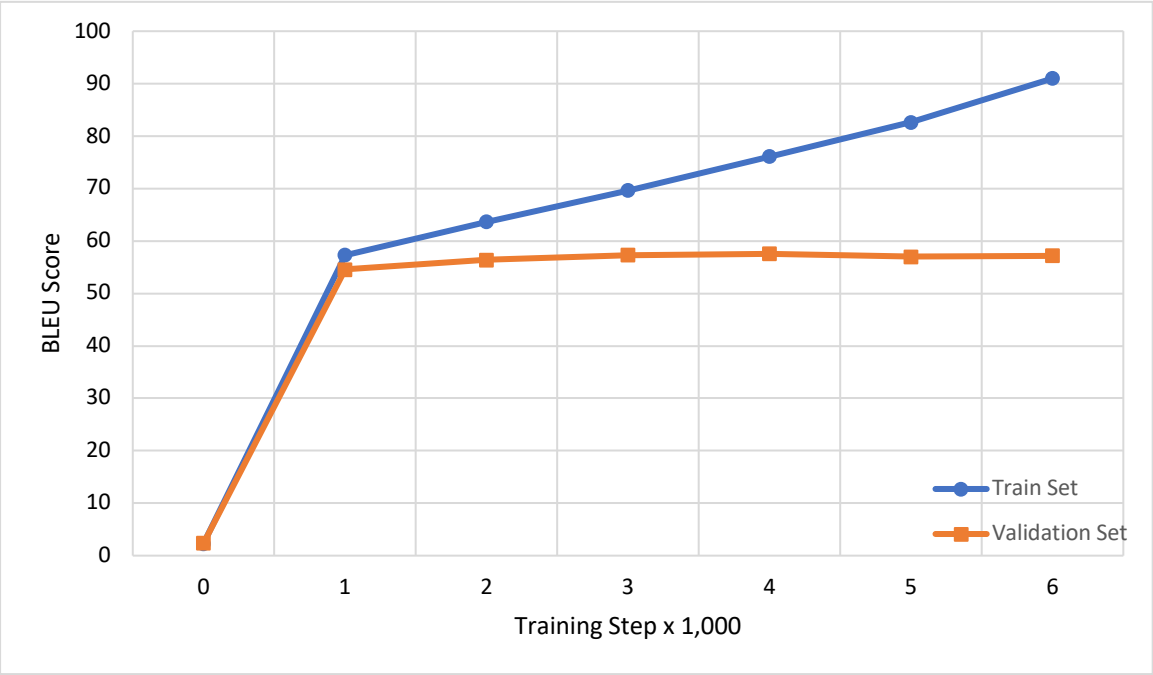

Figure 9. Training Curve with Batch Size of 128

- The best BLEU score on the validation set is reported at step 4,000 with a score of 57.54.
- The corresponding BLEU score reported on the test set at step 4,000 is 56.43.

The model's performance improved when a batch size of 128 was used as opposed to the performance reported by the baseline model.

5.2.2 Batch Size of 512

After training the model on the JODA training set, the model's performance as reported on the validation set peaks at training step 1,000. Figure 10 shows the BLEU score as reported

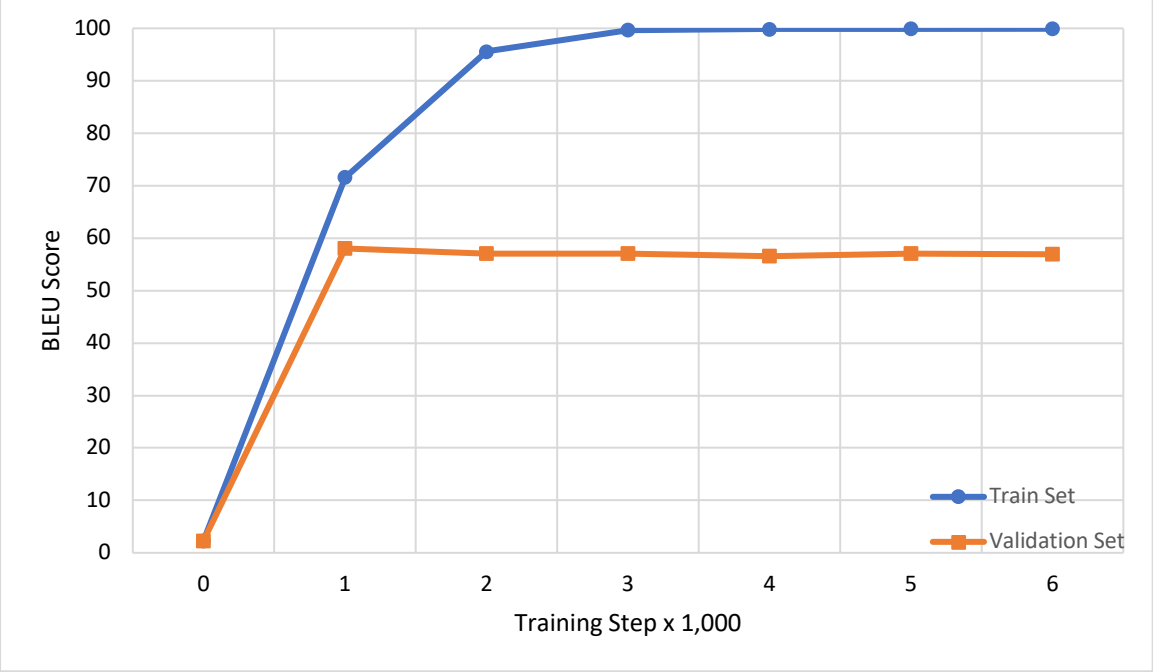

Figure 10. Training Curve with Batch Size of 512

on the JODA train and validation sets for the model when trained with a batch size of 512.

- The best BLEU score on the validation set is reported at step 1,000 with a score of 58.03.
- The corresponding BLEU score reported on the test set at step 1,000 is 56.32.

The model's performance as reported on the test set does improve when a batch size of 512 was used as opposed to the performance reported by the baseline model. Nevertheless, results reported with a batch size of 128 are better.

5.2.3 Comparison of Different Batch Sizes

Figure 11 shows a summary of the BLEU scores reported for different batch sizes. Each BLEU score is reported at the corresponding best validation step for the experiment.

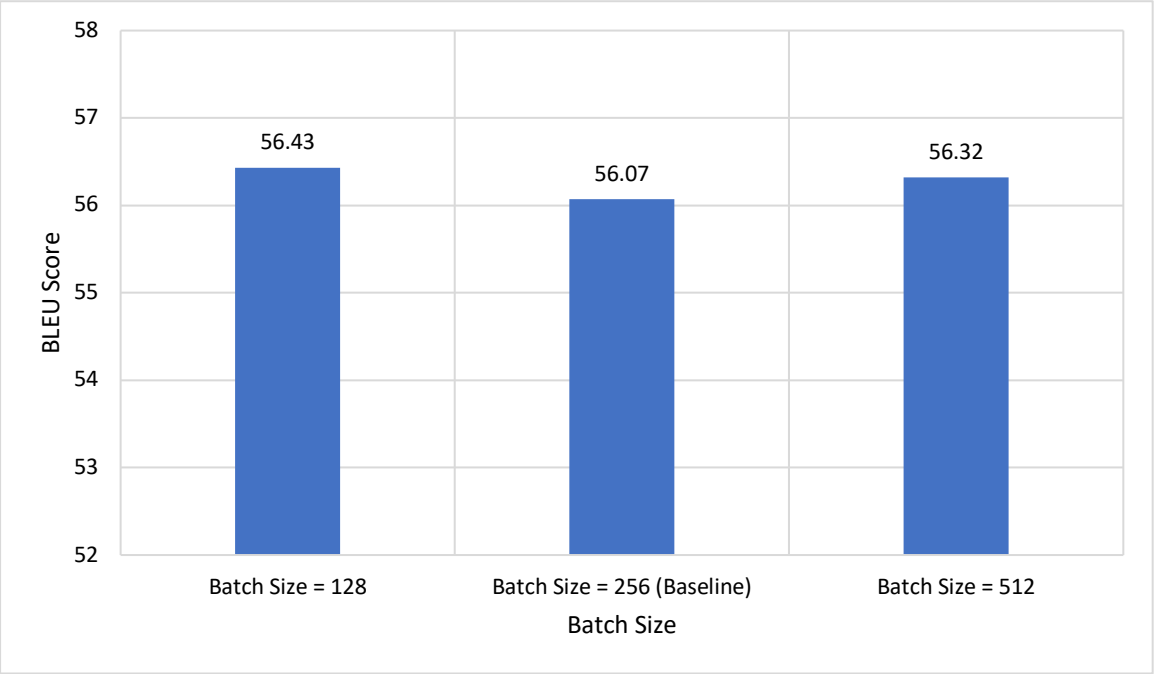

Figure 11. BLEU Scores on JODA Test Set for Three Batch Sizes

As an additional mean of comparison, we report the CER for different batch sizes on the JODA test set (overall), the Jordanian Dialect subset from the JODA test set, and the MSA with Mistakes subset from the JODA test set.

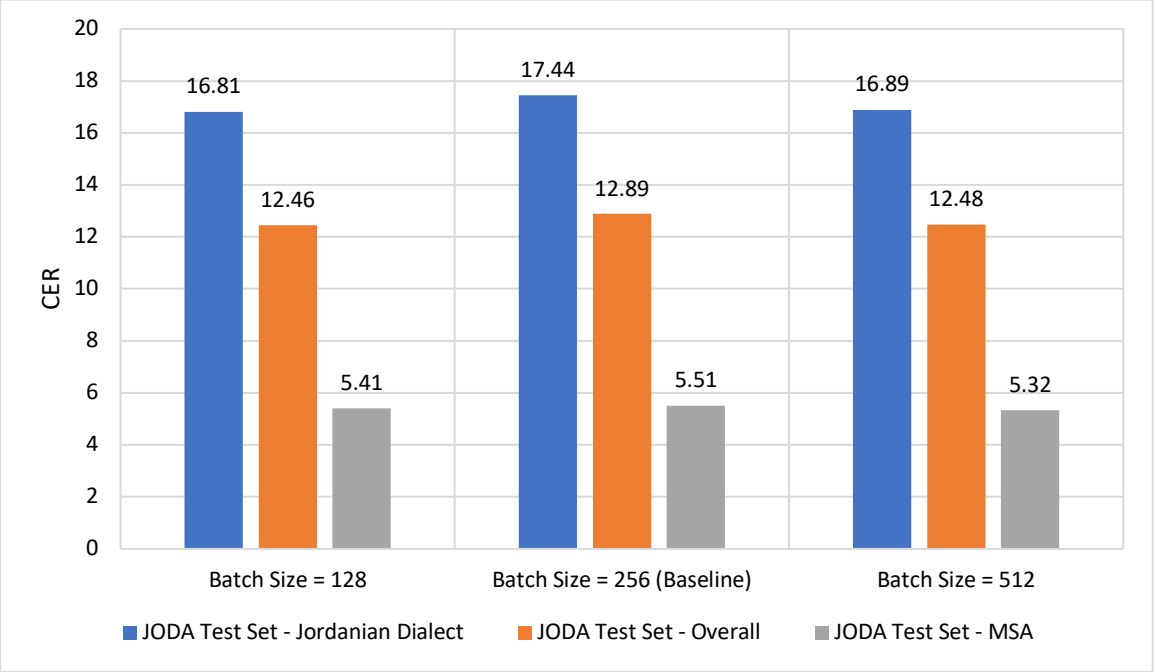

Figure 12. CER for Multiple Batch Sizes and Test Subsets

As observed in the figures 11 and 12, a batch size of 128 shows the best performance on the JODA test set with a notable BLEU score of 56.43. The model also shows better CER on the different JODA test set subsets.

5.3 Effect of Learning Rate

In our investigation, we try different learning rates while training the model. While our baseline model was trained with a learning rate of 0.003, we experiment training our model with learning rates of 0.01 and 0.0001. The different learning rate values we chose were based on experiments conducted by Al-Rfooh *et al*. (2023).

5.3.1 Learning Rate of 0.0001

After training the model on the JODA training set, the model's performance as reported on the validation set does not reach satisfactory values even after 20,000 steps of training. Figure 13 shows the BLEU score as reported on the JODA train and validation sets for the model when trained with a learning rate of 0.0001.

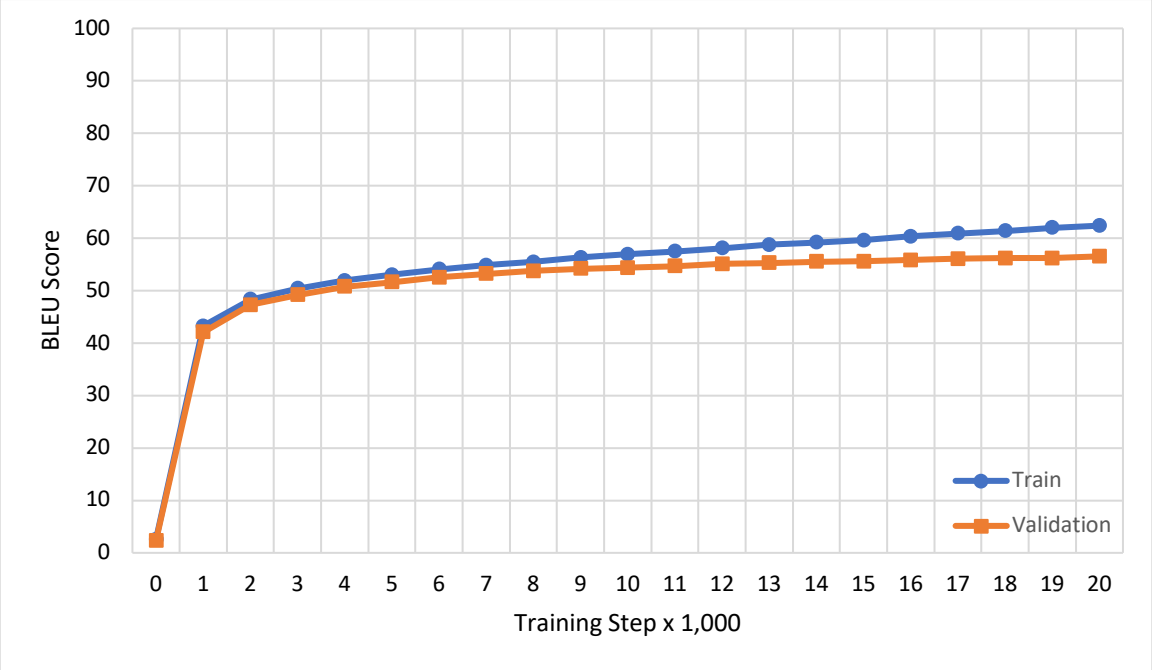

Figure 13. Training Curve with Learning Rate of 0.0001

- The best BLEU score on the validation set is reported at step 20,000 with a score of 56.53.
- The corresponding BLEU score reported on the test set at step 20,000 is 55.38.

