نموذج التفويض

أنا روزان علي محمد يونس، أفوض الجامعة الأردنية بتزويد نسخ من رسالتي /أطروحتي للمكتبات، أو المؤسسات، أو الهيئات، أو الأشخاص عند طلبهم حسب التعليمات النافذة ف الجامعة.

التوقيع:

التاريخ: 2024/6/23

The University of Jordan

Authorization Form

I, Rozan Ali Mohammed Younis, authorize the University of Jordan to supply copies of my Thesis/ Dissertation to libraries or establishments or individuals on request, according to the University of Jordan regulations.

Signature:

Date: 23/6/2024

AUTOMATIC DIACRITIZATION OF ARABIC TEXT AND POETRY USING PRE-TRAINED BYTE-TO-BYTE LANGUAGE MODELS AND MULTI-PHASE TRAINING

By

Rozan Ali Aljaber

Supervisor

Dr. Gheith Abandah, Prof.

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

School of Graduate Studies
The University of Jordan

June, 2024

COMMITTEE DECISION

This Thesis/Dissertation (Automatic Diacritization Of Arabic Text And Poetry Using Pre-Trained Byte-To-Byte Language Models And Multi-Phase Training) was Successfully Defended and Approved on

Examination Committee	**Signature**
Dr. Gheith Abandah (Supervisor) Prof. of Computer Engineering	----------
Dr. Waleed Dweik (Member) Associate Prof. of Computer Engineering	----------
Dr. Adham Al-Sharkawi (Member) Associate Prof. of Mechatronics Engineering	----------
Dr. Ahmad Al-Jaafreh (Member) Prof. of Computer Engineering Tafila Technical University	----------

DEDICATION

I dedicate this thesis to my beloved parents, Khetam and Ali, who have always been there for me with their love and encouragement. Their support has guided me through every step of my academic journey.

To my husband, Yazan, your patience and belief in me have motivated me to pursue this endeavor with determination.

To my precious daughter, Sofia, your laughter and curiosity remind me of the importance of my scholarly pursuits and inspire me to strive for a better future.

To my dear sisters, Rawan, Marah, and Farah, your constant encouragement has given me strength to achieve my dreams.

To my brothers, Emad and Mohammed, your support has made my academic journey richer.

To my dear Marwa al-Tahrawi, your insistence on walking hand in hand with me to complete my journey in the master's degree

And to all my friends who have encouraged and supported me through the tough times, I am grateful for your unwavering friendship.

AKNOWLEDGEMENT

My biggest gratitude to my advisor, Professor Gheith Abandah for helping me so much with this thesis. Your advice, knowledge, and constant support have been really important to me. You've not only helped me learn a lot academically but also inspired me to do my best.

Your feedback on my work has been really helpful. You've always been patient and encouraging, which has pushed me to work harder and learn more. I really appreciate all the time and effort you've put into helping me succeed. I want to thank my friend Malak Al-Smadi from the bottom of my heart for her support and advising me on how to deal with Google Cloud. Also, I want to thank my friend Dana Mehdawi for assisting me in proofreading my thesis and correcting it grammatically. You helped me improve the quality of the thesis and ensured it adhered to the linguistic rules correctly. Thank you for your efforts and continuous support.

Table of Contents

COMMITTEE DECISION ..... **ii**

DEDICATION ..... **iii**

AKNOWLEDGEMENT ..... **iv**

Table of Contents ..... **v**

LIST OF TABLES ..... **vii**

LIST OF FIGURES ..... **viii**

LIST OF ABBREVIATIONS ..... **ix**

ABSTRACT ..... **xi**

Chapter One **1**

Introduction **1**

1.1 Background *2*

1.2 Thesis Problem *5*

1.3 Importance of Arabic Diacritization *5*

1.4 Thesis Objectives *6*

1.5 Thesis Questions and Assumptions *7*

1.6 Thesis Contribution *7*

1.7 Thesis Methodology *8*

1.8 Thesis Outline *8*

Chapter Two **10**

Literature Review **10**

2.1 Introduction *10*

2.2 Traditional Approaches to Diacritize Arabic Text and Poetry *10*

2.3 Machine Learning Approaches to Diacritize Arabic Text *11*

2.4 Machine Learning Approaches to Diacritize Arabic Poetry *14*

2.5 Exploring Multi-stage Neural Network Models for Diacritization *15*

2.6 Sequence-to-Sequence Transformer Models *16*

Chapter Three **18**

Sequence Transcription Approaches **18**

3.1 Introduction *18*
3.2 RNN *18*
3.3 Transformers *22*
3.4 Transfer model *26*
3.5 Text-To-Text Transfer Transformer *27*
3.6 Pre-Trained mT5 ..... *30*
3.7 mC4 ..... *30*
3.8 Pre-Trained ByT5 ..... *31*

Chapter four **34**

Datasets Preparations and Experiments **34**

4.1 Clean-400 ..... *34*
4.2 Arabic Poem Comprehensive Dataset (APCD2) *35*
4.3 Experiments *38*

Chapter Five **45**

Experiments and Results **45**

5.1 Introduction *45*
5.3 Results *47*
5.4 Effect of Model Size *50*
5.5 Time Analysis of Models *51*
5.5 Comparison *52*
5.5 Discussion *54*
5.6 Predictions' Errors *58*

Chapter Six **60**

Conclusion and Future Work **60**

6.1 Conclusion *60*
6.2 Future Work *61*

References **62**

ملخص **68**

LIST OF TABLES

Table 1. Illustration the Arabic diacritical marks with s example 3

Table 2.Types of Arabic Poetic Meters and Their Demonstrations 4

Table 3.Impact of Diacritical Marks on Word Meaning 5

Table 4.Prior Research on Arabic Text Diacritization 14

Table 5.Prior Research on Arabic Poetry Diacritization 15

Table 6. Clean-400 Corpus Characteristics after wrapping 35

Table 7. Poetry Meter Class Distribution and Relative Verses Ratio 36

Table 8. A Poetry Example of Input and Target 37

Table 9. Characteristics Of the APCD2 Dataset 38

Table 10. ByT5 Model Specifications 39

Table 11. Colab Pro Plus Platform Computing Environment 41

Table 12. Predicted Diacritics and Their IDs in the ByT5 Model 42

Table 13. Comparison Of Experiments Trained Using 47

Table 14. Diacritics Restoration Results Comparison 54

Table 15. An Example of Prose Model's Predictions 58

Table 16. Examples of Prose Model's Predictions 59

LIST OF FIGURES

Figure 1.Visual Representation of RNN Architecture (Zhang *et al*. 2019) 19

Figure 2.Visual Representation of BiLSTM Architecture (Madhfar *et al*., 2020) 21

Figure 3.*Transformers* Structure (Boyle & Kalita. 2023) 22

Figure 4.Categorizing *Transformers* Based on Architecture Types (Nassiri and Akhloufi. 2023) 27

Figure 5.T5 Performance Across General NLP Tasks (Raffel *et al*., 2020) 27

Figure 6. Comparing Pre-training Methodology and Network Structure: mT5 vs. ByT5 (Xue *et al*., 2022) 32

Figure 7. The Distribution of Poetry Eras in the Dataset 37

Figure 8. The Accuracy Throughout the Training of The Small Diacritization Prose Model. 46

Figure 9. The DER Of the Poetry Diacritization Model Across Validation Sets for The Three Datasets Throughout Training ..... **Error! Bookmark not defined.**

Figure 10. The Accuracy Throughout the Training of The Base Diacritization Prose Model. ..... **Error! Bookmark not defined.**

Figure 11. The Effect of Model Size on Different Tasks 50

Figure 12. Analysis of Training Time for Diacritization Models 51

Figure 13. Impact of Different Learning Rates on DER During DS2 Training 55

Figure 14. Impact of Different Train Batch Size on DER During DS2 Training 56

Figure 15. Impact of Different Maximum Sequence Lenghts on DER During DS2 Training 57

LIST OF ABBREVIATIONS

AD	Arabic Diacritization
AI	Artificial Intelligence
APCD	Arabic Poem Comprehensive Dataset
ASR	Automatic Speech Recognition
AWD-LSTM	Asynchronous Stochastic Gradient Descent Weight-Dropped LSTM
BERT	Bidirectional Encoder Representations from Transformers
BiLSTM	Bidirectional Long-Short Term Memory
BNG	Block-Normalized Gradient
BRATS	Brain Tumor Segmentation
ByT5	Byte-to-Byte T5
C4	Colossal Clean Crawled Corpus
CA	Classical Arabic
CBHG	Convolutional Block Attention Module and Highway Network
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CRF	Conditional Random Field
DER	Diacritic Error Rate
FFNN	Feed-Forward Neural Networks
GCP	Google Cloud Platform
GEC	Grammatical Error Correction
HG	High Grade

HMMs	Hidden Markov Models
LG	Low Grade
LSTM	Long Short-Term Memory
MSA	Modern Standard Arabic
mT5	Multilingual T5
NLP	Natural Language Processing
RNNs	Recurrent Neural Networks
T5	Text-to-Text Transfer *Transformer*
TP	Tashkeela Processed dataset
TPU	Tensor Processing Unit
VGG-16	Visual Geometry Group 16
WER	Word Error Rate

AUTOMATIC DIACRITIZATION OF ARABIC TEXT AND POETRY USING PRE-TRAINED BYTE-TO-BYTE LANGUAGE MODELS AND MULTI-PHASE TRAINING

By

Rozan Ali Younis

Supervisor

Dr. Gheith Abandah, Prof.

ABSTRACT

Manual diacritization demands extensive time and a profound comprehension of Arabic grammar and morphology. While current automatic systems often suffer from accuracy and efficiency issues and require substantial training data and computational resources. The byte-to-byte model (ByT5), a variant of text-to-text transfer *transformers*, emerges as a promising solution for automating Arabic text and poetry diacritization within natural language processing (NLP). This thesis aims to address these challenges, particularly focusing on poetry, where available undiacritized or partially diacritized poetry datasets pose a significant hurdle. Leveraging the model's byte-level processing and auto-tokenization features, along with a multi-phase training strategy, is essential to overcome the scarcity of training data.

The study begins with training on the Clean-400 dataset to create a prose diacritization model and evaluating it on the Tashkeela test set. Then, the poetry diacritization model is trained on available DS1 and DS2 datasets, leveraging learned representations and weights from the prose model. The training phases on the DS1 and DS2 datasets aim to improve prediction accuracy for fully diacritized poetry.

The ByT5 model variants, small and base, along with optimizers Adam and Adafactor, are investigated to enhance model performance. Results show that the base model with Adam optimizer performs best for text diacritization, while the small model with Adam optimizer excels for poetry. The proposed solution achieves a diacritization error rate (DER) of 0.86% and a word error rate (WER) of 2.64% for Arabic prose. However, automating Arabic poetry diacritization presents additional challenges, resulting in a DER of 3.28% and a WER of 10.74%.

Chapter One

Introduction

Arabic is the native language of millions of people and is used by Muslims worldwide in prayers and religious recitations. Both classical Arabic (CA) and modern standard Arabic (MSA) are the two main types of Arabic text. The Qur'an and other antique literature and poetry are written in CA. Currently, Arabic countries most commonly use MSA for media, education, news, and formal speeches. Unlike CA, the majority of MSA texts either have incomplete or no diacritical marks, which are orthographic marks that affect the sound of letters and the meaning of words and help readers understand and decode written text.

It is difficult for children and non-native speakers to remember the specific meaning and pronunciation without diacritization, even if native speakers use context to decipher words' meaning and pronunciation. Therefore, Arabic texts and poetry must be available with diacritical marks to enable this particular group, in particular, to understand and read the Arabic language.

Diacritizing Arabic text and poetry manually is challenging due to the complex script, the unpredictability of diacritics, and the need to follow poetic standards. The process is difficult, especially for large texts or elaborate poetic creations. The presence of these issues has increased the demand for automatic diacritization solutions. Many researchers have worked on researching and developing automatic techniques for diacritizing Arabic prose and poetry.

Several diacritization techniques were devised to enable the automated processing of Arabic prose and poetry. Automated poetry diacritization is complex compared to prose. Poets usually use words and phrases that fit the rhyme system. There is a saying known from the past that “a poet can do what others cannot” that is, some poets can do

what prose writers cannot. Additionally, the poet sometimes goes beyond grammatical and morphological rules. Moreover, the automatic diacritization of Arab poetry using deep learning is a difficult problem because of the electronic availability of Arab poetry, either non-diacritized or partial-diacritized. According to Karim and Abandah (2021), in order to achieve correct diacritization, the existing dataset of annotated Arabic poetry is not enough, and a training dataset that is five times larger is necessary. All of these reasons make diacritizing Arabic poetry much more challenging than diacritizing Arabic prose.

This study presents a precise automated diacritization method that utilizes deep machine learning, a byte-to-byte transfer *transformer* methodology, and a multi-phase training strategy.

1.1 Background

The Arabic language is one of the best languages in the world, and it's complete and rich. That's why many people from different nations throughout history have a great interest in learning it. Arabic consists of 28 letters, and each letter has a specific pronunciation depending on its position in a word. Therefore, it was necessary to introduce marks to accurately indicate how these letters are pronounced.

Many people made mistakes in pronouncing words, especially when reading the Quran during the Umayyad period. This led Abu al-Aswad al-Du'ali to invent eight diacritical symbols, commonly referred to as “Tashkeel” or “Harakat”, which serve the purpose of indicating short vowels to clarify the pronunciation of characters.

The Arabic letter can be written with no diacritics, a single diacritic, or two diacritics. Table 1 presents the Arabic diacritical marks along with their Unicode values, the translated Arabic name, the associated English sound description, an image of the letter "S" with diacritics, and the resulting phonetic sound.

Table 1. Illustration the Arabic Diacritical Marks with Seen Example

Diacritic Sign	Unicode	Arabic Name	Diacritics Sound	Example	"S" Sound
◌َ	U+064E	*fatha*	*/a/*	سَ	*sa*
◌ِ	U+0650	*kasra*	*/i/*	سِ	*si*
◌ُ	U+064F	*ḍamma*	*/u/*	سُ	*su*
◌ّ	U+0651	*shadda*	doubling character	سّ	*ss*
◌ْ	U+0652	*Sukun*	No sound	سْ	*s*
◌ً	U+064B	*fathatan*	*/an/*	سً	*san*
◌ٍ	U+064D	*kasratan*	*/en/*	سٍ	*sen*
◌ٌ	U+064C	*dammatan*	*/oun/*	سٌ	*soun*

Diacritics are crucial in determining the poetic meters of Arabic poems. Additionally, the interpretation of Arabic text can vary based on the diacritical marks present on the characters. In the subsequent paragraphs, we will elucidate the significance of diacritical marks in both prose and poetry.

1.1.1 Arabic Poetry and Diacritics

Poetry is the archive and record of the history of Arab life, its aspirations, achievements, and wars from ancient times. Arabic poetry developed in ancient times, between the 9th and 12th centuries. The historical timeline of poetry is categorized into several distinct periods: the pre-Islamic, the early Islamic and Umayyad periods, the Abbasid, the Mamluk, the Ottoman, and the modern. Most of the poems date back to the pre-Islamic era, stating their ethics through highly diverse and complex poetic forms (Al-Musawi, 2006).

A poem in Arabic poetry follows a specific meter and rhyme scheme, highlighting the significance of sound patterns. In Arabic poetry, each line is divided into two equal halves in terms of metrical value. The first section of the verse is referred to as the "*sadr*," while the next part is known as the "*ajuz.*" Additionally, the poem employs a rhyme scheme called "*qafiyah*," in which each couplet concludes with the same letter or phonetic rhythm, indicated by a diacritical mark. If we assume that the poet ended the opening of the poem with the word "*bakhela*" "بخيلَ", then the poet must

conclude the rest of the poem's verses with a "*l* character with *fatha*" like "*nakhela*" and "*najeela*".

There are sixteen widely recognized meters in traditional Arabic poetry. Table 2 shows types of Arabic poetic meters and their demonstrations, such as Al-Wāfir and Al-Kāmil where each meter consists of a specific pattern of long and short. Although some poets prefer not to use diacritical marks, they claim it restricts their creativity. The placement of diacritical marks plays a critical role in indicating the length and stress of syllables, which are essential for maintaining the rhythm of the poem.

Table 2. Types Of Arabic Poetic Meters And Their Demonstrations

Meter	Metrical Pattern	Sample
Al-Taweel	فعولن مفاعيلن فعولن مفاعلن	أراك عصي الدمع شيمتك الصبر
Al-Kamel	متفاعلن متفاعلن متفاعلن	قم للمعلم وفّه التبجيل
Al-Wafer	مفاعلتن مفاعلتن فعولن	يطول اليوم لا ألقاك فيه
Al-Madeed	فاعلاتن فاعلن فاعلاتن	أدلال ذاك أم كسل
Al-Baseet	مستفعلن فاعلن مستفعلن فاعلن	يس الغريب غريب الشام واليمن
Al-Hazaj	مفاعيلن مفاعيلن	جميل الوجه أخلاني
Al-Sare'e	مستفعلن مستفعلن فاعلن	أودِعك الرحمن في غربتك
Al-Munsareh	مستفعلن فاعلاتن	ياليلُ طوبى لمعشرٍ رقدوا
Al-Mudhare'e	مفاعلن فاعلاتن	وغائبا عن عيوني
Al-Rajaz	مستفعلن مستفعلن مستفعلن	يا من إليه أشتكي من هجره
Al-Ramal	فاعلاتن فاعلاتن فاعلن	طلع البدر علينا
Al-Khafeef	فاعلاتن مستفعلن فاعلاتن	ما لليلي تبدلت
Al-Muqtadheb	فاعلات مفتعلن	ربما حصلت على اعتذار
Al-Mujtath	مستفعلن فاعلاتن	سئمت كل قديم
Al-Mutakareb	فعولن فعولن فعولن فعولن	أخي جاوز الظالمون المدى
Al-Mutadarak	فاعلن فاعلن فاعلن فاعلن	غنمي غنمي ما أجملها

1.1.2 Arabic Text and Diacritics

In Arabic, vowels are represented by dashes and symbols placed above or below the Arabic letters to ensure accurate pronunciation. However, as one becomes more proficient in the language, these vowels are used less frequently. However, they are still beneficial for beginners and intermediate learners to avoid any confusion when pronouncing Arabic correctly. Table 3 displays the effect of diacritical marks on word meaning. Let us take the word "الجَدّ" (grandfather) in Arabic; it is not equivalent to

“الجِدّ” (endeavor) or “الجُدّ” (aspect). As we can see, changing the diacritical mark on the same root word leads to entirely different meanings.

Table 3. Impact of Diacritical Marks on Word Meaning

Word	English Equivalent	Description
الجَدّ	*Aljadda*	Grandfather
الجِدّ	*Aljedda*	Endeavor
الجُدّ	*Aljoudda*	Aspect

It is important for our reader and learner to understand that certain combinations of Arabic letters and vowels produce distinct sounds. In addition, the presence of any of those signs on the Arabic letter results in the word having distinct grammatical functions.

1.2 Thesis Problem

Manual diacritization is time-consuming and requires a deep understanding of Arabic grammar. Existing automatic diacritization systems are limited in accuracy and effectiveness, often necessitating extensive training data and computational resources. To address these challenges, we propose a novel method leveraging pre-trained byte-to-byte language models and multi-phase training to enhance the precision and effectiveness of automatic diacritical restoration in Arabic text and poetry. The aim of this research is to develop a solution capable of handling the complexities of automatic diacritization in Arabic and providing viable solutions for this essential task.

1.3 Importance of Arabic Diacritization

Diacritization of Arabic plays a fundamental role in enhancing the accuracy, readability, and comprehension of Arabic text. It is essential for effective communication, understanding, and preservation of the Arabic language and its cultural heritage. Machine learning accurately predicts the diacritics of the Arabic language.

- Using machine learning techniques for diacritization prediction can lead to advancements in NLP technology for Arabic, which can benefit a wide range of applications such as machine translation, speech recognition, and sentiment analysis.
- The Arabic language has a rich literary and cultural past, with many major manuscripts available only in print. Machine learning techniques can help digitize these manuscripts and make them available online, allowing them to be preserved for future generations of academics, Arabic learners, and so on.

1.4 Thesis Objectives

The main objectives of this study are:

- To study and survey machine learning techniques for diacritization prediction.
- To study and analyze the diacritization in Arabic text and find its main characteristics and rules.
- To study and analyze the diacritization in Arabic poetry and find its main characteristics and rules.
- To address the challenges of diacritization in Arabic, such as the ambiguity and variability of diacritics and the complexity of Arabic morphology.
- To develop a machine learning model that can automatically predict diacritics for unmarked Arabic text. Specifically, the research aims to use the ByT5 model, a state-of-the-art language model, to perform diacritization on Arabic text and poetry.
- To improve upon existing diacritization models by leveraging the strengths of the ByT5 model, which is capable of handling multiple tasks (such as translation and summarization) and can be fine-tuned for specific tasks.
- To improve the accuracy of the proposed diacritization method using various metrics and to investigate the impact of training set size on the effectiveness of the approach.

- To use a multi-phase training approach that allows the model to gradually learn to predict diacritics for both fully and partially diacritic words.
- To improve the readability and comprehension of Arabic text and poetry.

1.5 Thesis Questions and Assumptions

This study will answer the following questions:

- What are the main challenges and limitations of diacritization in Arabic, and how can they be addressed using machine learning models such as the ByT5 model?
- How does the performance of the ByT5-based diacritization model compare to other state-of-the-art models in terms of accuracy, speed, and efficiency?
- Can multi-phase training improve the overall quality of the diacritization process in Arabic?
- The main assumptions of this research are:
- Pre-trained byte-to-byte language models can be effective for diacritization.
- Multi-phase training can improve the accuracy of diacritization.
- Arabic diacritization is a challenging task.

1.6 Thesis Contribution

We present a new way to automatically restore diacritics in Arabic text and poetry using the ByT5 model and working at the byte level in this manuscript. By using the unique features of byte-level processing to capture and restore the complex diacritical marks found in Arabic poetic texts, our research is different from other studies that have used the same method. We want to make diacritic restoration more accurate and faithful than ever before by using the ByT5 model's deep understanding of textual patterns at the byte level. This will also push the boundaries of computational linguistics for analyzing and preserving Arabic poetry.

1.7 Thesis Methodology

The research methodology of this thesis is summarized as follows:

- Studying and surveying the state-of-the-art diacritization prediction algorithms that are based on machine learning.
- Studying and analyzing Tashkeela's Clean-400 dataset and the second version of Arabic poem comprehensive dataset (APCD2).
- Selecting the ByT5 model, a state-of-the-art language model, and fine-tuning it for the task of diacritization on the training data.
- Evaluating, assessing, and testing the model by using well-known standard performance metrics such as WER, DER, F1-Scoor, and accuracy.
- Analyzing the results of the evaluation to identify the strengths and weaknesses of the ByT5-based diacritization model.
- Implementing a multi-phase training technique that involves using datasets with increasing sizes to tackle the problem of having limited training data for Arabic poetry. We begin with a smaller dataset (Clean-55k) and gradually progress to the full dataset (Clean-400 or APCD2). This method allows the model to develop the ability to predict and insert missing diacritics in both Arabic text and poetry.
- Report and document our work and present the results.

1.8 Thesis Outline

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

- Chapter 2 reviews previous research on our thesis topic.
- Chapter 3 provides a comprehensive explanation of our proposed model and explores the details of several key concepts in machine learning.
- Chapter 4 covers the investigation, illustration, and procedures involved in preparing the dataset for training.