The model's performance did not improve when a learning rate of 0.0001 was used as opposed to the performance reported by the baseline model.

5.3.2 Learning Rate of 0.01

After training the model on the JODA training set, the model's performance as reported on the validation set stabilizes after 9,000 steps of training. Figure 14 shows the BLEU score as reported on the JODA train and validation sets for the model when trained with a learning rate of 0.01.

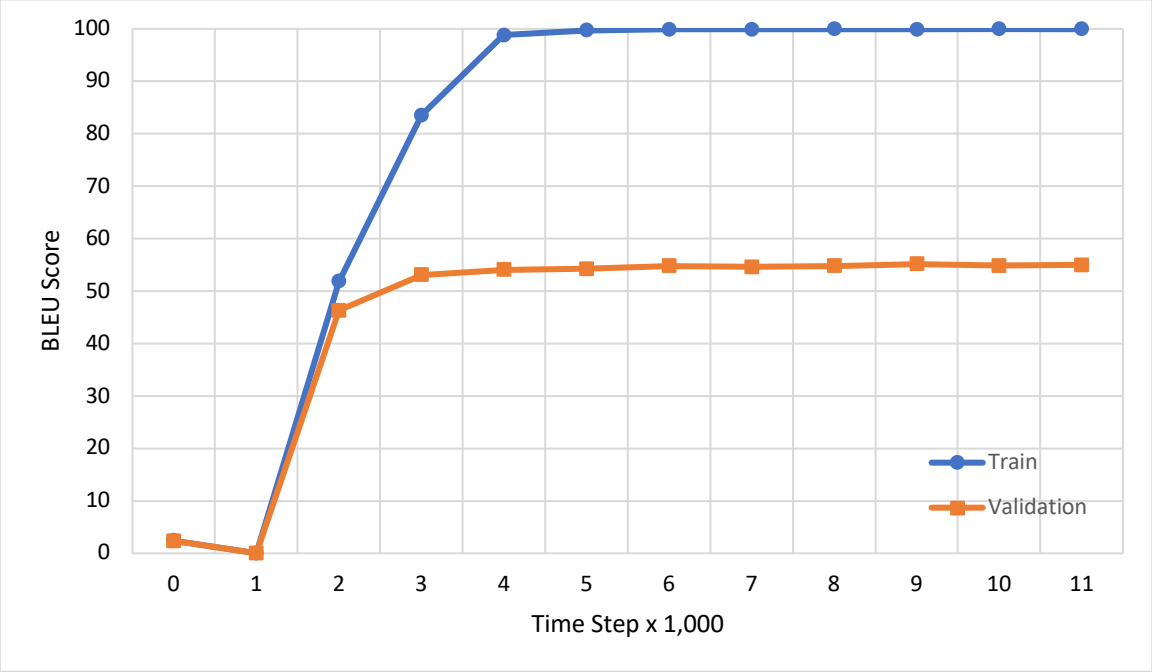

Figure 14. Training Curve with Learning Rate of 0.01

- The best BLEU score on the validation set is reported at step 9,000 with a score of 55.13.
- The corresponding BLEU score reported on the test set at step 9,000 is 53.90.

The model's performance did not improve when a learning rate of 0.01 was used as opposed to the performance reported by the baseline model.

5.3.3 Comparison of Different Learning Rates

Figure 15 shows a summary of the BLEU scores reported for different learning rates. Each BLEU score is reported at the corresponding best validation step for the experiment.

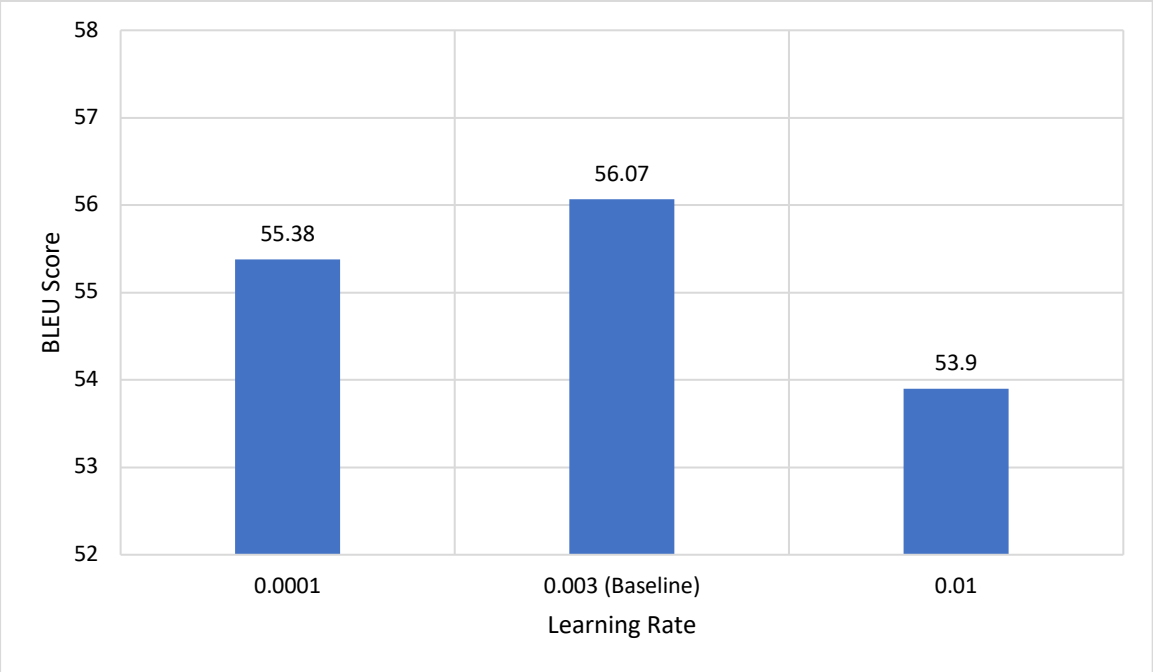

Figure 15. BLEU Scores for Three Learning Rates

As an additional mean of comparison, we report the CER for different learning rates on the JODA test set (overall), the Jordanian Dialect subset from the JODA test set, and the MSA with Mistakes subset from the JODA test set.

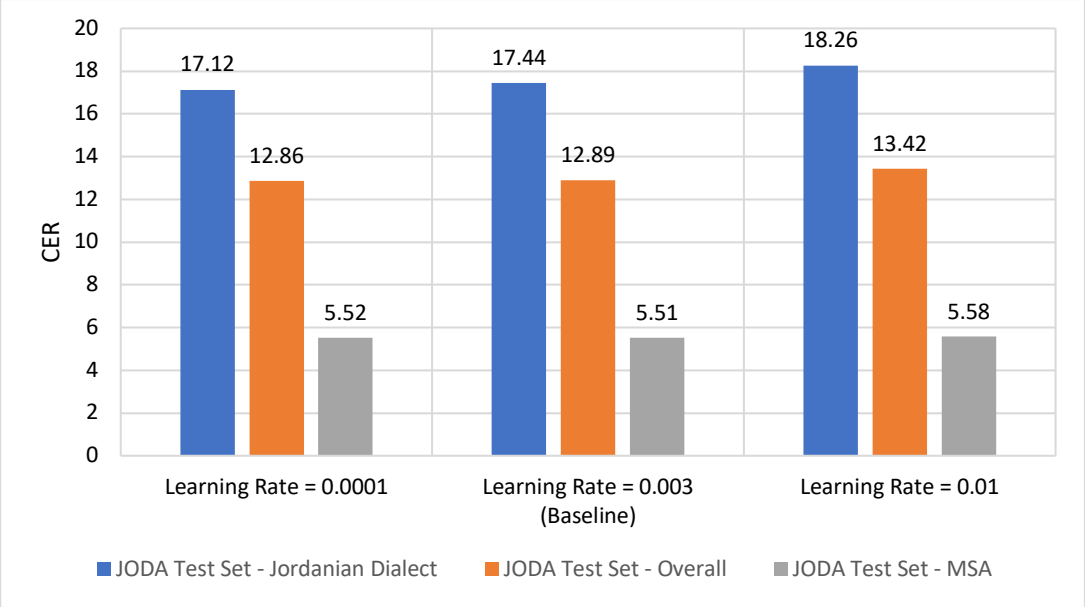

Figure 16. CER for Multiple Learning Rates and Test Subsets

As observed in figures 15 and 16, the baseline model shows the best performance on the JODA test set with a notable BLEU score of 56.07. The model also shows better CER on the different JODA test set subsets.

5.4 Effect of Optimizer

In our investigation, we experiment with training the model with different optimizers. While our baseline model was trained with AdaFactor optimizer, we experiment training our model with the Adam Weight Decay optimizer.

5.4.1 Adam Weight Decay Optimizer

After training the model on the JODA training set, the model's performance as reported on the validation set stabilizes after 2,000 steps of training. Figure 17 shows the BLEU score as reported on the JODA train and validation sets for the model when trained with the Adam Weight Decay Optimizer.

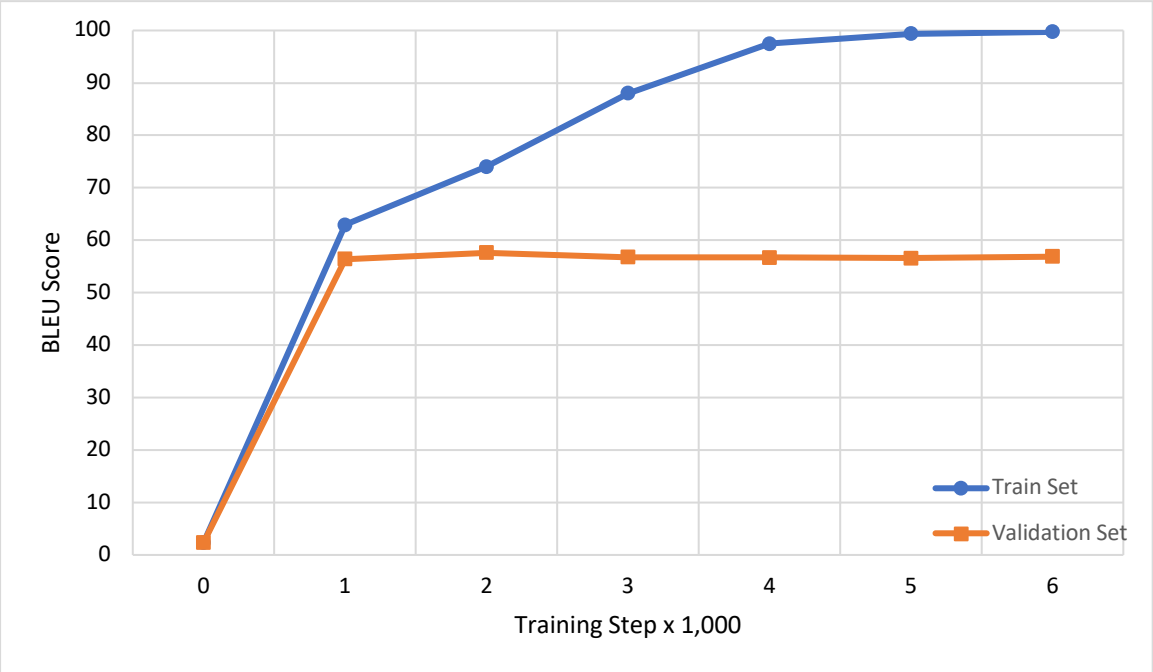

Figure 17. Training Curve with Adam Weight Decay Optimizer

- The best BLEU score on the validation set is reported at step 2,000 with a score of 57.60.
- The corresponding BLEU score reported on the test set at step 2,000 is 56.29.

The model's performance improved when trained with Adam Weight Decay optimizer as opposed to the performance reported by the baseline model.

5.4.2 Comparison of Different Optimizers

Figure 18 shows a summary of the BLEU scores reported for different optimizers. Each BLEU score is reported at the corresponding best validation step for the experiment.

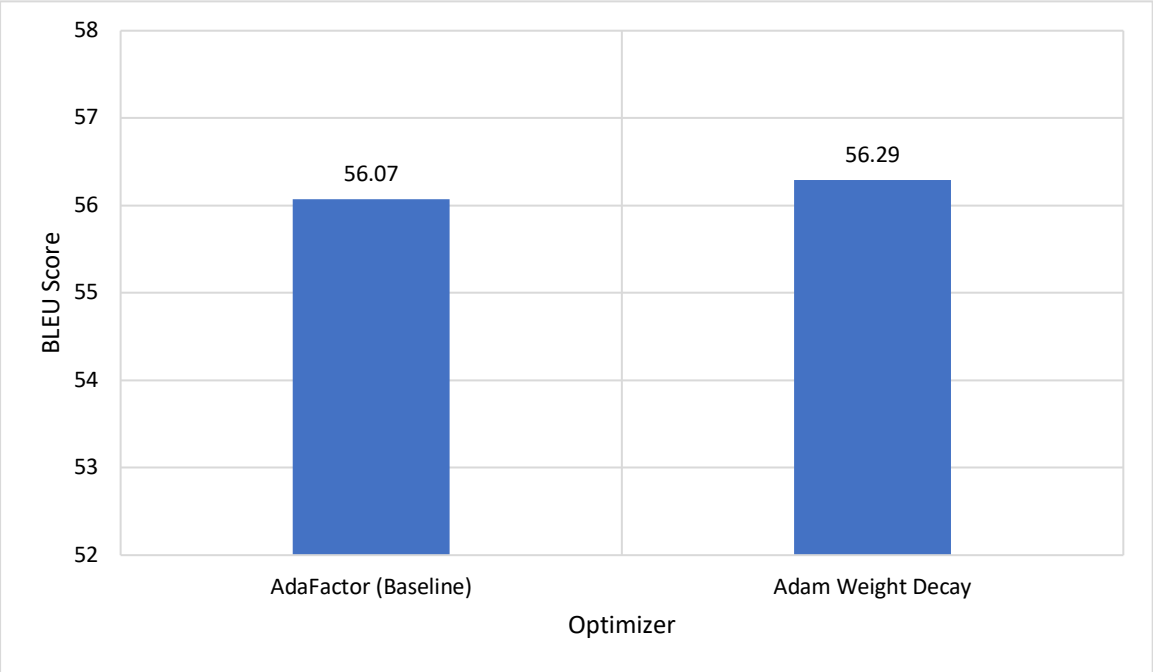

Figure 18. BLEU Scores for Two Optimizers

As an additional mean of comparison, we report the CER for different learning rates on the JODA test set (overall), the Jordanian Dialect subset from the JODA test set, and the MSA with Mistakes subset from the JODA test set.

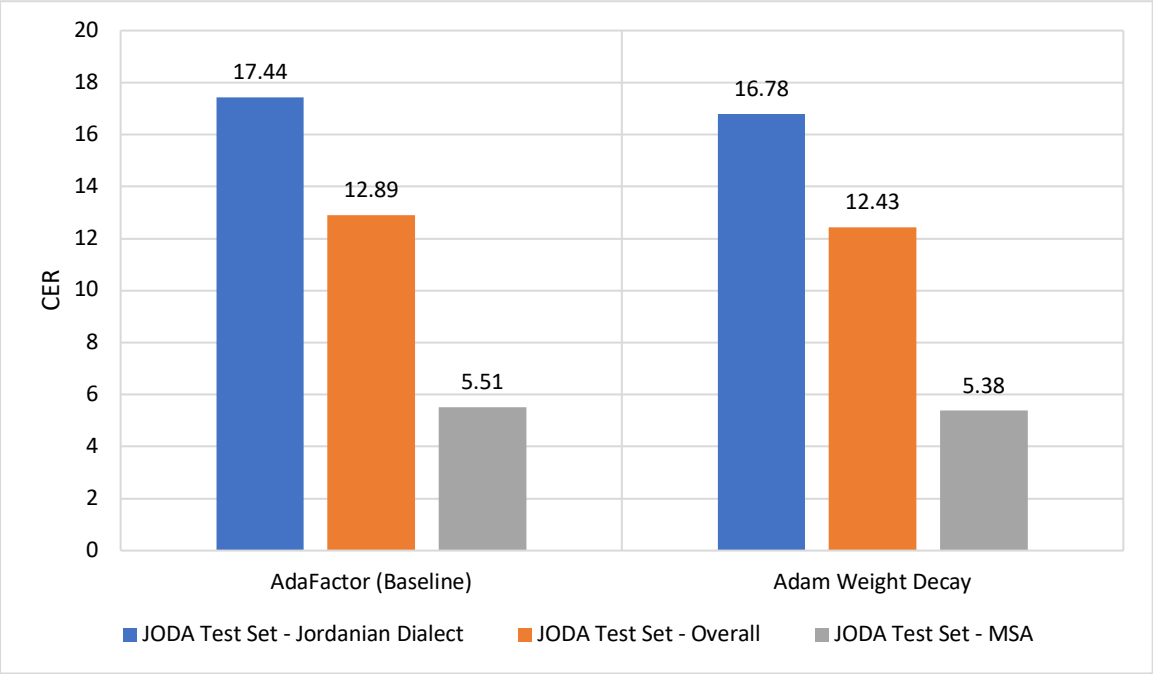

Figure 19. CER for Multiple Optimizers and Test Subsets

As observed in figures 18 and 19, using the Adam Weight Decay optimizer shows better performance on the JODA test set with a BLEU score of 56.29. The model also shows better CER on the different JODA test set subsets.

5.5 Effect of Model Size

In our investigation, we experiment with different model sizes while training the model. While our baseline model was a small model, we experiment training the base model. The choice of experimenting with a larger model was based on the positive experiments reported by Edman *et al*. (2024).

5.5.1 Base Model

After training the model on the JODA training set, the model's performance as reported on the validation set almost stabilizes after 1,500 steps of training. Figure 20 shows the BLEU score as reported on the JODA train and validation sets for the base model.

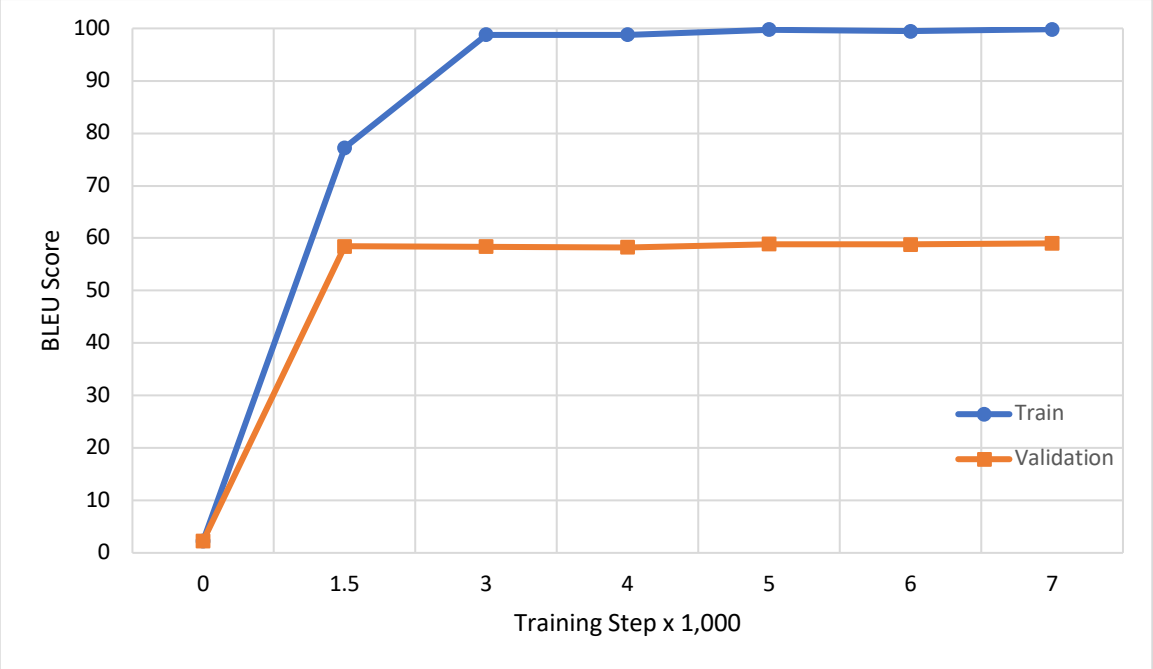

Figure 20. Training Curve with Base Model

- The best BLEU score on the validation set is reported at step 7,000 with a score of 59.01.
- The corresponding BLEU score reported on the test set at step 7,000 is 57.08.

The model's performance improved when a base model was used as opposed to the performance reported by the baseline model.

5.5.2 Comparison of Two Model Sizes

Figure 21 shows a summary of the BLEU scores reported for different model sizes. Each BLEU score is reported at the corresponding best validation step for the experiment.

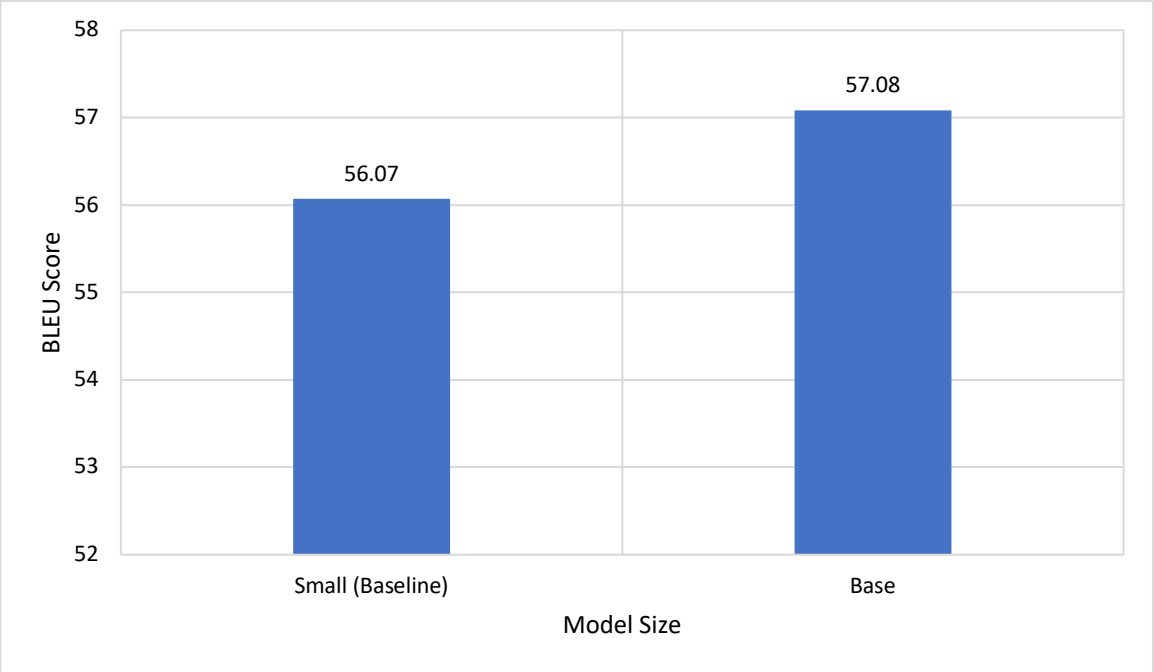

Figure 21. BLEU Scores for Two Model Sizes

As an additional mean of comparison, we report the CER for different model sizes on the JODA test set (overall), the Jordanian Dialect subset from the JODA test set, and the MSA with Mistakes subset from the JODA test set.

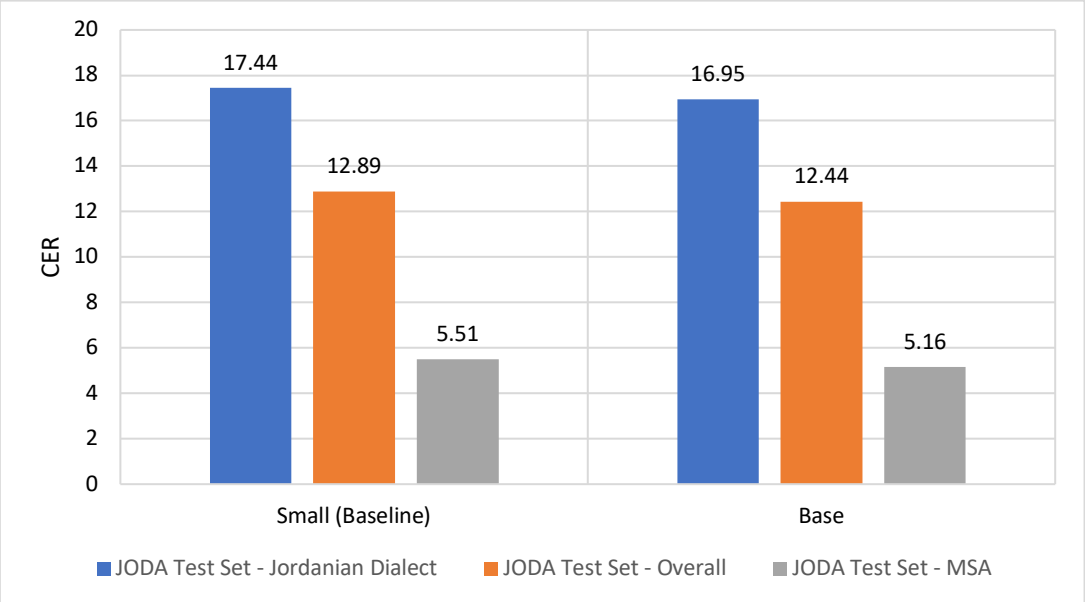

Figure 22. CER for Multiple Model Sizes and Test Subsets

As observed in figures 21 and 22, using the Base model shows better performance on the JODA test set with a BLEU score of 57.08. The model also shows better CER on the different JODA test set subsets.

5.6 Combining Findings

In the previous experiments conducted, we experimented with different batch sizes, learning rates, model sizes, and optimizer types. Here are the best individual hyperparameters found:

- **Batch Size:** a batch size of 128 performed best with a BLEU score of 56.43.
- **Learning Rate:** a learning rate of 0.003 performed best with a BLEU score of 56.07.
- **Optimizer:** the Adam Weight Decay optimizer performed best with a BLEU score of 56.29.
- **Model Size:** the base model performed best with a BLEU score of 57.08.

In this section, we combine the previous findings together and train a new model on the JODA dataset with the given hyperparameters. Figure 23 shows the BLEU score as reported on the JODA train and validation sets for the new model.