- Chapter 5 presents the experiments conducted to improve the base model and the corresponding outcomes.
- Chapter 6 provides a summary of the conclusions and presents some future ideas to enhance the performance of the model.

Chapter Two

Literature Review

2.1 Introduction

This chapter will focus on previous studies on non-machine learning techniques for Diacritizing Arabic text and poetry. Also, previous studies used multi-stage techniques, and previous studies employed the ByT5 model.

This chapter will provide a brief summary of previous research that focuses on the implementation of machine learning techniques in the automated diacritization of Arabic text and poetry. This chapter consists of five sections. The first section focuses on papers that studied non-machine learning approaches to diacritizing Arabic text and poetry. The second one examines papers that have investigated the process of automated diacritization of Arabic text. The third section discusses papers that have investigated the process of automatic Arabic poetry diacritization. The fourth section presents research findings on multiple-phase neural network models. The final section focuses on papers related to text-to-text transfer *transformers*.

2.2 Traditional Approaches to Diacritize Arabic Text and Poetry

In the past, researchers used rule-based approaches to automatic diacritization of Arabic text and poetry that rely on a set of handcrafted rules to predict diacritics based on the context, dictionaries, syntactic analyzers and generators, and morphology of the word. El-Imam. (2003) assigned each letter a suitable sound using extensive Arabic-dependent criteria supplemented by an exception word dictionary. Using a rule-based method, Shaalan. (2010) proposed an Arabic morphological analyzer and syntax analyzer. While rule-based approaches can be effective for simple text, they are limited by the complexity and variability of the Arabic language.

Different statistical techniques, including hidden Markov models (HMMs) (Gal. 2002), Conditional random fields (CRFs) (Azim *et al*., 2012), n-grams (Hifny *et al*., 2012), and neural networks, have been employed to estimate the probability of diacritics in unaccented text. Gal. (2002) suggested a statistical method utilizing HMM to reconstruct Arabic and Hebrew diacritics. The Holy Qur'an served as the corpus for the suggested system. To accomplish this, these techniques need a substantial amount of labeled data to anticipate the likelihood of diacritics for a character sequence. The most noteworthy benefit is that they don't rely on manually created rules to address the problem, eliminating the need for extensive linguistic expertise. However, these approaches may be affected by noise and errors.

Another suggested system is the hybrid approach (Habash and Rambow. 2007), which combines the strengths of rule-based and statistical approaches for diacritizing Arabic text. It uses rules to handle simple cases and statistical models to handle more complex cases. This approach can achieve higher accuracy and robustness compared to rule-based or statistical approaches alone. However, it requires a large amount of annotated data and may be more complex to implement and maintain compared to other approaches.

2.3 Machine Learning Approaches to Diacritize Arabic Text

In recent times, recurrent neural networks (RNNs) have been effectively utilized to tackle the issue of sequence transcription, specifically in the context of recovering diacritical marks in Arabic texts and poetry.

Abandah *et al*. (2015) proposed a novel approach that utilized RNNs for the task of diacritization, which is the process of adding diacritical marks to unmarked Arabic text. They used a deep bidirectional long-short-term memory network (BiLSTM) that incorporates both forward and backward connections, enabling the network to acquire

contextual details from both preceding and succeeding words in a text sequence. The suggested method achieved an average WER of 5.82% and a DER of 2.09% on the LDC, ATB3 benchmark. To enhance the accuracy and performance of diacritization, they proposed (Abandah and Arabiyat, 2017) the MADAMIRA full morphological and syntactical analyzer. This analyzer's high-confidence diacritics and word segmentation output are fed only to the RNN.

Mubarak *et al*. (2019) proposed a LSTM RNN model that utilizes character-level sequence-to-sequence modeling techniques to accurately add diacritics to Arabic text. The effectiveness of the proposed method is that they trained a model with a fixed-length sliding window of n words to address the poor handling of long-range dependencies. They also used a voting algorithm to determine the most commonly diacritized formula of a word in various contexts. The result shows that the LSTM RNN model gives a WER of 4.49%, which is 64% lower than the previous work.

Fadel *et al*. (2019) introduced models that were built utilizing two different approaches: feed-forward neural networks (FFNN) and RNN. These models utilized 100-hot encoding, embeddings, CRF, and block-normalized gradient (BNG) techniques. The authors extracted a subset of 55K sequences from the Tashkeela dataset and cleaned them to evaluate and compare the proposed model with automatic diacritization tools. The result shows that the proposed models have the best results for automatic diacritization.

Karim and Abandah (2021) utilized the extensive Tashkeela dataset to build a cleaned corpus with about 550k sequences. In order to convert undiacritized Arabic letter sequences into fully diacritized ones, their baseline model incorporates a deep BiLSTM with RNN. Their method yielded an average WER of 3.89% and a DER of 1.45%.

Madhfar and Qamar (2020) introduced the convolutional block attention module and highway network (CBHG), which consists of three deep learning models, for the purpose of restoring diacritics in Arabic text. This model is an extension of their prior work in text-to-speech synthesis using deep learning. The initial model acts as a reference point and comprises six layers, which include an embedding layer, three BiLSTM layers, a projection layer, and a SoftMax layer. It is utilized to evaluate the efficacy of a basic deep learning model on the corpus. The second model utilizes an encoder-decoder architecture that has been specifically modified to address the diacritic recovery problem. These adjustments include the incorporation of location-based attention. The third model utilizes the encoder component from the second model as its foundation. It was shown that the CBHG model has a quicker training speed and delivers state-of-the-art results in all areas of measurement. The model yielded a DER of 1.13%.

Kharsa *et al*. (2024) introduced a novel approach to diacritizing Arabic text by leveraging bidirectional encoder representations from *transformer* (BERT) models. They employed two datasets, namely the Arabic Diacritization (AD) dataset and the Tashkeela Processed (TP) dataset. The suggested system attained a DER of 0.92% and a WER of 1.91% when evaluated on the AD dataset. Furthermore, the bidirectional encoder representations from *transformers* (BERT) models demonstrated a substantial reduction in DER on the Tashkeela processed (TP) dataset, lowering the benchmark from 4.0% to 1.11%. Furthermore, they implemented SUKOUN, a diacritization system that operates in real-time and features a user-friendly website.

Table 4 illustrates the most significant previous studies on Arabic text diacritization.

Table 4. Prior Research on Arabic Text Diacritization

Study Authors	Dataset	Model	Evaluation Metrics	Results Report
Abandah *et al.* (2015)	Tashkeela	BiLSTM	DER, WER	2.09%, 5.82%
Mubarak *et al.* (2019)	Tashkeela	LSTM	WER	4.49%
Fadel *et al.* (2019)	Clean-50	FFNN, RNN	DER, WER	2.18%, 4.44%
Madhfar and Qamar (2020)	Tashkeela	Three BiLSTM	DER, WER	1.13%, 4.43%
Karim and Abandah (2021)	Clean-400	BiLSTM	DER, WER	1.45%, 3.89%
Al-Rfooh *et al*. (2023)	Clean-400	Base ByT5	DER, WER	1.00%, 2.92%
Kharsa *et al.* (2024)	Clean-50 Tashkeela	BERT	DER, WER DER	0.92%, 1.14% 1.11%

2.4 Machine Learning Approaches to Diacritize Arabic Poetry

Yousef *et al*. (2019) presented a model that utilized RNN trained at the character level to acquire the ability to identify and understand the meters of 16 Arabic and 4 English poems in written form. The researchers gathered a substantial 1.5 million verses of data, and they launched the data repository "Poem Comprehensive Dataset (PCD)," which is publicly accessible, providing them in a cleaned, structured, and well-documented format for use in future research endeavors. The neural networks demonstrated high accuracy in correctly classifying poems, achieving an overall accuracy rate of 96.38% for Arabic poetry and 82.31% for English poetry.

Abandah *et al*. (2022) proposed Bi-LSTM Deep RNN to diacritize Arabic poetry and classify input verses into 16 poetry meters and prose, utilizing a big dataset (APCD2) of 1657 k lines of poems and prose that have been updated and expanded from APCD. The proposed network is properly adjusted and achieves a substantially higher average accuracy (97.27%) than earlier work; it also detects 16 poetry meters and prose.

Abandah *et al*. (2022) handled the problem of adding diacritics to Arabic poetry verses by leveraging pattern characteristics from a pre-trained classification model via transfer learning. Furthermore, they address the scarcity of training data by training their diacritization model in stages (multi-phase training) using the second version of the APCD2. When compared to the most successful previous outcomes, the solutions they proposed yield a significant improvement in the diacritization error rate, reducing it from 6.08% to 3.54%, which corresponds to a 42% enhancement.

Table 5 illustrates the most significant previous studies on Arabic poetry diacritization.

Table 5.Prior Research on Arabic Poetry Diacritization

Study Authors	Dataset	Model	Type	Evaluation Metrics	Results Report
Yousef *et al.* (2019)	APCD	RNN	Classification	Accuracy	96.38%
Abandah *et al.* (2022)	APCD2	BiLSTM	Classification	Accuracy	97.27%
Abandah *et al.* (2022)	APCD2	BiLSTM	Diacritics	DER, WER	3.54%, 12.34%

2.5 Exploring Multi-stage Neural Network Models for Diacritization

Multistage models, also known as cascaded models, are an approach to building models that involves using multiple consecutive stages in order to make accurate predictions and classifications.

Shahzadi *et al*. (2018) implemented a CNN cascade with a long short-term memory (LSTM) network on the brain Tumor segmentation (BRATS) dataset to classify 3D MR images of brain tumors into HG and LG gliomas. They used the LSTM network to learn high-level feature representations from pre-trained visual geometry group 16 (VGG-16) and classified 3D brain tumor volumes into HG and LG gliomas.

Keran *et al*. (2022) proposed multi-stage transfer learning for fake news detection using the asynchronous stochastic gradient descent weight-dropped LSTM (AWD-

LSTM) on two benchmark datasets: semEval-2017 task 8 and PHEME, in which user comments are used to predict response tweet stances. Based on these predictions, it detects fake news by determining whether a tweet is true or false.

Ristea *et al*. (2024) proposed transcribing speech using automatic speech recognition (ASR) models and translating it into different languages using pretrained translation models to take advantage of multimodal representations. They got an audio-textual multimodal representation for each data sample. Then they used a novel cascaded cross-modal *transformer* to combine language-specific BERT with Wav2Vec2.0 audio features. The model uses two cascaded *transformer* blocks: the first combines text-specific features from different languages, while the second combines acoustic features with multilingual features learned by the first *transformer* block. The authors also applied their framework to the Speech Commands v2 and HarperValleyBank dialog data sets.

2.6 Sequence-to-Sequence *Transformer* Models

Stankevičius. (2022) utilized the recently created universal ByT5 byte-level seq2seq *transformer* model to restore diacritics and correct spelling errors. They conduct diacritic restoration on benchmark datasets for 12 languages, including Lithuanian. Their method successfully restores diacritics in previously unseen words, with an accuracy rate exceeding 76%. Furthermore, their method of restoring diacritics and correcting typos simultaneously achieves an alpha-word accuracy of over 94% across 13 different languages.

Jentoft. (2023) utilized the ByT5 byte-level seq2seq *transformer* model for grammatical error correction (GEC) tasks in both English and Norwegian. For English, they employed the NAIST dataset, which comprises learner texts from lang-8.com, a language-exchange social network, and the FCE dataset, containing responses to

prompts written by candidates who have taken the Cambridge ESOL First Certificate in English exam. Their model achieved an F0.5 score of 0.51 for English. Additionally, for Norwegian, they utilized the ASK dataset, consisting of approximately 50,000 sentences written by non-native language learners at two different levels of Norwegian proficiency. With their best model, they achieved an F0.5 score of 0.581.