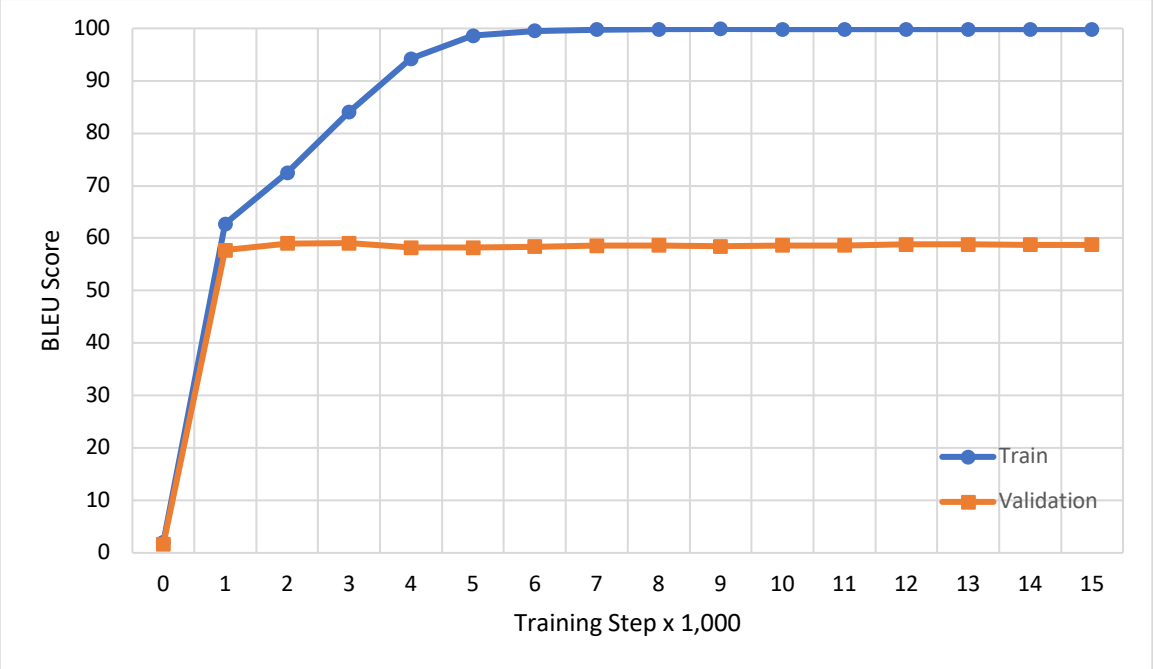

Figure 23. Training Curve with Best Individual Hyperparameters

- The best BLEU score on the validation set is reported at step 3,000 with a score of 59.07.
- The corresponding BLEU score reported on the test set at step 3,000 is 57.77.

The model's performance improved when the previous findings were combined into one model. Figure 24 compares the new model with the baseline model.

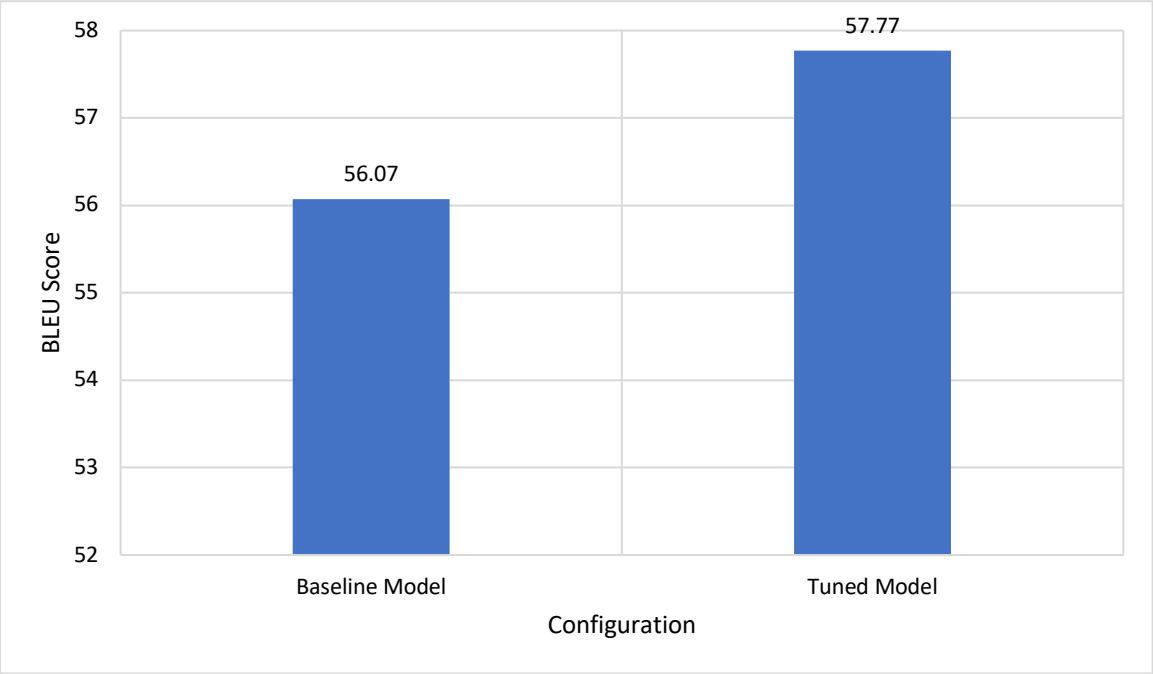

Figure 24. Comparison Between the Baseline and the Best Model

As an additional mean of comparison, we report the CER for different model configurations on the JODA test set (overall), the Jordanian Dialect subset from the JODA test set, and the MSA with Mistakes subset from the JODA test set.

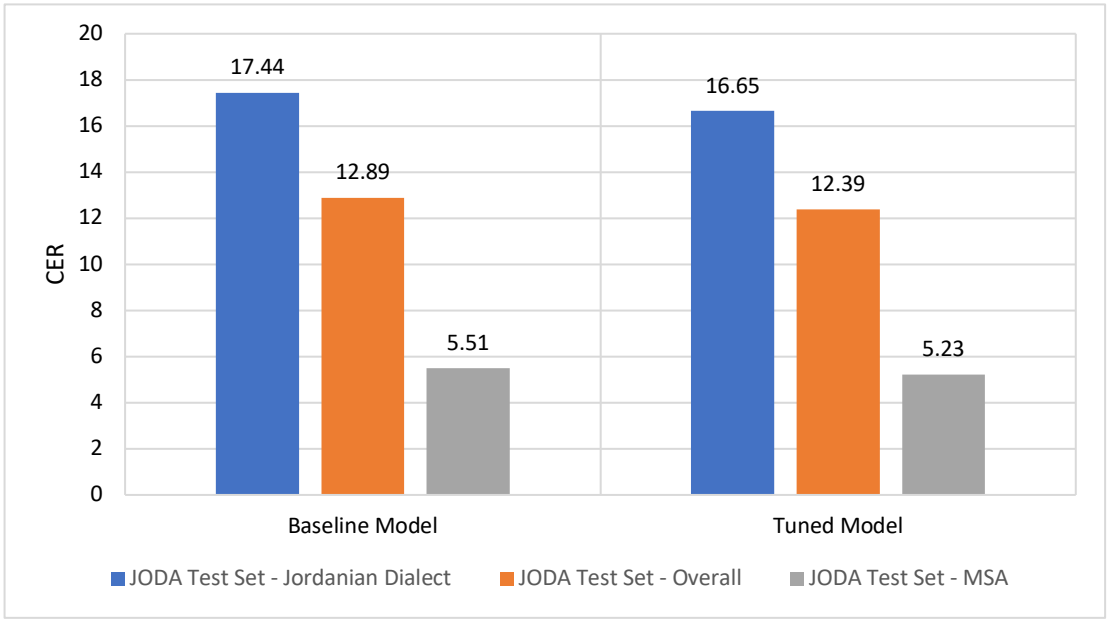

Figure 25. CER for Two Configurations and Test Subsets

5.7 Additional Datasets

The tuned model provides best results thus far. In this section, we try to complement our training using additional datasets. The hypothesis is that by training on additional datasets, the model will be better able to generalize trends beyond the JODA dataset. We target directed soft errors and general errors in our training.

Before training our model with a combination of the JODA and additional datasets, we must determine the best complimentary dataset. In this section, we explore finding the best general error injection rate for the Clean-50 dataset, the best directed error injection rate for the Clean-50 dataset and apply the best error injection rates on the Clean-400 dataset to examine dataset size effect. We conduct the experiments using the baseline model.

5.7.1 Directed Error Injection

As outlined in Chapter 4, we will use the directed error injection technique to prepare the Clean-50 dataset and then train the model using the dataset. We evaluate the results using the CER on the Test200 set. We report results for three error injection rates: 2.5%, 10%, and 40%.

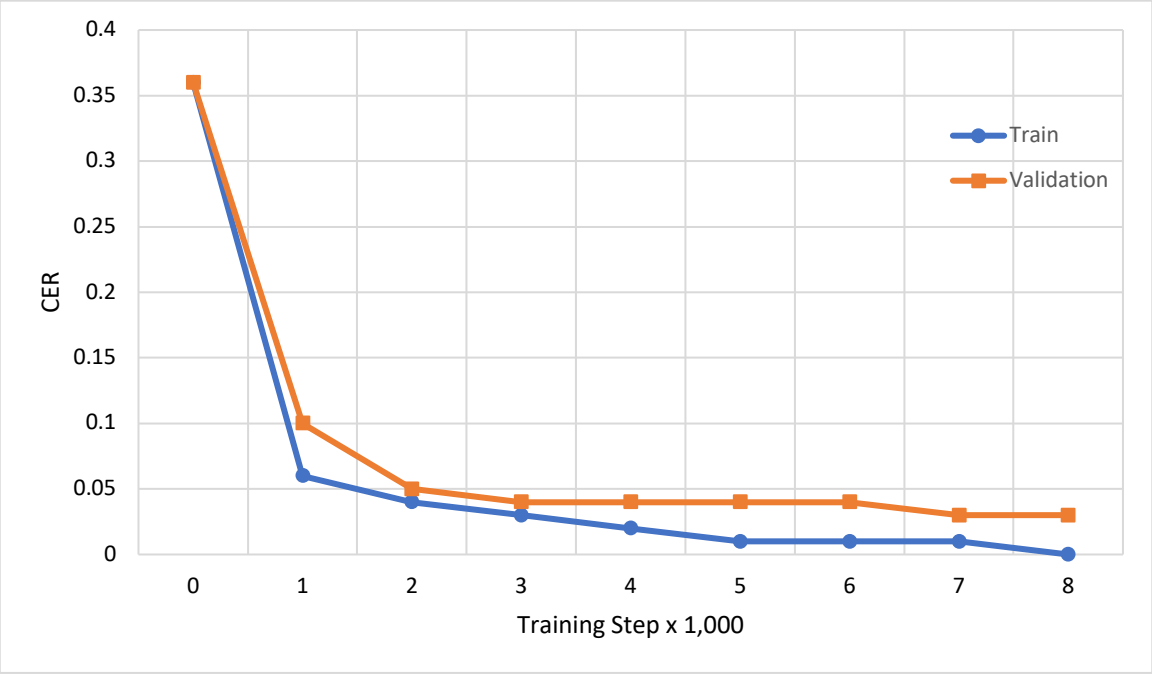

Figure 26. Training Curve with directed EIR of 2.5%

- The best CER on the validation set is reported at step 8,000 with a CER of 0.03%.
- The corresponding CER reported on the Test200 set at step 8,000 is 2.26%.

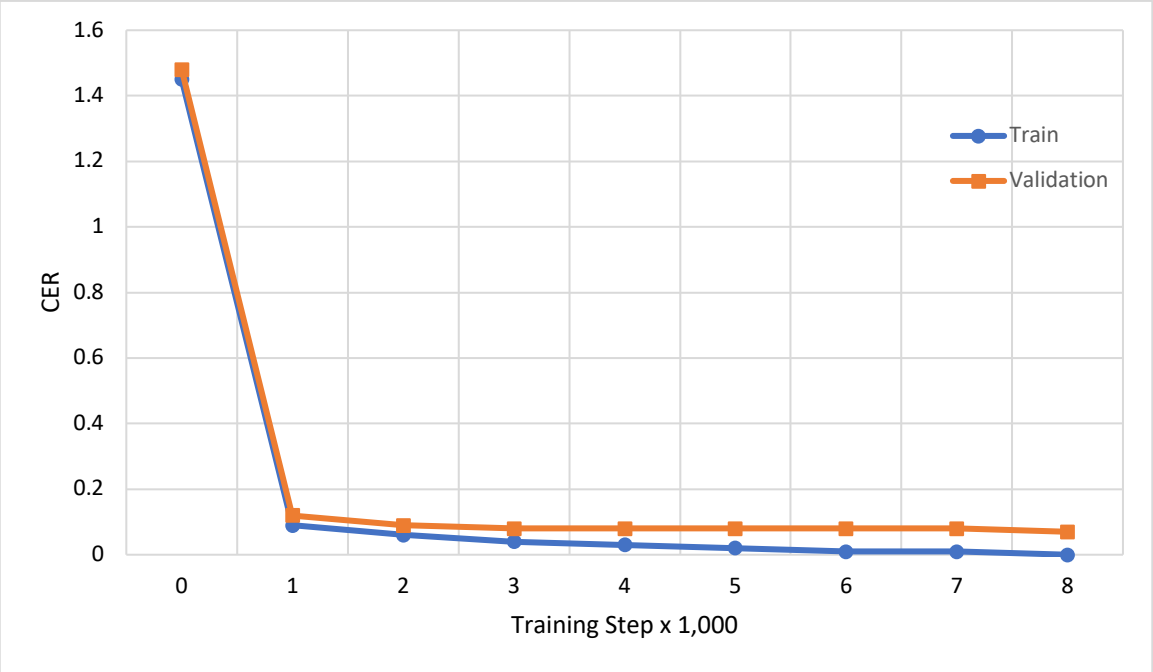

Figure 27. Training Curve with directed EIR of 10%

- The best CER on the validation set is reported at step 8,000 with a CER of 0.07%.
- The corresponding CER reported on the Test200 set at step 8,000 is 1.37%.

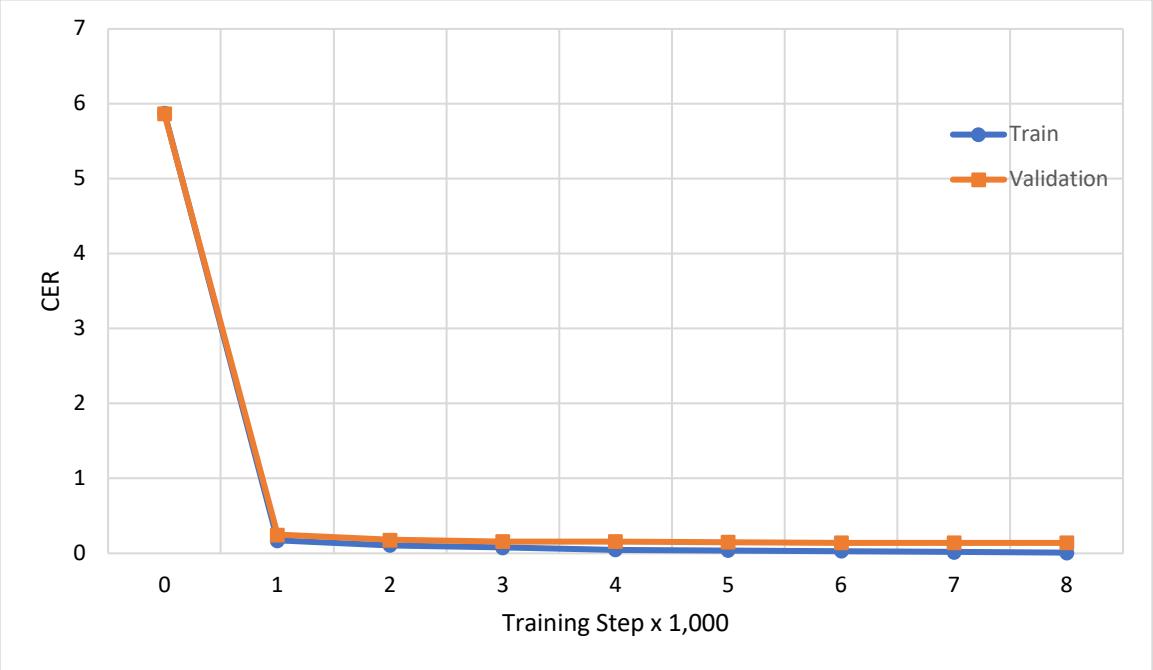

Figure 28. Training Curve with directed EIR of 40%

- The best CER on the validation set is reported at step 8,000 with a CER of 0.14%.

- The corresponding CER reported on the Test200 set at step 8,000 is 1.53%.

Figure 29 compares the different error injection rates.

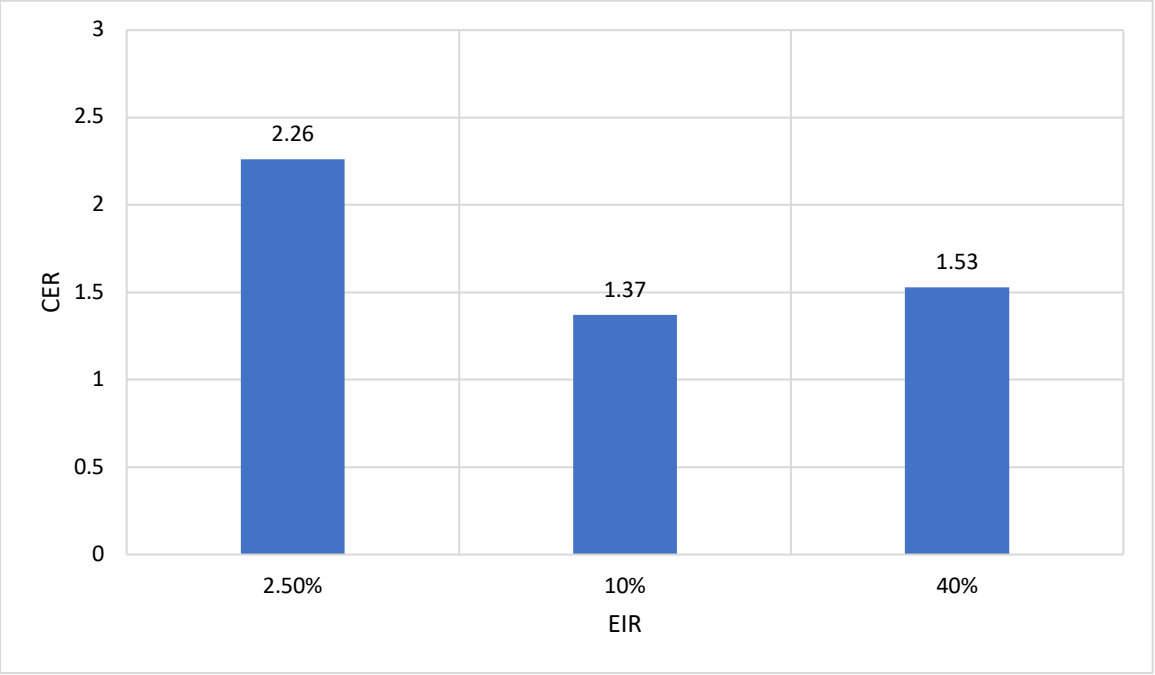

Figure 29. Comparison of Multiple Directed EIRs on Test200

We can see that an EIR of 10% performs best. Since we have the Clean-400 dataset readily available, we also test injecting it with a directed EIR of 10% and examine if that improves results on the Test200 set.

Figure 30. Training Curve with Directed EIR of 10% on Clean-400

- The best CER on the validation set is reported at step 13,000 with a CER of 0.04%.
- The corresponding CER reported on the Test200 set at step 13,000 is 1.23%.

Figure 31 compares the 10% directed error injected Clean-50 with the 10% directed error injected Clean-400.

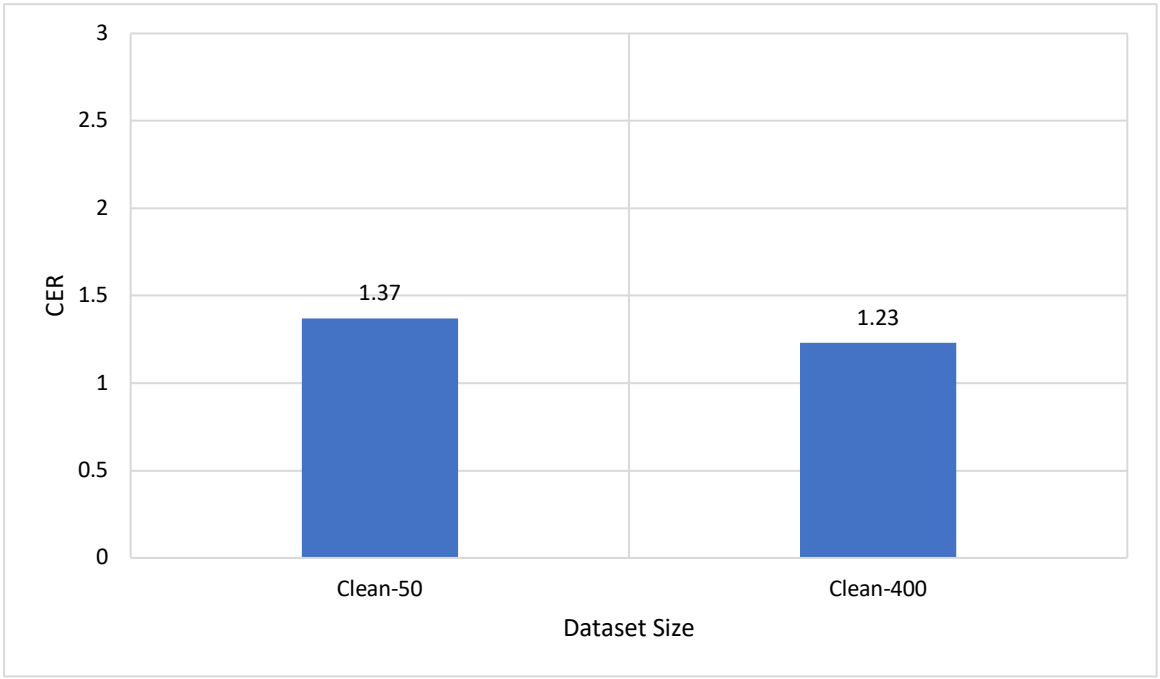

Figure 31. CER with 10% Directed EIR on Two Training Datasets

As shown in figure 31, training on the Clean-400 dataset with a directed error injection rate of 10% shows the lowest CER on the Test200 set with a CER of 1.23%.

5.7.2 General Error Injection

As outlined in Chapter 4, we will use the general error injection technique to prepare the Clean-50 dataset and then train the model using the dataset. We evaluate the results using the CER on the TSMTS. We report results for three error injection rates: 2.5%, 10%, and 40%.

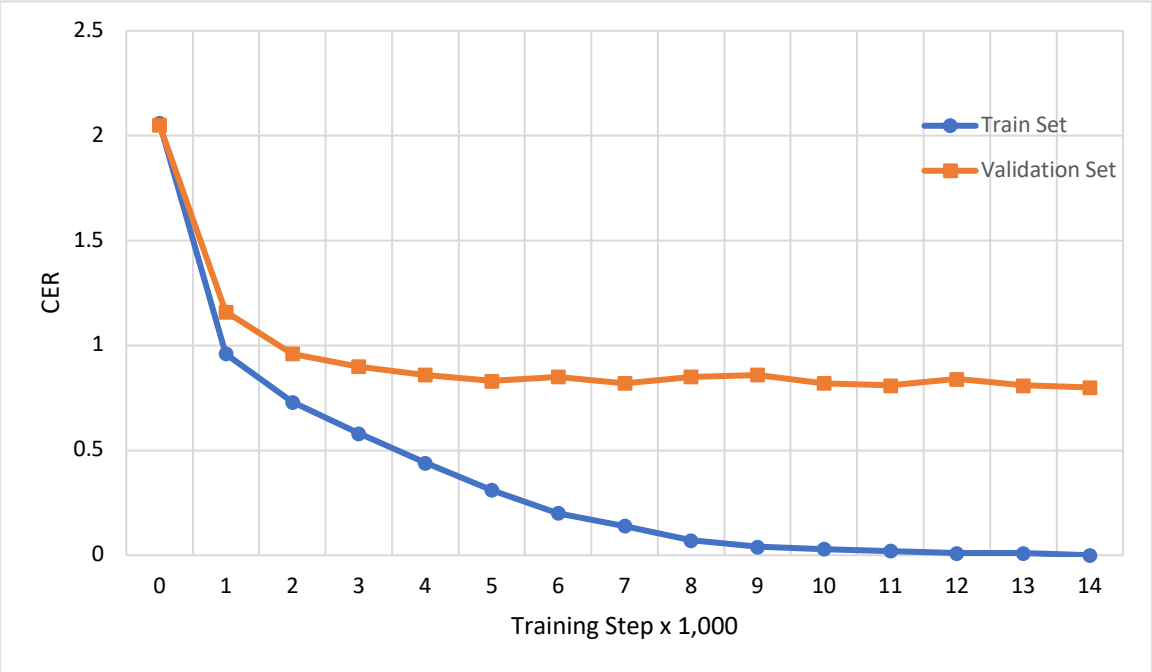

Figure 32. Training Curve with General EIR of 2.5%

- The best CER on the validation set is reported at step 14,000 with a CER of 0.80%.
- The corresponding CER reported on the TSMTS at step 14,000 is 2.09%.

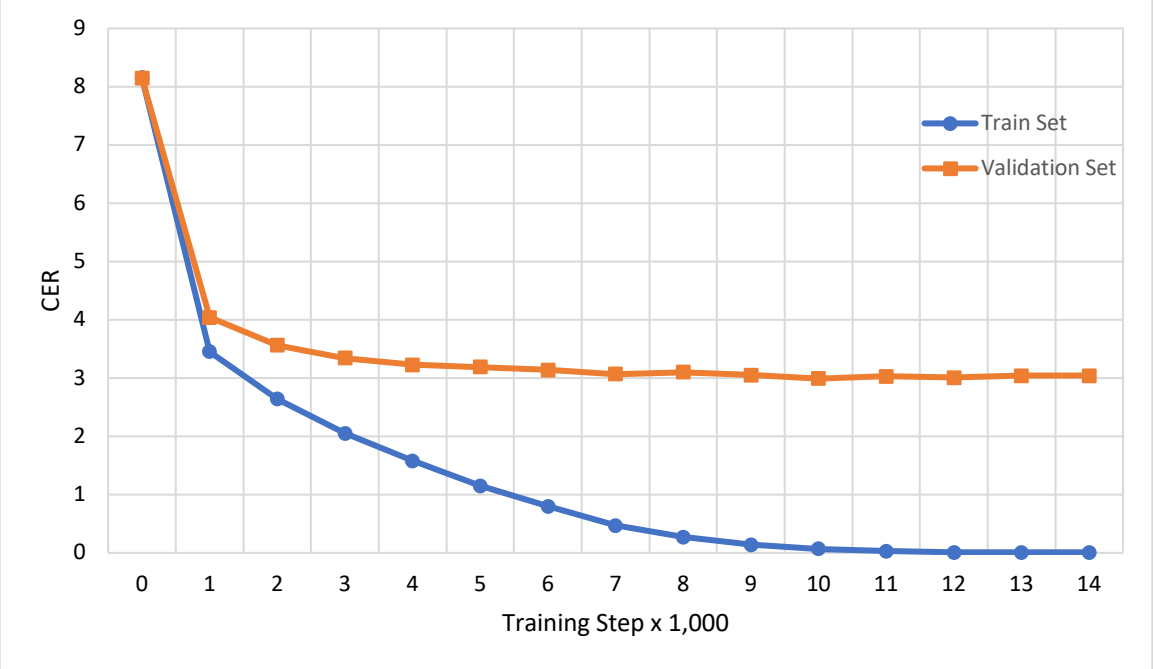

Figure 33. Training Curve with General EIR of 10%

- The best CER on the validation set is reported at step 10,000 with a CER of 2.99%.
- The corresponding CER reported on the TSMTS at step 10,000 is 1.77%.