Al-Rfooh *et al*. (2023) suggested the use of fine-tuning token-free pre-trained multilingual models, notably ByT5, to acquire the ability to predict and add missing diacritics in Arabic text. They did the first tests using Tashkeela's dataset, namely Clean-400. The methods they provided demonstrate a significant decrease in diacritization error rate when compared to the most successful previous findings, reducing it from 1.13% to 0.74%, which corresponds to a 40% improvement.

Kirov *et al*. (2024) conducted a study utilizing the ByT5 byte-level seq2seq *transformer* model for transliteration of full sentences from informal romanization prevalent in South Asia to their native scripts. This study covered all 12 languages included in the Dakshina dataset. Their findings showed an overall 3.3% absolute (18.6% relative) mean word-error rate reduction.

The purpose of our thesis is to explore the use of token-free, pre-trained multilingual models, specifically ByT5, by fine-tuning them through experimentation. Furthermore, we address the issue of limited training data by utilizing a multi-phase training approach with an incremental dataset. In this approach, the results obtained from the Clean-400 training phase are used as input for subsequent training stages on the APCD2. The primary objective is to enable the model to predict and incorporate missing diacritics in Arabic text and poetry. The study seeks to evaluate the accuracy of the trained model under various conditions and analyze how the size of the model impacts diacritization accuracy.

Chapter Three

Sequence Transcription Approaches

This chapter outlines the evolution of models used for automatic diacritization of Arabic text and poetry, leading up to the model examined in this study (ByT5). Additionally, we explore two stages in these models. In the first stage, a diacritization text model is utilized, trained to forecast diacritics for prose based on input prose without diacritics. In the second stage, a diacritization poetry model is employed, predicting diacritics for input verses. The diacritization text model undergoes training on a substantial, clean dataset, achieving high-performance diacritic prediction for text. The second stage utilizes this text data to enhance diacritic prediction for input verses. We propose these models under the assumption that the diacritization stage could achieve more precise diacritic prediction when familiar with diacritization patterns; verse diacritics are closely linked to the verse meter. Further details regarding the investigated model are provided in Subsection 3.8.

3.1 Introduction

This section represents a crucial turning point in understanding and advancing deep neural network models for Arabic text and poetry diacritization applications. This chapter focuses on the utilization of deep neural networks, including RNNs and LSTM networks, as well as the ByT5 model in diacritizing Arabic text and poetry.

3.2 RNN

According to Rumelhart *et al*. (1986), RNN are a model of parallel distributed processing that performs time-dependent processing through connections based on feedback. RNNs have a memory that enables them to capture temporal dependencies and sequential patterns. Therefore, they are widely used in NLP tasks due to their ability to handle sequences of variable lengths while retaining memory of previous

inputs. Additionally, they excel at capturing context and dependencies between words or characters in sentences.

As a result, the following figure illustrates the relationship between networks in the recurrent processing model. Each input sequence benefits from the output of the previous iteration to make predictions about the output of the next iteration (Zhang *et al*. 2019).

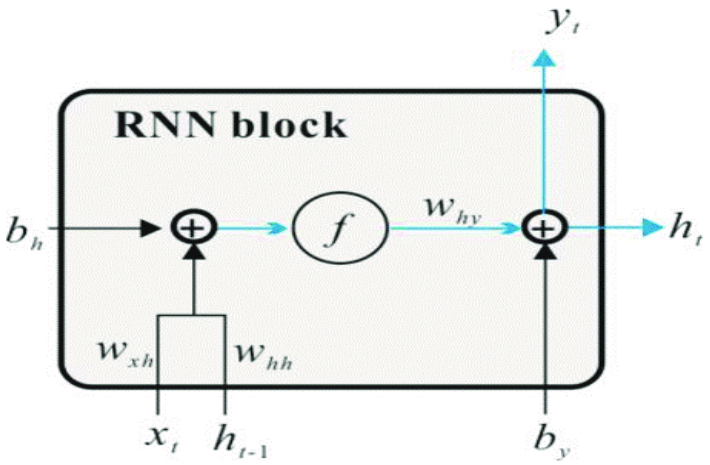

Figure 1. Visual Representation of RNN Architecture (Zhang *et al*. 2019)

In a standard recurrent neural network (RNN), when given an input sequence $x = (x_1, \cdots, x_T)$ the RNN computes a sequence of hidden vectors $h = (h_1, \cdots, h_T)$ and an output vector by following the below equations iteratively from $t = 1$ to $t = T$:

A hidden state h_t is maintained while it receives an input x_t, generates an output y_t, and keeps a hidden state. The following equations are used to characterize the system:

$$h_t = H(W_{xh}x_t + W_{hh}h_{t-1} + b_h) \quad (1)$$

$$y_t = W_{hy}h_t + b_y \quad (2)$$

Equation 1 represents a hidden state (h_t) at time step t which is computed based on the current input (x_t), the previous hidden state (h_{t-1}), and bias (b_h), using weight matrices $W_{xh}$and W_{hh}.

Equation 2, y_t denotes the output at time step t. It is calculated based on the current hidden state (h_t) using weight matrix (W_{hy}) and bias (b_y), followed by applying an activation function. (Zhang *et al*., 2019).

Recurrent neural networks can retain recent input events in short-term memory through feedback connections. However, they may face challenges in capturing long-term dependencies between characters or words in NLP applications such as text generation, machine translation, and text diacritization. Moreover, they may suffer from either prolonged training times or ineffective performance due to the vanishing gradient problem. To address these issues, a model called long short-term memory was developed.

Long short-term memory is a specific variant of the recurrent neural network (RNN) that was specifically developed to tackle the vanishing gradient problem. It utilizes memory cells to retain the hidden states from earlier stages, enabling the model to effectively analyze and process long-range sequences (Hochreiter and Schmidhuber, 1997).

Schuster and Paliwal (1997) proposed bidirectional RNNs because they capture dependencies not only from past context but also from future context, allowing them to better understand and capture temporal patterns in sequential data. The main difference between BiLSTM and traditional LSTM lies in their architecture. While LSTM processes input sequences sequentially from past to future, BiLSTM processes sequences in both directions simultaneously: from past to future and from future to past, as shown in Figure 2. Therefore, BiLSTM was used instead of LSTM in some applications to enhance the model's ability to capture long-range dependencies and improve overall performance on tasks that require understanding both past and future context in the data.

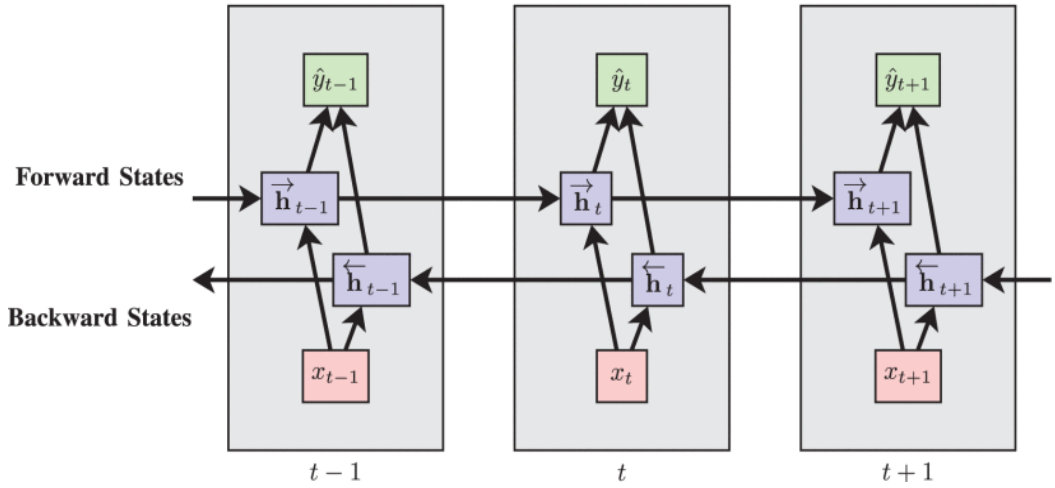

Figure 2. Visual Representation of BiLSTM Architecture (Madhfar *et al.*, 2020)

Researchers began stacking multiple RNNs, LSTM, and BiLSTM layers on top of each other due to the need to capture complex and hierarchical features in sequential data. These designs are called deep architectures (deep RNNs, deep LSTM, and deep BiLSTM).

Deep RNN are formed by vertically arranging numerous RNN hidden layers, with the output sequence of one layer being used as the input sequence for the following layer.

$$h_t^n = H(W_{h^{n-1}h^n} h_t^{n-1} + W_{h^n h^n} h_{t-1}^n + b_h^n) \quad (3)$$

$$y_t = W_{h^N y} h_t^N + b^y \quad (4)$$

Equation 3 displays the hidden vector sequences (h_t) which are generated repeatedly for all layers from $n = 1$ to $n = N$ level and from $t = 1$ to $t = T$ time step, assuming a constant hidden layer function across all N levels in the stack. The network's output (y_t) is determined by equation 4. This iterative procedure enables the network to progressively collect more complex and organized characteristics from the input data, resulting in enhanced performance in applications like sequence transcriptions in hybrid HMM neural network systems (Graves *et al.*, 2013).

Furthermore, the majority of prior models depend on LSTM or RNN as foundational models, which are computationally demanding and require lengthy training periods because of their sequential structure. On the other hand, *transformer* models that utilize self-attention layers provide benefits in terms of computing complexity and the ability to parallelize tasks.

3.3 *Transformers*

Transformer models rely on the self-attention mechanism, which is a system that connects distinct points inside a single sequence in order to determine the representation of the sequence to efficiently deal with long-term contexts, making them an ideal choice as an alternative to RNNs (Vaswani *et al*., 2017).

3.3.1 *Transformer* Architecture

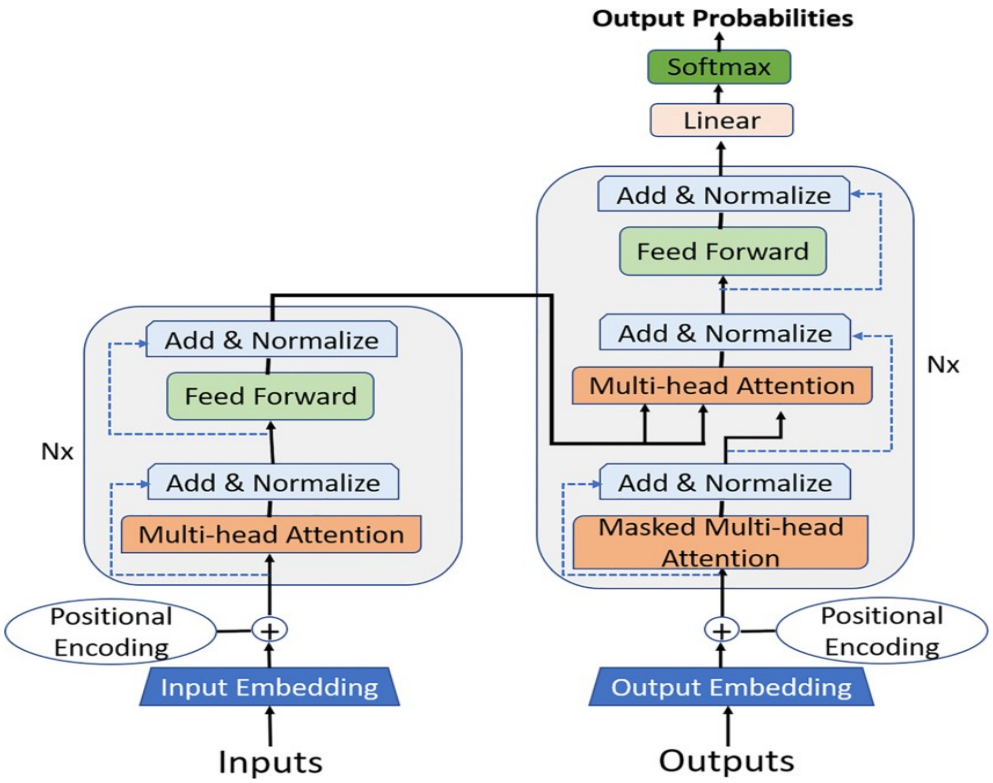

Figure 3.*Transformers* Structure (Boyle and Kalita, 2023)

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks (CNN) that include an encoder and a decoder. The best-performing models also connect the encoder and decoder through an attention mechanism. Vaswani *et al*. (2017) proposed a new simple network architecture, the *Transformer*, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Figure 3 illustrates the *transformer* structure, which consists of embeddings, encoder-decoder, feed-forward neural network (FNN), and multi-head attention.