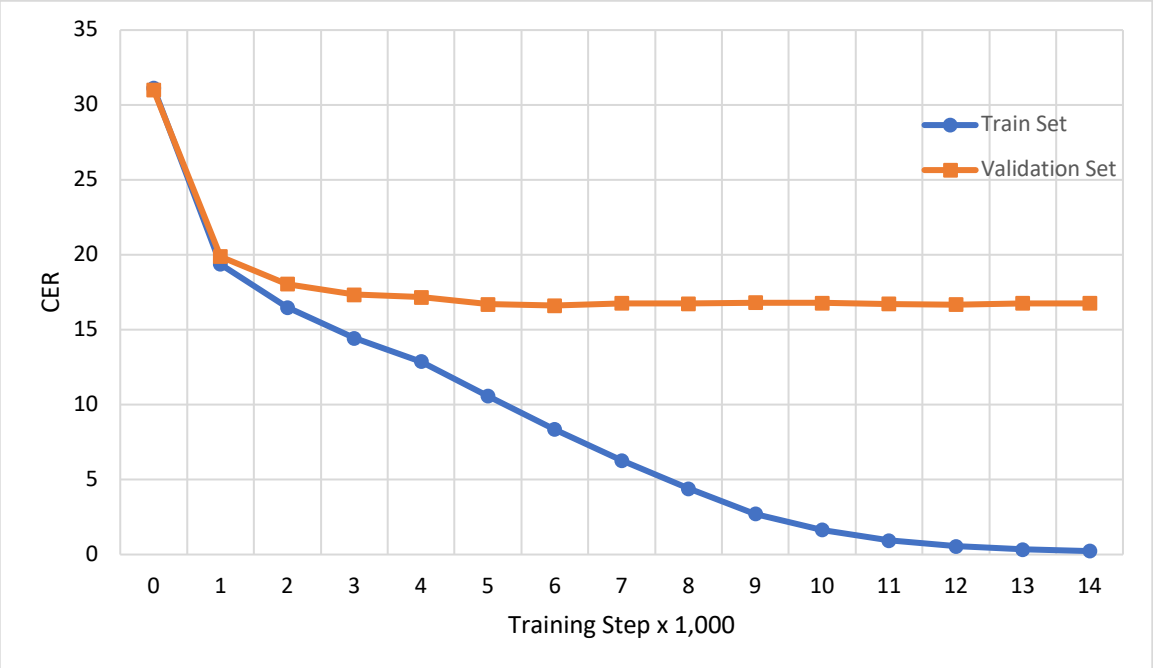

Figure 34. Training Curve with General EIR of 40%

- The best CER on the validation set is reported at step 12,000 with a CER of 16.69%.
- The corresponding CER reported on the TSMTS at step 12,000 is 2.78%.

Figure 35 compares the different error injection rates.

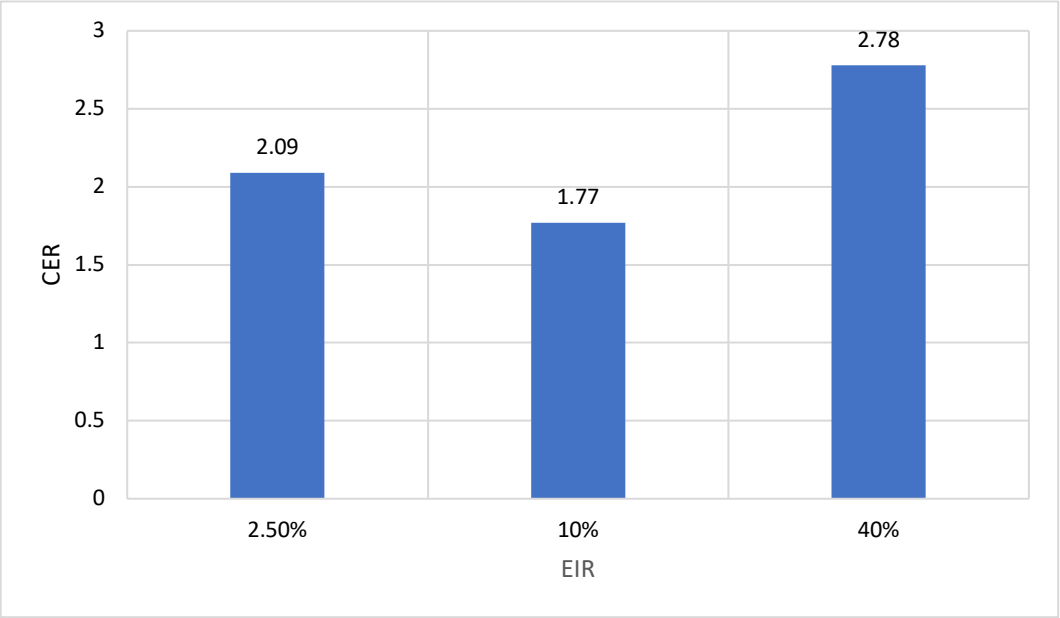

Figure 35. Comparison of Three General EIRs on the TSMTS

We can see that an EIR of 10% performs best. Since we have the Clean-400 dataset readily available, we also test injecting it with a general EIR of 10% and examine if that improves results on the TSMTS.

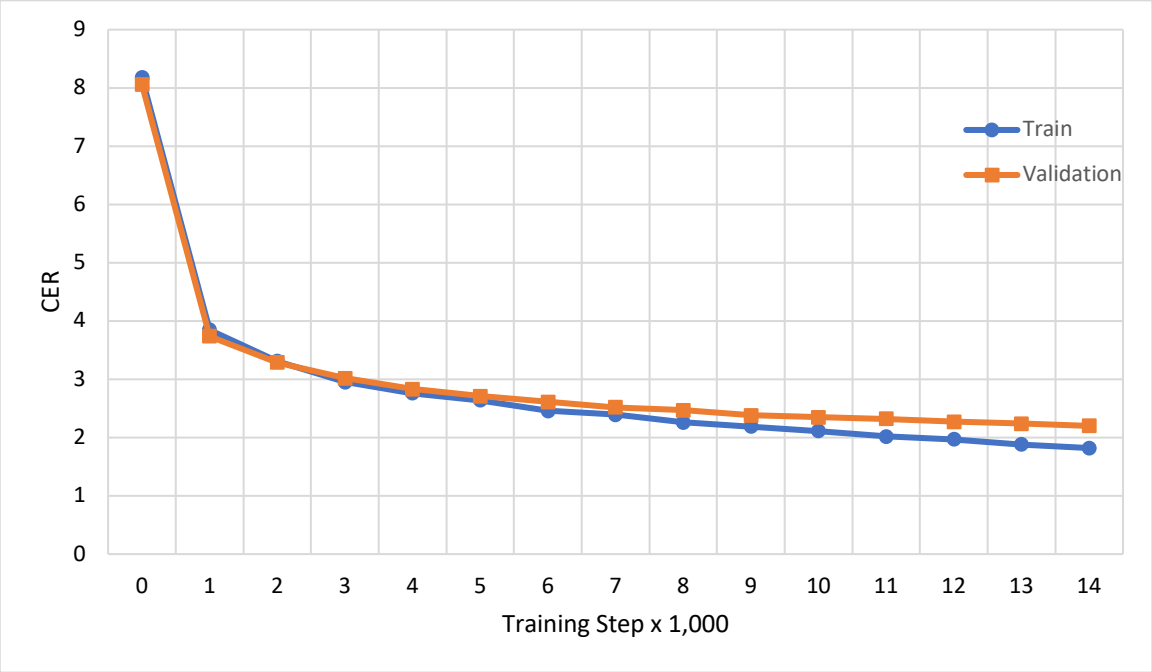

Figure 36. Training Curve with general EIR of 10% on Clean-400

- The best CER on the validation set is reported at step 14,000 with a CER of 2.20%.
- The corresponding CER reported on the TSMTS at step 14,000 is 1.28%.

Figure 37 compares the 10% general error injected Clean-50 with the 10% general error injected Clean-400.

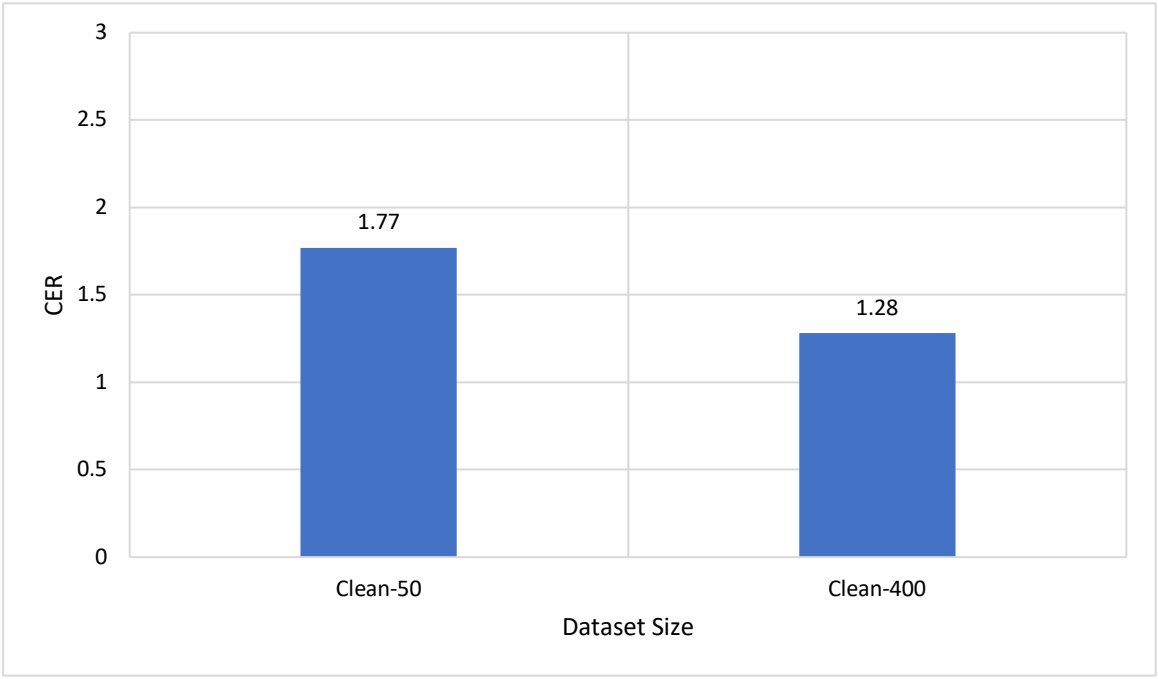

Figure 37. CER with 10% General EIR on Two Datasets

As shown in figure 37, training on the Clean-400 dataset with a general error injection rate of 10% shows the lowest CER on the TSMTS with a CER of 1.28%.

5.8 Effect of Complimentary Datasets

In the previous section, we found that the model performs best in correcting directed spelling errors when trained on a dataset with 10% directed error injection rate. We also found that the model performs best in correcting general errors when trained on a dataset with 10% general error injection rate.

In this section, we examine the performance of the model when trained on a combination of datasets. We train the model on a combination of the JODA, the 10% direct error injected Clean-50, and 10% general error injected Clean-50. We also train the model on a combination of the JODA, the 10% direct error injected Clean-400, and 10% general error injected Clean-400.

The purpose of this section is to examine if the model will be able to generalize its performance beyond the JODA dataset and be able to correct directed and general spelling mistakes.

5.8.1 Mixing JODA with Error Injected Clean-50 Datasets

In this section, we will train the tuned model (outlined in Section 5.6) on the JODA dataset along with a combination of two Clean-50 datasets. The first Clean-50 dataset is injected with a 10% direct error injection rate. The second Clean-50 dataset in injected with a 10% general error injection rate.

After training the model on the combination of datasets, the model's performance as reported on the validation set almost stabilizes after 11,000 steps of training. Figure 38 shows the BLEU score as reported on the JODA train and validation sets for the model.

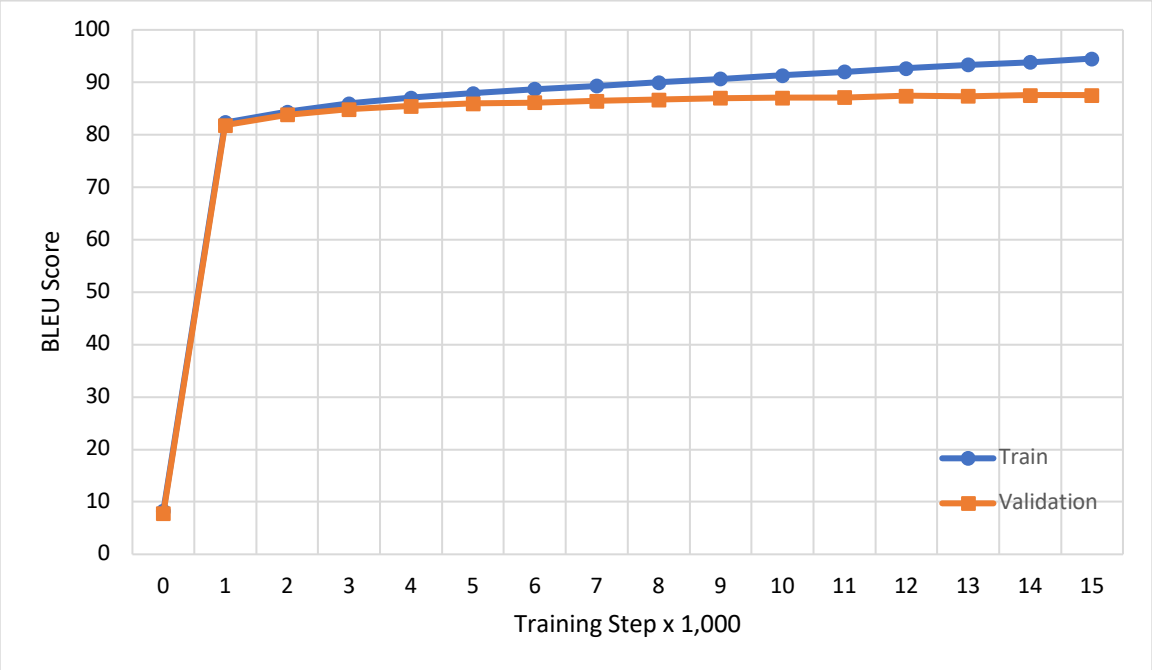

Figure 38. Training Curve with Combination of Error Injected Clean-50 datasets

- The best BLEU score on the validation set is reported at step 15,000 with a score of 87.57.
- The corresponding BLEU score reported on the test set at step 15,000 is 57.39.