3.3.2 Word Embedding

Word embedding is a technique used in NLP to represent words as dense vectors of real numbers in a low-dimensional space. This transformation allows words to be represented in a continuous vector space with size, where similar words are closer together and dissimilar words are farther apart; therefore, it allows the model to understand the relationships between words and analyze the text better. This is done by multiplying the encoded input data with the learned weight representation trained during the *Transformer* model's training. Different techniques and pretrained models have been suggested to create compact word embeddings efficiently. Nonetheless, the *Transformer* employs an embedded word layer integrated into the model architecture and trained from the beginning.

3.3.3 Stacks of Encoder and Decoder

A *transformer* model's encoder is responsible for translating the input sequence into a form that machines can understand. This form summarizes the correlation between word similarity and its placement in the sequence. The input sequence undergoes initial processing through input embedding and positional encoding layers.

These operations transform the sequence into a format suitable for processing by the encoder layer.

- **Encoder**

The encoder consists of $N = 6$ identical layers stacked on top of each other. Each layer comprises two sub-layers: the first is a multi-head self-attention mechanism, and the second is a basic, fully connected feed-forward network operating based on position. After each sub-layer's operation, the input x is added to the output $sublayer(x)$, and then layer normalization is applied to the sum. This process, known as "Residual Connection" helps in better preserving information flow throughout the network (He *et al*., 2016). All of the model's sub-layers generate vectors of equal length, referred to as d_m.

- **Decoder**

The decoder in the *Transformer* model, like the encoder, comprises a series of $N = 6$ identical layers. However, it introduces an additional sub-layer to conduct multi-head attention over the encoder stack's output.

3.3.4 Attention Mechanism

In *Transformer*s, the attention mechanism helps the model understand how words in a sentence relate to each other. It does this by looking at the importance of each word and how it connects to others. This is crucial for tasks like understanding text, translating languages, and generating new sentences.

The function of attention is a process that takes a query and a set of key-value pairs as input and generates an output. All inputs and outputs are represented as vectors. The output is obtained by calculating the weighted sum of the values, where the weight for each value is determined based on the compatibility between the query and its corresponding key.

multi-head attention sub-layer incorporates the self-attention mechanism. The attention scores are derived by computing the dot product between the query and the keys, which is then scaled by the square root of dimension of key. The SoftMax activation function is then applied to the resulting values.

Suppose Q, K, V, d_k and d_v represent the matrices of queries, keys, values, a dimension of key and a dimension of value respectively. Then, the attention matrix is computed using the formula:

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

To simultaneously consider different aspects of the input data, the multi-head attention layer in a *Transformer* model employs several parallel attention mechanisms. These mechanisms utilize linear transformations of the query, key, and value matrices using trainable weight matrices $W^Q \in \mathcal{R}^{d_m \times d_q}$,$W^K \in \mathcal{R}^{d_m \times d_k}$and $W^V \in \mathcal{R}^{d_m \times d_v}$. By decreasing the dimensionality of the query, key, and value vectors (d_v=d_k=d_m/h), h represents the number of attention heads, the computational load of multi-head attention is reduced, while ensuring that attention scores are outputted in the necessary model dimension.

After computing the attention scores for each head, they are concatenated and passed through a final linear projection$W_0 \in \mathcal{R}^{hd_v \times d_m}$ to produce the final output vector in the d-dimensional space. The equation for the multi-head attention mechanism is represented as follows:

$$MultiHead(Q,K,V) = Concat(head_1, \dots, head_h)\, W_0 \quad (6)$$

$$where\ head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \quad (7)$$

The result from the multi-head self-attention layer is directed to the position-wise FNN.

3.3.5 Feed Forward Network (FNN)

The position-wise FNN is described as a two-layer Multi-Layer Perceptron (MLP), the equation is expressed in Equation 8. Where x represents the input, $W1$ and $W2$ denote weight matrices, and $b1$ and $b2$ are bias terms.

$$FFN(x) = max(0, xW1 + b1)\,W2 + b2 \quad (5)$$

3.4 Transfer model

Transfer learning, a machine learning technique, involves repurposing or adjusting a model trained on one task as a starting point for a related task. It leverages knowledge acquired from solving one problem (the source task) to aid in solving a different yet related problem (the target task), showing effectiveness in enhancing diverse tasks and reducing training time. Pre-trained models are commonly used in transfer learning; they are initially trained on a large dataset for a specific task and then fine-tuned on a smaller dataset for a related task. There is a growing trend of pre-training NLP models on tasks with large datasets. Early results on transfer learning for NLP used RNN (Peters *et al*., 2018), but it has become more common to use models based on the *transformer* architecture.

Transformer models can generally be classified into different types depending on their structure and purpose. Figure 4 demonstrates three main categories of *transformers*: encoder-based, decoder-based, and encoder-decoder-based. The transfer *transformer* model employed in our study falls under the encoder-decoder based category and adheres closely to its original design, as proposed by Vaswani *et al*. (2017), with slight adjustments for handling byte sequences, notably pre-trained ByT5. ByT5 design closely resembles mT5 which is the multilingual variant of T5, introduced by Xue *et al*. (2020).

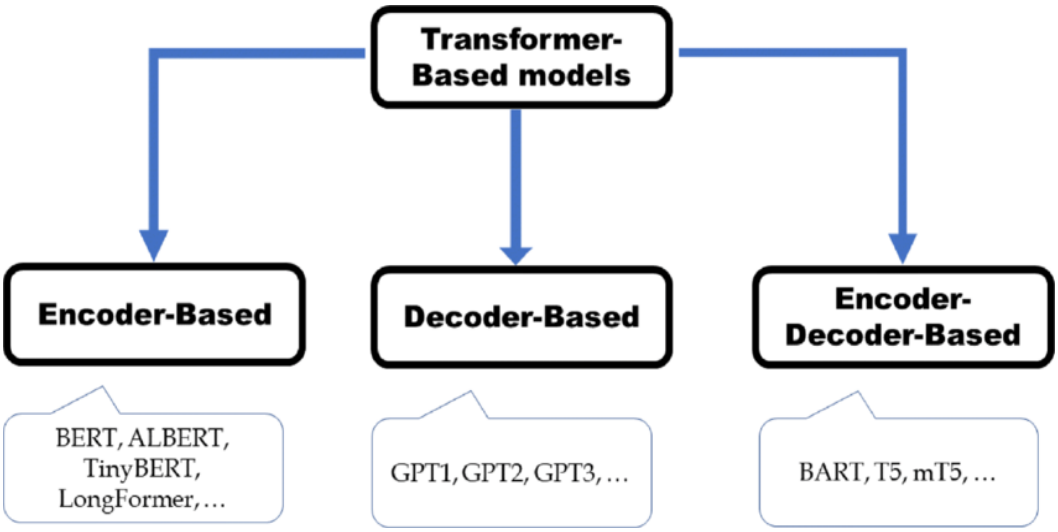

Figure 4. Categorizing *Transformers* Based on Architecture Types (Nassiri and Akhloufi. 2023)

3.5 Text-To-Text Transfer Transformer

T5, or "Text-To-Text Transfer *Transformer*," is an NLP model developed by Google researchers, as described in the study by Raffel *et al*. (2020). Unlike other models, T5 approaches each language task as a text-generation task. This versatility allows T5 to handle various tasks such as translation, summarization, question answering, and text classification using a unified method, as illustrated in Figure 5. This adaptability and effectiveness make T5 highly flexible and powerful, establishing it as one of the leading language models available.

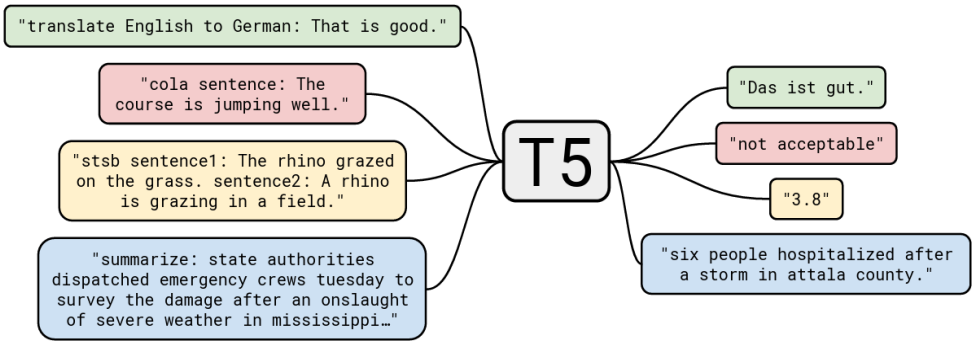

Figure 5.T5 Performance Across General NLP Tasks (Raffel *et al*., 2020)

In the following subsections, we will describe the architecture of T5 in the first part. Following that, in the second part, we will outline the model setup. Subsequently, we will move on to discuss the data used to train T5. Lastly, we will provide an evaluation of T5 performance.

3.5.1 T5 Architecture

The T5 architecture consists of an encoder-decoder *transformer* framework, similar to the farmwork outlined in the *transformer* subsection. However, it incorporates some key enhancements. One significant difference is the inclusion of normalization layers and position embeddings.

The normalization layers ensure stable training and improved convergence by standardizing activations across different layers. Additionally, T5 incorporates position embeddings into input embeddings to encode word positions in sequences, aiding in understanding text structure and enhancing contextual comprehension, particularly for tasks like language translation and text summarization.

3.5.2 The Model Setup

Vaswani *et al*. (2017) suggested using a standard encoder-decoder *transformer* for the T5 model. The configuration and size of the encoder-decoder T5 model closely resemble that of a "$BERT_{BASE}$" (Devlin *et al*., 2018), with both the encoder and decoder consisting of 12 blocks. Each block encompasses self-attention, optional encoder-decoder attention, and a feed-forward network. These networks feature a dense layer output at 3072 dimensions, followed by a ReLU activation function and another dense layer. All attention mechanisms have an inner dimensionality of 64 and 12 heads. Additionally, all other sub-layers and embeddings are set to 768 dimensions. In total, this yields a model with approximately 220 million parameters, roughly twice the count of "$BERT_{BASE}$", given the inclusion of two-layer stacks instead of one. To mitigate

overfitting, dropout with a probability of 0.1 is applied throughout the model (Raffel *et al*., 2020).

3.5.3 Colossal Clean Crawled Corpus (C4)

The C4 sourced from Common Crawl is a valuable asset for training NLP models. Common Crawl provides a vast repository of text data obtained through web scraping, totaling around 20TB monthly. Despite containing various non-textual content like gibberish and aggressive language, Common Crawl has found extensive use in NLP applications such as language modeling and machine translation. To prepare the dataset for NLP tasks, it underwent cleaning processes. The resulting dataset is C4, contains approximately 750 GB of clean and natural, unlabeled English text. It is publicly available via TensorFlow Datasets (Raffel *et al*., 2020).

3.5.4 T5 Evaluation Performance

Raffel *et al*. evaluate the performance of T5 across multiple domains, including language translation, question answering, text summarization, and text classification. These evaluations involve utilizing benchmarks such as GLUE and SuperGLUE for text categorization, CNN/Daily Mail for text summarization, and SQuAD for question answering. Additionally, they examine T5's translation capabilities from English to German, French, and Romanian using WMT datasets (Raffel *et al*., 2020).

The T5 pretrained model is well-known for its unique approach, using a single “text-to-text” format that covers various text-based NLP tasks like translation and summarization. Instead of just giving a class index, T5 directly produces the actual text of the label, making training more straightforward with a consistent objective, namely teacher-forced maximum likelihood. However, T5 has limitations. It is trained exclusively in the English context, restricting its effectiveness in multilingual contexts. To address these limitations, the mT5 model emerges. mT5 builds upon T5's strengths

but overcomes its weaknesses by being multilingual, allowing it to perform well across different languages.

3.6 Pre-Trained mT5

Xue *et al*. (2021) introduced the mT5 model, "Massively Multilingual Pre-Trained Text-to-Text *Transformer*", which shares similarities with T5 in architecture and training approach. mT5 is based on the "T5.1.1" formula, enhancing T5 by integrating GeGLU nonlinearities, adjusting d_{model} and d_{ff} dimensions, and conducting dropout-free pre-training exclusively on unlabeled data (Shazeer, 2020).

As a multilingual model, mT5 faces the challenge of effectively selecting data from diverse linguistic contexts. Over-representing low-resource languages might cause overfitting, whereas overlooking high-resource languages might cause underfitting. To address this, the model employs the boost lower-resource languages approach, sampling examples based on the probability $P(L)$ α $|L|^{\alpha}$, 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 (typically with $\alpha < 1$) enables to manage the extent of boosting on languages with limited resources. Researchers found the optimal value for α to be 0.3 (Xue *et al*., 2021).

Given that the mT5 model spans over 100 languages, it requires a larger vocabulary, which has consequently been extended to 250,000 wordpieces. Furthermore, the model utilizes SentencePiece as T5 (Xue *et al*., 2021).

3.7 mC4

The mC4 dataset is a multilingual expanded version of the C4 dataset and a crucial resource for training models like mT5 and ByT5. These models leverage the diverse text data from mC4, advancing NLP tasks. They include text data collected

through monthly data collection efforts by Common Crawl in over 100 different languages.

The dataset is significantly larger than C4, amounting to 6.6 billion pages and 6.3 trillion tokens (Xue *et al*., 2021).

3.8 Pre-Trained ByT5

In 2022, Xue *et al*. introduced the byte-to-byte Model (ByT5), a token-free adaptation of the multilingual mT5 model. ByT5 simplifies the NLP workflow by eliminating the need for vocabulary creation, text preprocessing, and tokenization. Trained on the mC4 dataset, ByT5 has demonstrated exceptional performance across various community benchmarks. Available in five sizes" small, base, large, xl, xxl" comparable to T5 and mT5, it supports over 100 languages. ByT5 proves particularly beneficial for tasks involving short to medium-length text sequences.

The following subsections outline the ByT5 architecture. The first subsection delves into key modifications in ByT5 architecture, while the second subsection explains the token-free concept.

3.8.1 Key Modifications in ByT5 Architecture

Xue *et al*. (2022) introduced several modifications to mT5 to create ByT5. These modifications are present in Figure 6, which includes:

- ByT5 eliminates the use of SentencePiece vocabulary, opting to directly process UTF-8 bytes into the model without prior text preprocessing (auto-tokenization).
- ByT5 changes how it trains compared to mT5. Instead of corrupting spans of tokens as mT5 does, ByT5 replaces spans with the final 100-byte IDs. It also introduces masking for longer byte-spans by using a mean mask span length of 20 bytes, which is different from mT5's average span length of 3 sub-word tokens.

- ByT5 adjusts its architecture by increasing the depth of the encoder compared to the decoder, aiming for a more robust encoder similar to architectures like BERT. This change enhances performance in classification and generation tasks.
- To comply with the UTF-8 standard, ByT5 excludes any illegal byte sequences from the model's output, although such instances are infrequent in real-world scenarios.

·ByT5 retains several settings from mT5, including a sequence length of 1024 (measured in bytes) and training for 1 million steps over batches consisting of 220 tokens.

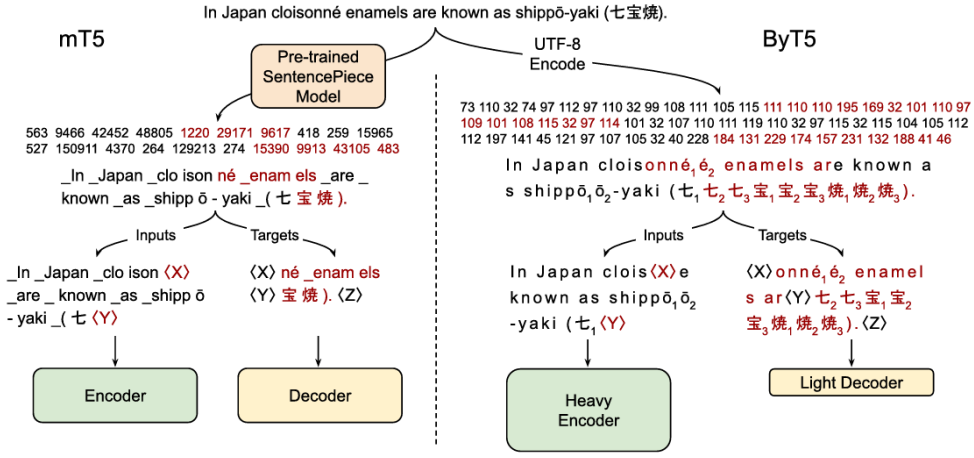

Figure 6. Comparing Pre-training Methodology and Network Structure: mT5 vs. ByT5 (Xue *et al.*, 2022)

3.8.2 Evolution from Token-Based to Token-Free Models in NLP

Tokenization refers to the process of assigning unique token identifiers to words within a predefined vocabulary, which facilitates the transformation of text into sequences of tokens. Nevertheless, challenges arise due to the presence of words not included in the vocabulary and various undesirable behaviors observed with subword tokenizers. In response to these challenges, the concept of token-free NLP models has emerged, which operate directly on raw text without relying on predefined vocabularies. These models, such as byte-level models, process text sequences as

sequences of individual bytes, providing benefits in terms of model size, adaptability to different languages, and variations in orthography. However, byte-level models often result in longer sequences, which can lead to higher computational costs. Nonetheless, recent research indicates that *transformer* architectures can be effectively adapted to handle byte sequences, mitigating some of these challenges.

The sequence-to-sequence design of the ByT5 model overcomes the constraints of the most recent advanced diacritics restoration model, which relies on BERT (Kharsa et al., 2024). The latter model operates as an auto-encoder, conducting classifications for individual tokens. Specifically, it predicts the correct classes for each token correction, defined by position and diacritic type. However, this system is confined to its predetermined set of instructions (correction classes), which is heavily influenced by language and focuses solely on diacritic restoration. In contrast, our sequence-to-sequence approach with ByT5 enables us to address various grammatical errors and learn to generate output sequences in a more universal and language-independent manner.

Chapter four

Datasets Preparations and Experiments

In Chapter 4, the focus is on preparing the necessary datasets and developing the model required for the study. This chapter begins with an introduction to the datasets used, namely the Clean-400 and the APCD2, followed by an outline of the training methodology. Additionally, it covers the ByT5 model selection, training hyperparameters, experimental environment, and performance evaluation metrics such as accuracy, DER, and WER. Through this comprehensive approach, Chapter 4 aims to provide a thorough understanding of the dataset preparation process and the model setup employed in the study.

4.1 Clean-400

The Tashkeela dataset is a collection of Arabic vocalized texts that comprises both modern and CA languages. It contains about 75 million fully vocalized words sourced from 97 novels and organized in text files. The corpus is primarily derived from Islamic canonical texts with a semi-automatic web crawling approach. Only a small portion, approximately 1.15% of Tashkeela, consists of MSA texts, so it is considered a CA dataset (Zerrouki and Balla, 2017).

Fadel *et al*. (2021) isolated 55,000 sequences from the Tashkeela dataset with a diacritic-to-character rate of at least 80%. The subset was cleaned using various procedures, including the removal of English letters and whitespace, correcting diacritics, and isolating numbers from words. Karim *et al*. (2021) employed Fadel's filtering and cleaning methodologies to extract a larger dataset, ensuring an 80% diacritization-to-character ratio. Subsequently, they wrapped the sequences to 400 characters per sequence maximum.

In the initial stage of our investigation, we utilized the Clean-400 dataset, which was created and supplied by Abandah and Abdel-Karim. This dataset comprises roughly 496,000 sequences, each of which is formatted into 400 characters per sequence. The decision to employ this formatted dataset stemmed from its capacity to reduce memory consumption and training time. Table 6 illustrates various characteristics of the dataset, such as its size, word count, letter count, letters per word, and words per sequence. For the purpose of training and assessing our model in each experiment, we partitioned the dataset into 489,000 lines for training, 3,060 lines for validation, and an additional 3,120 lines for testing.

Table 6. Clean-400 Corpus Characteristics After Wrapping

Criteria	Clean-400 dataset		
	Training	**Validation**	**Test**
Size (MB)	263	1.59	1.66
Number of samples ($\times 10^3$)	489	3.06	3.12
Word count ($\times 10^6$)	19	0.12	0.13
Character count ($\times 10^6$)	130	0.79	0.82
Letters per Word	3.97	3.97	3.97
Words per sequence	40.08	39.18	40.13

4.2 Arabic Poem Comprehensive Dataset (APCD2)

The primary dataset conducted in diacritization Arabic poetry is the second version of the Arabic poem comprehensive dataset (APCD2), which was extracted from the original APCD by removing overly short and long verses based on specific criteria and taking out incomplete verses. It comprises 1,657 thousand Arabic poems, and the prose dataset includes 16 poetry meters and some Arabic prose (Abandah *et al*., 2020). The table below displays the poetry meters class distribution in APCD2.

Table 7. Poetry Meter Class Distribution and Relative Verses Ratio

No.	Meter	Meter Ratio
1	*Al-Ṭawīl*	24.35%
2	*Al-Kāmil*	22.04%
3	*Al-Basīṭ*	14.42%
4	*Al-Khafīf*	9.41%
5	*Al-Wāfir*	8.04%
6	*Al-Rajaz*	6.19%
7	*Al-Ramal*	4.39%
8	*Al-Mutaqārib*	3.81%
9	*Al-Sarīʿ*	3.47%
10	*Al-Munsariḥ*	1.70%
11	*Al-Mujtathth*	0.95%
12	*Al-Madīd*	0.46%
13	*Al-Hazaj*	0.41%
14	*Al-Mutadārik*	0.27%
15	*Al-Muqtaḍab*	0.04%
16	*Al-Muḍāriʿ*	0.01%

The dataset contains different poetry periods, each contributing differently to the overall collection. Below, the figure shows how poetry is distributed across these periods. The modern era has the most poetry, making up around 39.55% of the dataset; so that there is a lack of diacritization in APCD2. In the modern era, poets are not obliged to employ diacritical marks. Given that the bulk of the dataset originates from modern eras, we can deduce the underlying cause for the absence of diacritical marks within the data. Conversely, the Islamic era constitutes the smallest proportion, comprising only approximately 0.13% of the dataset. This variety in period representation underscores the dataset's inclusiveness, covering poetry from various historical times.