We can see a slight drop in the BLEU score from 57.77 for our best model to 57.39. While the drop is insignificant, it is worth examining if the model is now able to generalize its performance beyond translating the Jordanian dialect to MSA.

To examine whether the model can generalize its performance beyond the JODA dataset, we report the CER on the Test200 and TSMTS sets for this model and for our best model trained on JODA only (outlined in Section 5.6).

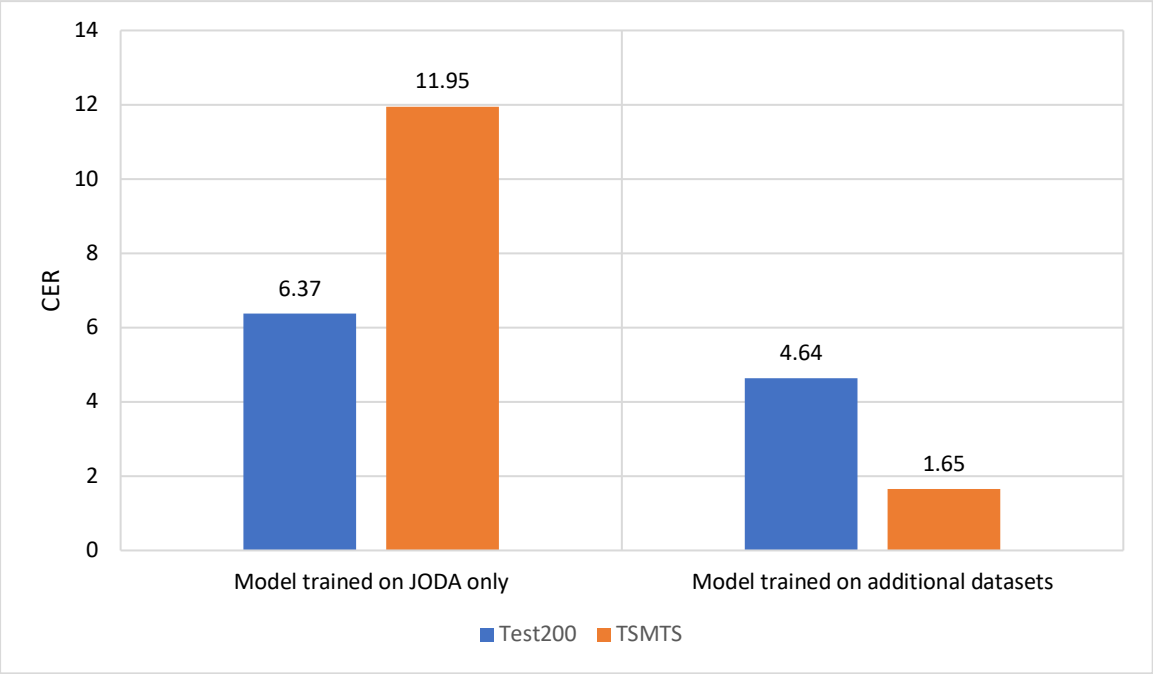

Figure 39. CER on Two Test Sets for Models Trained with Two Datasets

From figure 39 we can see that the model is now able to generalize its performance beyond the JODA dataset. The model can perform well on the JODA dataset with a 57.39 BLEU score and perform better on the Test200 and TSMTS test sets. This indicates that the model is also able to correct spelling mistakes, including general common ones.

5.8.2 Mixing JODA with Error Injected Clean-400 Datasets

In this section, we will train the tuned model (outlined in Section 5.6) on the JODA dataset along with a combination of two Clean-400 datasets. The first Clean-400 dataset is injected with a 10% direct error injection rate. The second Clean-400 dataset in injected with a 10% general error injection rate. The hypothesis is that this model will perform better than the one trained on a combination of the JODA and Clean-50 datasets.

After training the model on the combination of datasets, the model's performance as reported on the validation set almost stabilizes after 12,000 steps of training. Figure 40 shows the BLEU score as reported on the JODA train and validation sets for the model.

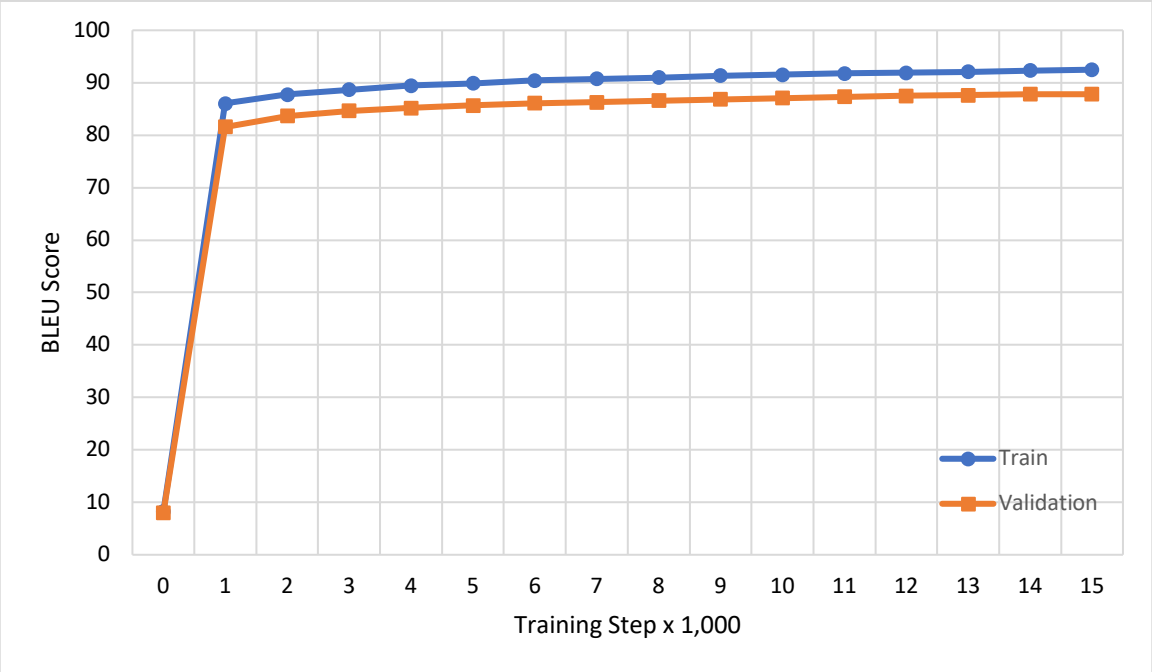

Figure 40. Training Curve with Combination of Error Injected Clean-400 datasets

- The best BLEU score on the validation set is reported at step 15,000 with a score of 87.81.
- The corresponding BLEU score reported on the test set at step 15,000 is 53.24.

As highlighted above, there is a huge drop in the BLEU score from 57.77 for our best model (as outlined in Section 5.6) and the BLEU score reported here at 53.24. Since there is a

significant drop in the BLEU score as reported on the test set, we will not move forward with this model as the main target for this thesis is to find a model able to translate the Jordanian dialect to MSA. Being able to correct directed and general errors does not outweigh the drop in performance on the JODA dataset.

We hypothesize that the reason for this drop is caused by the additional 800 thousand sequences (size of two Clean-400 datasets combined) the model is getting trained on. The JODA dataset is significantly smaller in size than the combination of Clean-400 datasets. While training on the combination of Clean-400 and JODA datasets, the model disregards what it learns from the JODA dataset in favor of attempting to capture more general trends across the remaining sequences.

In another experiment, we attempted to train the model on a combination of Clean-400s followed by training the model on the JODA dataset. The BLEU score improved to 55.38 but was still significantly less than BLEU scores reported earlier. We therefore leave it for future research to find better means for multi-phase training on JODA and Clean-400 datasets.

5.9 Comparison of Best Models

Throughout the previous sections, we experimented with different model configurations, hyperparameters, and training techniques. We started with a baseline model and finetuned our way through to the "tuned" model highlighted in Section 5.6. We were also able to generalize the model's performance beyond the JODA dataset by training the model on a combination of JODA and Clean 50 datasets as shown in Subsection 5.8.1. Table 14 summarizes results for the 3 models.

Table 14. Comparison between Different Models

	BLEU Score on JODA Test Set	**CER on Test200**	**CER on TSMTS**	**Notes**
Baseline Model	56.07	6.58%	12.41%	Initial model we started with.
Tuned Model Trained on JODA Only	57.77	6.37%	11.95%	Best reported BLEU score on the JODA test set.
Tuned Model Trained on JODA and Clean-50	57.39	4.64%	1.65%	Model generalizes its performance beyond the JODA dataset.

5.10 Time Analysis

In comparing different models, it is also crucial to factor in the time required for model training. The training methodology used throughout this thesis involved saving training checkpoints every 500 steps, allowing performance evaluation at different intervals. While checkpoints were being saved at 500 step intervals, it is important to note the overall training time that was needed to reach the best performing step on the JODA test set. This will allow

for a comparison of training time besides the comparison in results that was done in earlier sections. Figure 41 shows the total training time needed by the model to reach the performance reported on the JODA test set. Figure 41 reports the training time for the baseline model, the tuned model trained on JODA only, and the tuned model trained on JODA and a combination of Clean-50s.

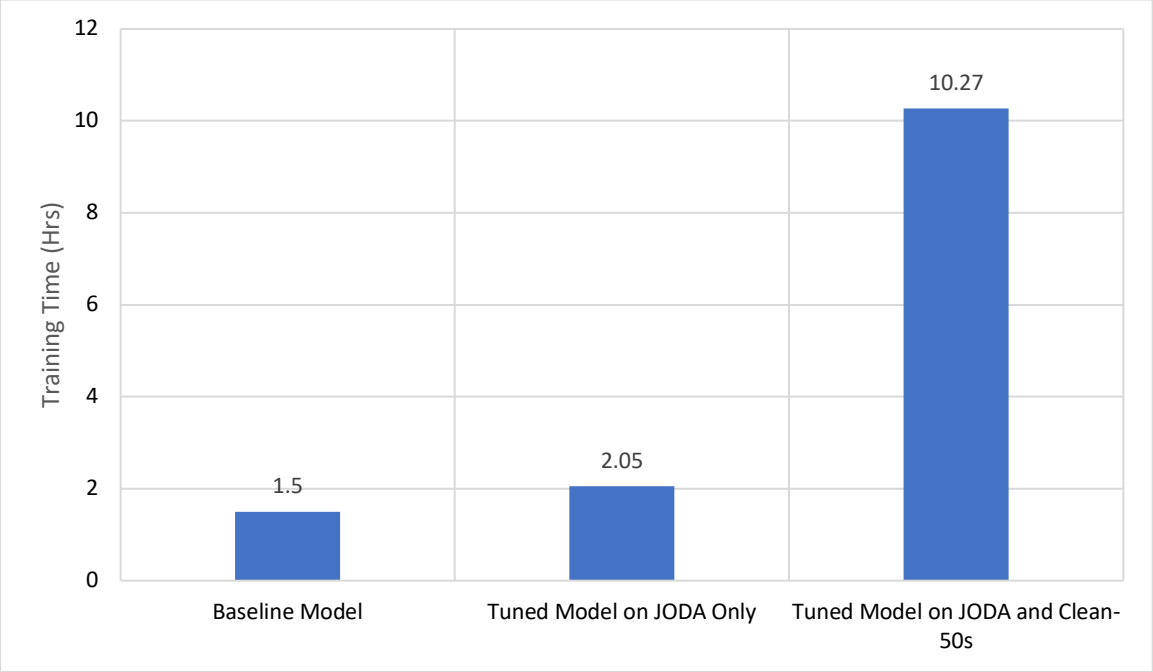

Figure 41. Training Time for Three Models

Although the tuned model trained on JODA and a combination of Clean-50s takes a longer time to reach convergence, the model can generalize its performance beyond the JODA dataset. The Baseline and tuned model trained on JODA only need less training time since they only get trained on a relatively small dataset, and hence convergence is realized quicker.

5.11 Manual Examination of Results

Throughout this thesis, we depended on the BLEU score and CER as evaluation metrics. While those two metrics are valid for evaluating translation quality, they might also under-estimate the model's performance. The Arabic language is a rich language. One sentence can be translated in many forms and can still be correct. There is an abundance of synonyms in

Arabic. While the model might have predicted a sentence in MSA correctly, the model could have been penalized due to the prediction not matching the target sequence exactly.

This thesis aims to find a model capable of translating Jordanian dialect to MSA. This thesis is not targeting translating texts to an exact match with the target sequences. For illustration, we manually audited 100 predictions made by the tuned model outlined in Section 5.6 and classified the type of errors found. In our manual auditing, the model was not penalized for using valid synonyms. We also recalculated the CER over the different JODA test set subsets to see if the value changes.

Table 15. Classification of 100 Predictions Manually Audited

Type of Prediction	Count	Examples		
		Input Sequence	Target Sequence	Model Prediction
Correct Prediction	51	بامكاني انزل لمستواك و احكي بالطريقة الهمجية الي بتحكي فيها ،	بإمكاني أن أنزل إلى مستواك وأتكلم بالطريقة الهمجية التي تتكلمين بها	بإمكاني أن أنزل إلى مستواك وأتكلم بالطريقة الهمجية التي تتكلم بها
Minor Error	40	المذيعه مش معاه ابدا ما خلتله مجال يحكي أسلوبها مزعج	المذيعة ليست معه أبدا، لم تترك له مجالا ليتكلم، أسلوبها مزعج	المذيعة ليست معه أبدا، لم تترك له مجالا ليتكلم بأسلوبها مزعج
Major Error	9	يا بنت الحلال مهو طول عمره هون شو عملك يعني	يا بنت الحلال، هو طوال عمره هنا، ماذا فعل لك إذا؟	يا بنت الحلال فهو طوال عمره هنا ماذا عمل لك إذا؟

Table 16. CER on Different JODA Test Subsets Before and After Manual Auditing

	CER on Jordanian Dialect Subset	CER on MSA Subset	Overall CER	Number of Correct Predictions
Original predictions	16.65%	5.23%	12.39%	34
Audited predictions	8.93%	2.29%	5.56%	51

As seen in the tables 15 and 16, the mode's performance is actually better than that reported by the used metrics. Even though the results reported are excellent, in practice, the model's performance is better.