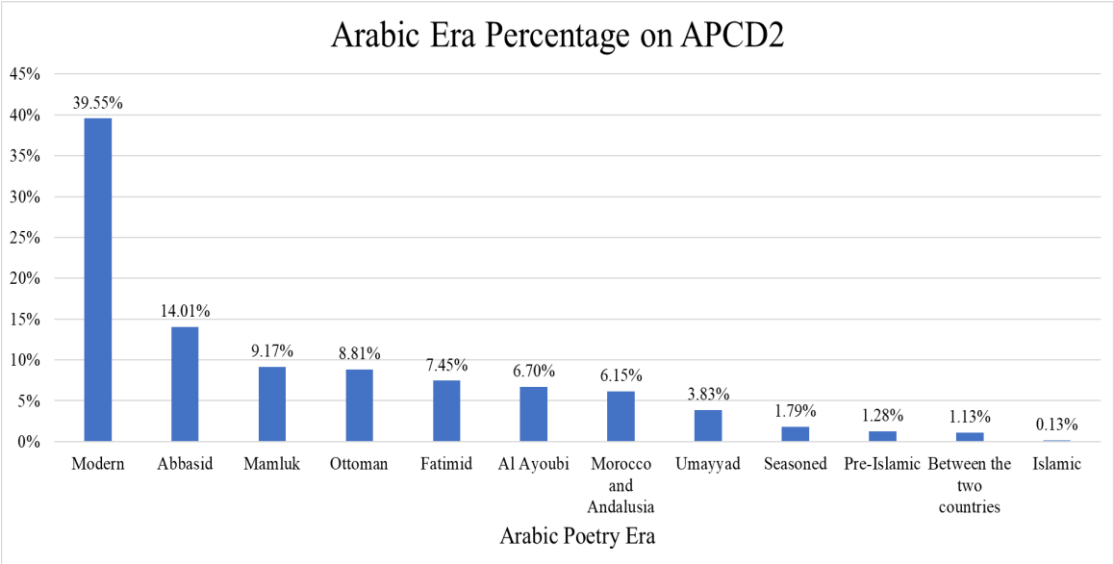

Figure 7. The Distribution of Poetry Eras in the Dataset

Although the dataset plays a significant role in our modeling, there are several shortcomings that complicate our learning tasks, including: (a) the absence of diacritics; (b) the presence of more than two diacritics on a single character; and (c) inconsistent Unicode representation. In the next subsection, we outline data preparation to overcome these issues.

4.2.1 Preparing APCD2 For Training

In the data preprocessing phase, we began by excluding the prose dataset, which represents 1.73% of the total APCD2. Subsequently, we shuffled the APCD2 dataset to ensure that each mini-batch during training contains a diverse mix of data samples, thereby improving generalization performance. Also, we remove diacritics from the available dataset to create input. The following table provides examples of input and target entries.

Table 8. A Poetry Example of Input and Target

Input	Target
وهم كلفوها الرمل رمل معبر تقول أهذا مشي حرد تلقف	وَهُم كَلَّفوها الرَملَ رَملَ مُعَبِّرٍ تَقولُ أَهَذا مَشيُ حُردٍ تَلَقُّفُ

Finally, we derived two datasets using the methodology outlined in the paper by Abandah *et al*. (2022), which categorizes them based on the diacritics-to-letters ratio to address the initial issue we mentioned earlier. Thus, as presented in the table below, we introduce three datasets: APCD2, DS1, and DS2. This table offers a comparative summary of these datasets, including their sizes, diacritics-to-letter ratios, and average ratios.

Table 9. Characteristics Of the APCD2 Dataset And Two Datasets Derived from It

Criteria	APCD2	DS1	DS2
Size (KB)	437 MB	36.9 MB	8.71 MB
Number of samples	1,467,119	313,325	76,034
Diacritics to letter ratio	All	≥ 0.50	≥ 0.667
Average ratio	0.27	0.61	0.75

Thus, this study employs both datasets for training and testing the model: the Clean-400 dataset, along with the training set proposed by Karim *et al*. (2021), and the APCD2 dataset developed by Abandah *et al*. (2020).

4.3 Experiments

Our model underwent experimentation with several variable modifications throughout the process, including:

- Model Size Modification: We altered the model's size to observe its impact on performance.
- Dataset Variation: Different datasets were utilized for training and evaluation to gauge the model's adaptability to varying data sources.
- Training Steps Adjustment: The number of training steps was tweaked to optimize both model training and performance.
- Learning Rate Variation: We adjusted the learning rate to fine-tune the training process and improve model convergence.

- Optimizer Type Change: Experimentation with different optimizers was conducted to assess their effectiveness in efficiently training the model.

4.3.1 ByT5 Model Selection

We selected the small and base size of the ByT5 model, guided by Al-Rfouh *et al.* (2023) findings, which suggest that base models yield better results for Arabic diacritization tasks than small size. In the table below, we outline the specifications of the ByT5/small and ByT5/base models.

Table 10. ByT5 Model Specifications

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

4.3.2 Byt5 Model Hyperparameters

Training hyperparameters are crucial for adjusting the weights of artificial neural networks. The Byt5 primary hyperparameters include:

- Batch Size: This parameter indicates the sample number processed before weight updates occur. While values of 256 or higher are typical, larger batches may exhibit diminishing returns in terms of performance improvement.
- Maximum Sequence Length: it refers to the limit on the number of bytes permitted in a sample. Two key considerations arise: firstly, the *transformer* model's time complexity. Secondly, our model operates at byte granularity, requiring more tokens to represent the same text compared to word-level models. Consequently, the maximum sequence length for the ByT5 model is capped at 1024 tokens.

- Learning rate (η): it regulates how much parameters of the model are adjusted throughout the optimization process. Smaller values of η lead to slower adjustments to the model's weights, promoting gradual convergence and ensuring training stability. In contrast, higher learning rates result in larger weight updates, speeding up the training process but increasing the likelihood of weight oscillations, which can lead to training instability and higher final loss. Optimal values, such as 0.001 for the T5 family with the Adafactor optimizer, ensure a smooth convergence process.
- Optimizers: the choice of optimizers plays a crucial role in the efficiency and effectiveness of the training process. The primary optimizers used in our model are:

1- Adam Weight Decay Optimizer: It is an optimizer built on the Adam algorithm and adds a function to reduce L2 weight. It uses Adam as a base and adds an L2 weight reduction function to avoid overfitting the model.

2- Adafactor Optimizer: It is the primary optimizer used in pre-trained models based on the Adafactor algorithm. It provides a way to automatically set learning rates based on parameter size and performs well for a variety of models.

4.3.3 Experimental Frameworks and Platforms

This section outlines the experiment platform. The proposed models are employed using two platforms: Google cloud platform (GCP) and Google colab pro plus. We partition the training process across multiple" slices" of TPU v2 units available in Colab Pro Plus. TPU v2 units are part of a set of interconnected units via a high-speed 2D mesh network, supported by host machines running on the central processing unit (CPU). The remaining details are outlined in the table below.

Table 11. Colab Pro Plus Platform Computing Environment

Hardware	Specification
CPU	Intel(R) Xeon(R) CPU @ 2.20GHz, 4 cores, 56 MB cache
TPU	v2
GPU	Nvidia-Tesla V100-SXM2 @ 1.32GHz, 16 GB Nvidia-Tesla P100-PCIE @ 1.32GHz, 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

To facilitate parallel processing of both models and data, we utilize the Mesh TensorFlow library. Additionally, we leverage cloud memory to store the data and models trained on specific tasks.

4.3.4 Byt5 Training

We adopt the methodology outlined in Al-Rfooh *et al*. (2023) to train our text diacritization model. Our poetry diacritization model undergoes three phases of training to predict diacritics in verses based on the input, which is without diacritics.

From the following table, it can be concluded that the ByT5 model utilizes 2 bytes to represent Arabic letters and diacritics, predicting an expectation of 8 classes in the model. However, the combinations of diacritics, such as *shadda* and *fatha*, are treated individually.

Table 12. Predicted Diacritics and Their Ids in The Byt5 Model

The Predicted Diacritics	Diacritics ID	The Predicted Diacritics	Diacritics ID
No diacritics	-	*Kasratan*	[220, 144]
Fatha (◌َ)	[220, 145]	*Shadda* with *Fatha*	[220, 148, 220, 145]
Damma (◌ُ)	[220, 146]	*Shadda* with *Damma*	[220, 148, 220, 146]
Kasra (◌ِ)	[220, 147]	*Shadda* with *Kasra*	[220, 148, 220, 147]
Shadda (◌ّ)	[220, 148]	*Shadda* with *Sukun*	[220, 148, 220, 149]
Sukun (◌ْ)	[220, 149]	*Shadda* with *Fathatan*	[220, 148, 220, 142]
Fathatan (◌ً)	[220, 142]	*Shadda* with *Dammatan*	[220, 148, 220, 143]
Dammatan (◌ٌ)	[220, 143]	*Shadda* with *Kasratan*	[220, 148, 220, 144]

datasets DS1 and DS2 are divided into 85% for training and 15% for testing, consistent with previous studies (Abandah *et al*., 2020). We exclusively utilize the training data for training, progressing through a series of checkpoints. During each checkpoint, 85% of the training data is used to train the model, while the remaining 15% is used to assess its learning progress. Subsequently, after each checkpoint, we evaluate the model's ability to predict diacritics accurately based on the validation data. Additionally, we employ dropout during training to enhance the model's performance. To assess the network's performance, we employ the weights obtained from the most optimal step to predict diacritics for the test dataset.

ByT5 implements a step-by-step training process where model parameters are updated based on batches of data. Checkpoints are strategically utilized to save the model's state at intervals, including weights and parameters, for the purpose of

resuming training or evaluating performance. In our training, the model underwent 43,100 steps to achieve optimal weights.

Mini-batch training was utilized, where each training step comprises a batch of 256 verses. This batch size was determined through experimentation, leading to faster training and improved outcomes. Additionally, a learning rate of 0.003 was employed, also determined through experimentation. Furthermore, two optimizers were conducted for training the network parameters. Among these, the Adam Weight Decay Optimizer demonstrates superior accuracy compared to Adafactor (Kingma and Ba, 2014).

4.3.5 Post-Processing of Output

In the post-processing stage, we implement corrections to address certain errors in the model's output, focusing on adhering to Arabic diacritic usage rules. Similar to Abandah *et al*. (2015), our approach involves *Sukun* correction and *Fatha* correction.

Sukun correction involves the removal of *Sukun*, which denotes the absence of a short vowel. Its presence or absence is not considered an error. On the other hand, in *Fatha* correction, any diacritic other than Fatha preceding *Alef*, *Alef Maksura*, and *Teh Marbuta* is corrected to *Fatha*.

4.3.6 Performance Evaluation

We evaluate diacritic restoration capabilities using the metrics below, which capture their performance.

- Accuracy is a common metric for evaluating NLP and speech recognition tasks, which measures the percentage of correct predicted diacritics in all outputs. Higer accuracy indicates that the model is better at reliably predicting diacritics. While lower accuracy indicates that the model struggles with diacritic prediction.
- Diacritics Error Rate (DER): is a percentage of incorrect Arabic letter diacritics predicted to all diacritics count.

- Word Error Rate (WER): is a percentage of incorrectly predicted diacritic words compared to target verses. The presence of one diacritic error in the predicted word is sufficient for it to be considered incorrect.

When evaluating the poetry model, we customized DER and WER metrics. Due to variations in the ratio of diacritics to letters across verses in the dataset, the model predicts additional diacritics on letters that were originally undiacritized. The model sometimes predicts additional diacritics on originally undiacritized letters. When applying DER/WER to evaluate the poetry diacritization model, it inaccurately considers these additional diacritics as errors. Hence, we chose to exclude the originally undiacritized letters from the dataset when computing DER/WER.

Chapter Five

Experiments and Results

5.1 Introduction

In this chapter, we will outline the experiments conducted for both prose and poetry diacritization task. Next, we will present the main results, analyzing the overall performance, accuracy, DER and WER of the models. We will also explore the effect of model size on performance, examining the balance between accuracy and complexity.

Following this, we will analyze the time efficiency of the models, investigating the time required for predictions and ways to optimize it. We will then provide a comprehensive comparison of the different models, highlighting their strengths and weaknesses. Finally, we will analyze prediction errors, exploring their potential causes and how to mitigate them for more accurate and reliable results.

5.2 Experiments

In this section, we will detail the experiments conducted to evaluate the performance of our diacritization models. Two primary models were developed: one for prose diacritization and another for poetry diacritization. We will discuss the performance of each model in detail.