To evaluate the performance of the model beyond the JODA dataset, we provide the model with a set of custom inputs and report the corresponding predictions made by the model. This set of inputs was manually curated and is not included within the JODA dataset.

Table 17. Model's Performance on Examples from Outside JODA

Input	Prediction	Classification
وين بدك تروح اليوم؟	أين ستذهب اليوم؟	Correct
شلونك يا زلمة؟	كيف حالك يا رجل؟	Correct
الجو دفا اليوم	الجو دفأ اليوم	Minor Error
قديش الساعة هسا؟	كم الساعة الآن؟	Correct
امبارح كنت تعبان كتير	البارحة كنت متعبا كثيرا	Correct
اشتقتلك يخوي	اشتقت لك يا أخي	Correct
شو وراك يا زلمة	ما وراءك يا رجل؟	Correct

Chapter Six

Conclusion and Future Work

This final chapter summarizes the thesis and draws a map for future work we hope can build on the results from this thesis.

6.1 Conclusion

This thesis investigated developing a solution for translating texts from the Jordanian dialect to MSA. The investigation involved utilizing a pre-trained ByT5 LLM. We examined two variations of the ByT5 model size, investigated hyperparameter finetuning, and trained the model with different datasets.

In this thesis, we decided to follow the same approach followed by the T5 model authors (Phakmongkol *et al*., 2021) in changing one hyperparameter at a time and building on the results we find. While this approach might miss some combinations that can achieve better scores, it provides a good tradeoff given the resources we have at our disposal.

While we started with a base-line model that achieved a BLEU score of 56.07 on the JODA test set, we were able to experiment with different hyperparameters and reach a tuned model (as highlighted in Section 5.6) that achieved a remarkable 57.77 BLEU score. While our tuned model was trained on the JODA dataset, it was unable to extend its performance beyond the dataset. We explored training the model on multiple Clean-50 datasets, which resulted in a comparable BLEU score of 57.39 while being able to perform well on the Test200 and TSMTS test sets, extending its performance beyond the JODA dataset.

The achieved BLEU scores are considered excellent. For comparison, Edman et al. (2024) used the mT5 and ByT5 models on tasks involving translating texts from German to English, English to German, Russian to English, and English to Russian, and the authors were unable to achieve a BLEU score higher than 35 in any of the tasks.

Training the model on a combination of the JODA and Clean-50 datasets helped extend the model's performance beyond translation. By training the model on error-injected datasets, the model was able to capture additional behaviors. The error-injected Clean-50 datasets provided the model with sufficient data to extend its performance without significantly downgrading the BLEU score on the JODA dataset.

The models we outline in this thesis can be used for translating the Jordanian dialect to MSA, and can also be used for correcting general and directed errors. We hope this model can benefit our local community pave the way for further research within our region.

6.2 Future Work

The resources we have at our disposal are limited. There is a limited number of experiments that we can conduct. The amount of time and computing power we had was also limited. We hope that this research will pave the way for more research in this field, focusing on translating the Jordanian dialect and other local dialects to MSA. This will help bridge the gap between the Arabic language speakers. More research in this field will also provide more momentum for research focused on the Arabic language.

There are other model sizes that we were not able to explore. We saw how the "base" model performed better than the "small" model. The ByT5 model comes in other bigger sizes that are worth exploring. We hope we will be able to conduct further research with larger models in the future.

We hope that the JODA dataset, introduced for the first time, can be utilized in more research. We conducted a handful of experiments with the dataset and saw its ability for training LLMs. We hope that more people will be able to use the dataset and explore it in ways that were not explored in this thesis.

REFERENCES

Abandah, G., Suyyagh, A., & Khedher, M. Z. (2022a). Correcting Arabic Soft Spelling Mistakes using BiLSTM-based Machine Learning. **International Journal of Advanced Computer Science and Applications**, 13(5). https://doi.org/10.14569/ijacsa.2022.0130594

Abandah, G. A., Suyyagh, A. E., & Abdel-Majeed, M. R. (2022b). Transfer learning and multi-phase training for accurate diacritization of Arabic poetry. **Journal of King Saud University. Computer and Information Sciences**/Mağalaẗ Ğam'aẗ Al-malīk Saud : Ùlm Al-ḥasib Wa Al-ma'lumat, 34(6), 3744–3757. https://doi.org/10.1016/j.jksuci.2022.04.005

Abdellah, Y., Lhoussain, A. S., Hicham, G., & Mohamed, N. (2020). Spelling correction for the Arabic language space deletion errors-. **Procedia Computer Science**, 177, 568–574. https://doi.org/10.1016/j.procs.2020.10.080

Al-Ibrahim, R., & Duwairi, R. M. (2020). Neural Machine Translation from Jordanian Dialect to Modern Standard Arabic. **11th International Conference on Information and Communication Systems (ICICS)**, 173–178. https://doi.org/10.1109/icics49469.2020.239505

Al-Rfooh, B., Abandah, G., & Al-Rfou, R. (2023). Fine-Tashkeel: Finetuning byte-level models for accurate Arabic text diacritization.

AlShenaifi, N., AlNefie, R., Al-Yahya, M., & Al-Khalifa, H. (2015). Arib@QALB-2015 Shared Task: A Hybrid Cascade Model for Arabic Spelling Error Detection and Correction. **Proceedings of the Second Workshop on Arabic Natural Language Processing**, 127–132. https://doi.org/10.18653/v1/w15-3214

Analytics Vidhya. (2021, July 9). **In-depth explanation of recurrent neural network**. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/07/in-depth-explanation-of-recurrent-neural-network/

Attia, M., Pecina, P., Samih, Y., Shaalan, K., & Van Genabith, J. (2015). Arabic spelling error detection and correction. **Natural Language Engineering**, 22(5), 751–773. https://doi.org/10.1017/s1351324915000030

Boyle, D., & Kalita, J. (2023). Spatiotemporal Transformer for stock movement prediction. **Proceedings of the Conference on Neural Information Processing Systems (NeurIPS 2023)**.

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of deep bidirectional transformers for language understanding.

Edman, L., Sarti, G., Toral, A., Van Noord, G., & Bisazza, A. (2024). Are character-level translations worth the wait? Comparing BYT5 and MT5 for machine translation. **Transactions of the Association for Computational Linguistics**, 12, 392–410. https://doi.org/10.1162/tacl_a_00651

Ethnologue. (n.d.). Arabic. Ethnologue: Languages of the World. Retrieved 2023, from https://www.ethnologue.com/language/arb/

Fadel, A., Oueslati, O., Gahbiche, H., & Shafik, R. (2019). Neural Arabic text diacritization: State of the art results and a novel approach for machine translation. **Proceedings of the 6th Workshop on Asian Translation.**

Gehring, J., Auli, M., Grangier, D., Yarats, D., & Dauphin, Y. N. (2017). Convolutional sequence to sequence learning. In **Proceedings of the 34th International Conference on Machine Learning** (Vol. 70, pp. 1243-1252).

Gehring, J., Auli, M., Grangier, D., Yarats, D., & Dauphin, Y. N. (2017). Convolutional sequence to sequence learning. In **Proceedings of the 34th International Conference on Machine Learning** - Volume 70 (ICML'17) (pp. 1243–1252). JMLR.org.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition** (pp. 770-778).

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. **Neural Computation**, 9(8), 1735-1780.

Jentoft, M. (2023). GEC with byte-level language models **(Master's thesis).**

Karim, A. A., & Abandah, G. (2021). On the Training of Deep Neural Networks for Automatic Arabic-Text Diacritization. **International Journal of Advanced Computer Science and Applications**, 12(8). https://doi.org/10.14569/ijacsa.2021.0120832

Kirov, C., Johny, C., Katanova, A., Gutkin, A., & Roark, B. (2024). Context-aware transliteration of Romanized South Asian languages. **Computational Linguistics**, 1-61.

Li, Y.-H., Harfiya, L. N., Purwandari, K., & Lin, Y.-D. (2020). Real-time cuffless continuous blood pressure estimation using deep learning model. **Sensors**, *20*(19), 5606. https://doi.org/10.3390/s20195606

Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., & Neubig, G. (2023). Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. **ACM Computing Surveys**, 55(9), 1-35.

Mubarak, H., & 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).

Phakmongkol, P., & Vateekul, P. (2021). Enhance text-to-text transfer transformer with generated questions for Thai question answering. **Applied Sciences**, 11(21), 10267. https://doi.org/10.3390/app112110267

Rajan, R., Sivan, R., Ravindran, R., & Soman, K. (2009). Rule-based machine translation from English to Malayalam. In **2009 International Conference on Advances in Computing, Control, and Telecommunication Technologies**. https://doi.org/10.1109/act.2009.113

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., & Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. **Journal of Machine Learning Research**, *21*(1), Article 140.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. **Nature**, 323(6088), 533-536.

Schuster, M., & Paliwal, K. (1997). Bidirectional recurrent neural networks. **IEEE Transactions on Signal Processing**, 45, 2673-2681. https://doi.org/10.1109/78.650093

Sherstinsky, A. (2020). Fundamentals of Recurrent Neural Network (RNN) and long short-term memory (LSTM) network. **Physica D: Nonlinear Phenomena**, *404*, 132306. https://doi.org/10.1016/j.physd.2019.132306

Singh, S. P., Kumar, A., Darbari, H., & Jain, S. (2017, July). Machine translation using deep learning: An overview. **Proceedings of the 2017 International Conference on Computer, Communications and Electronics**.

Sreelekha, S., Bhattacharyya, P., & Malathi, D. (2017). Statistical vs. rule-based machine translation: A comparative study on Indian languages. In **Advances in Intelligent Systems and Computing** (pp. 663–675). https://doi.org/10.1007/978-981-10-5520-1_59

Sutskever, I. (2020). Sequence to sequence learning with neural networks. https://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf.

Xue, L., Constant, N., Roberts, A., Kale, M., Al-Rfou, R., Siddhant, A., Barua, A., & Raffel, C. (2021). mT5: A Massively Multilingual Pre-trained Text-to-Text Transformer (pp. 483–498). https://aclanthology.org/2021.naacl-main.41.pdf

Xue, L., Barua, A., Constant, N., Al-Rfou, 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, 291–306. https://doi.org/10.1162/tacl_a_00461

Watson, J. (2011). Arabic dialects (general article). In S. Weninger, G. Khan, M. Streck, & J. C. E. Watson (Eds.), **The Semitic Languages: An international handbook** (pp. 851-896). Walter de Gruyter.

Weng, R., Huang, S., Zheng, Z., Dai, X., & Chen, J. (2017). Neural machine translation with word predictions. In **EMNLP 2017 - Conference on Empirical Methods in Natural Language Processing, Proceedings** (pp. 136–145).

ترجمة النصوص من اللهجة الأردنية إلى اللغة العربية الفصيحة عن طريق نماذج اللغة الكبيرة

إعداد

معاذ رابح خليل

المشرف

الأستاذ الدكتور غيث علي عبندة

ملخص

يتطلب ترجمة النصوص من اللهجة الأردنية إلى اللغة العربية الفصحى قدرًا كبيرًا من الجهد اليدوي. من الصعب جدًا تطوير أنظمة تعتمد على القواعد مسبقة التعريف، خاصة عند التعامل مع لغة مثل اللغة العربية التي تزخر بالقواعد الكتابية والإملائية. توفر النماذج اللغوية الكبيرة حلولًا عملية لهذه المهمة، وبالأخص تلك التي تتعامل مع الأحرف لا الكلمات، حيث تقدم حلولاً واعدة لأتمتة ترجمة النصوص من اللهجة الأردنية إلى اللغة العربية الفصحى.

تهدف هذه الأطروحة إلى معالجة التحديات التي تواجه هذا المجال قليل الموارد. كما نقدم لأول مرة مجموعة بيانات (JODA)، ونجري عليها مجموعة من التجارب التي تعمل على تحسين أداء النموذج وتقديم أفضل ترجمة ممكنة للنصوص من اللهجة الأردنية إلى اللغة العربية الفصحى. لا يقتصر هذا البحث على تدريب النموذج على الترجمة فقط، بل نستكشف استخدام مجموعات بيانات إضافية في التدريب لتعزيز أداء النموذج بما يتجاوز الترجمة إلى تصحيح الأخطاء الإملائية المباشرة والعامة. نستفيد من معالجة النموذج على مستوى الأحرف وميزاته في تجزئة النصوص، إلى جانب توليد البيانات الاصطناعية، للتغلب على ندرة بيانات التدريب.

تبدأ الدراسة بتدريب نموذج أولي على مجموعة بيانات (JODA)، ثم نقوم بتعديل مختلف المتغيرات من أجل تحسين أداء النموذج وتحقيق نتائج (BLEU) فريدة من نوعها. نقوم بعدها بتدريب النموذج على مجموعة بيانات إضافية تتكون من (Clean 50) و (Clean 400)، لتمكين النموذج من القدرة على تصحيح الأخطاء اللغوية المباشرة والغير مباشرة.

نقوم باستعمال نماذج بأحجام مختلفة، من ضمنها الحجم الصغير ومتوسط الحجم، ونقوم باستعمال محسنات أداء مختلفة، من ضمنها (Adam) و (Adafactor) لتحسين أداء النموذج. تظهر النتائج أن النموذج متوسط الحجم الذي يستعمل محسن أداء (Adam) يحقق أفضل نتيجة (BLEU) مقدارها ٥٧.٧٧ باستعمال مجموعة بيانات (JODA). كما تظهر النتائج أن التدريب الإضافي على مجموعات بيانات أخرى يقلل من نتيجة ال (BLEU) قليلاً، لكنه يحقق نتائج جيدة جداً على مجموعة بيانات (Test200) ب (CER) مقداره ٤.٦٤٪ ونتائج جيدة جداً على مجموعة بيانات (TSMTS) ب (CER) مقداره ١.٦٥٪ مما يظهر إمكانية النموذج على ألا يقتصر أداءه على الترجمة فقط.