5.2.1 Prose Diacritization Model

Our diacritization model successfully applied diacritization to Arabic text. In the first experiment, we used the pretrained Small ByT5 model with a maximum sequence length of 1024 for input and 2048 for target.

In order to determine the optimal steps containing the best weights for evaluating the test set based on accuracy metrics, we compared results across checkpoints up to

25k, as depicted in Figure 8. We observed stabilization after the 18k checkpoint, with no further accuracy improvements for the validation set. Notable results were achieved, including a peak accuracy of 51.78% at the 18,000th checkpoint. The best DER on the validation set was recorded at 0.81%, along with a corresponding WER of 2.46%.

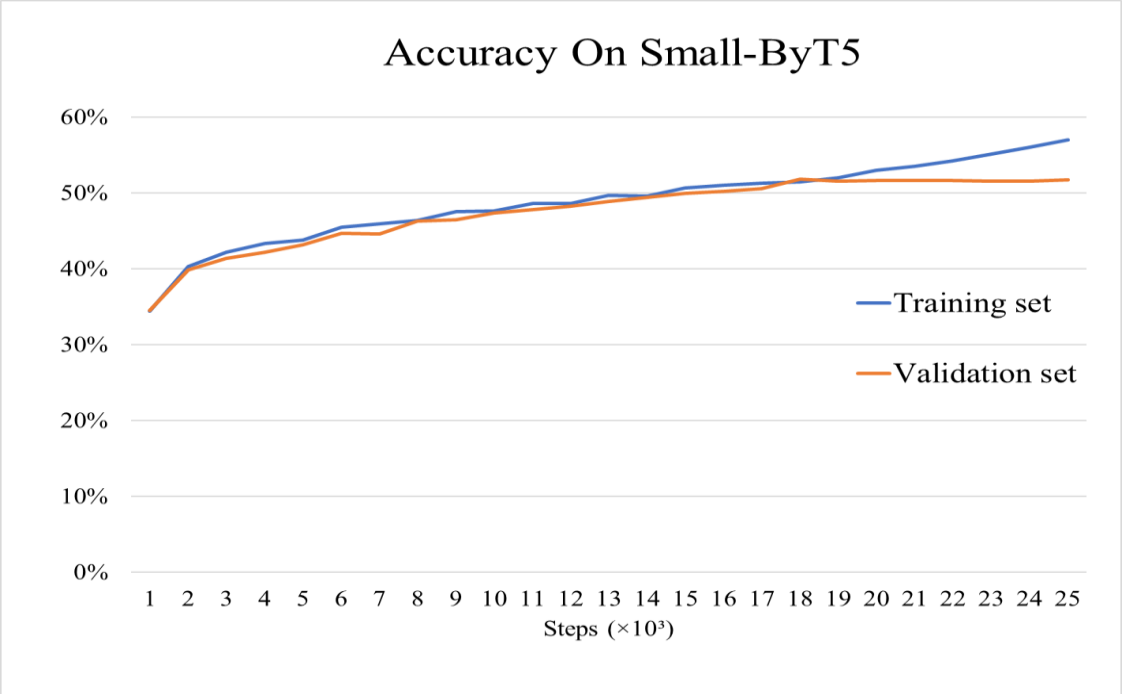

Figure 8. The Accuracy Throughout the Training of The Small Diacritization Prose Model

Notable results were achieved, including a peak accuracy of 51.78% at the 18,000th checkpoint. The best DER on the validation set was recorded at 0.81%, along with a corresponding WER of 2.46%.

5.2.2 Poetry Diacritization Model

Our diacritization model successfully applied diacritization to Arabic poetry These assessments were conducted using a sequence length of 250 for the input and 400 for the target.

We performed an assessment of the six experiments by training small Byt5 on the clean-400 train set, as outlined in the preceding subsection, DS1 and DS2, and

subsequently evaluating their diacritic and WER using the DS2 test set. As elaborated in Chapter 4, the Byt5 model undergoes training in one phase, two phases, and three phases. Table 13 encapsulates this evaluation, providing details on the dataset trained in each experiment and the accuracy of the validation set on DS2, DER, and WER.

Table 13. Comparison Of Experiments Trained Using Different Methods and Evaluated

Phase 1	Phase 2	Phase 3	Accuracy	DER	WER
DS1	-	-	9.48%	4.18%	14.02%
DS1	DS2	-	12.39%	3.45%	11.74%
DS2	-	-	8.37%	4.55%	14.53%
Clean_400	DS1	-	9.67%	4.08%	13.66%
Clean_400	DS2	-	8.79%	4.41%	14.13%
Clean_400	**DS1**	**DS2**	**12.56%**	**3.42%**	**11.20%**

After examining the outcomes of the six experiments, we determined that the three-phases experiment produced the best results, demonstrating the best weights for evaluating the DS2 test set.

5.3 Results

We present the outcomes of our diacritization experiments on both Arabic prose and poetry texts. Each task involved distinct model configurations and optimization strategies aimed at achieving optimal performance in handling Arabic text diacritization.

Our prose diacritization base model successfully applied diacritization to Arabic text. Through a comparison of results across checkpoints up to 25k, as depicted in Figure 10, we observed stabilization after the 19k checkpoint, with no further accuracy improvements for the validation set. At that checkpoint the model achieved notable results, including a peak accuracy of 55.30%. The DER and WER values obtained from the model's evaluation on the Clean-400 validation set at the 19th checkpoint step were

0.74% and 2.29%, respectively. We achieved the best result at the same checkpoint for the Clean-400 test set, with a DER of 0.86% and a WER of 2.64%.

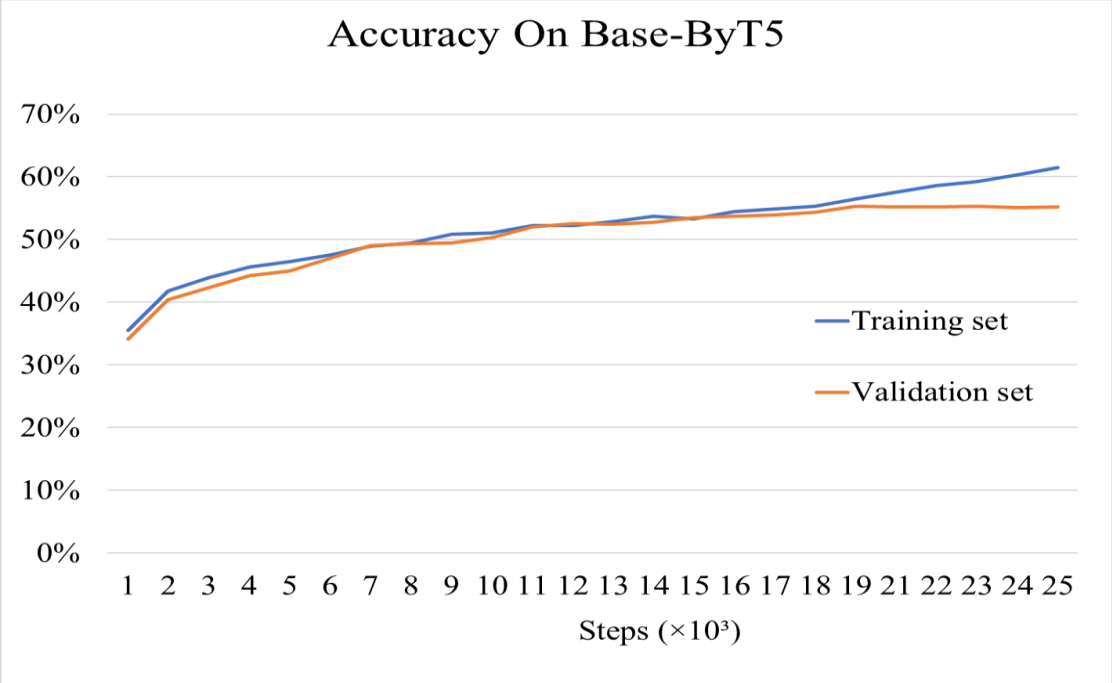

Figure 9. The Accuracy Throughout the Training of The Base Diacritization Prose Model

In diacritizing Arabic poetry, as discussed in the previous section, the three-phases experiment yielded the optimal results, demonstrating the best weights for evaluating the DS2 test set. Figure 11 illustrates the training progress of the ByT5 Small model across the three-phases, demonstrating the DER of the validation set at each training step.

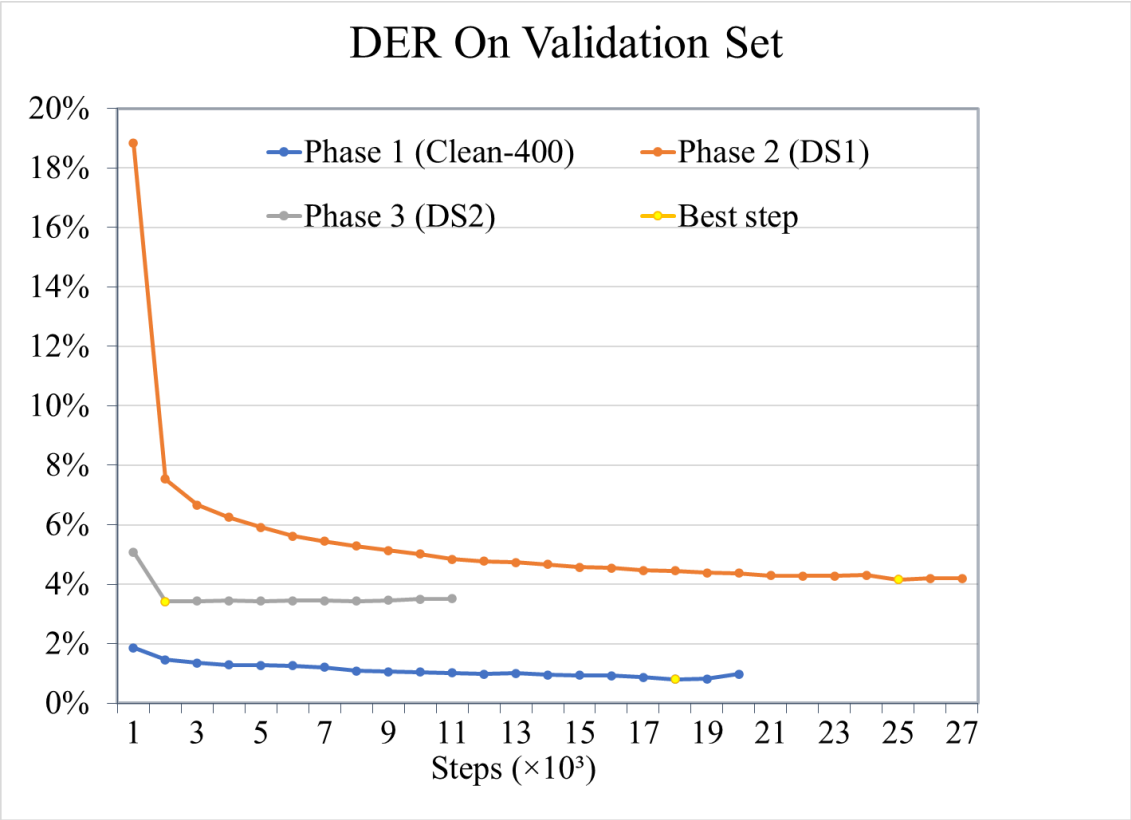

Figure 10. The DER Of the Poetry Diacritization Model Across Validation Sets for The Three Datasets Throughout Training

As shown in Figure 11, phase 1, which initiates the training process, is previously discussed. During phase 2, the model is trained on the DS1 train set, reaching its peak accuracy of 9.67% at step 25,000. However, step 25,000 also records the highest DER at 4.28%, resulting in the conclusion of training at step 26,000. Phase 3 involves the retraining of all layers on the DS2 training set, leading to a notable improvement in DER. The model swiftly adjusts to DS2 within 100 steps, with step 100 displaying the highest DER at 3.42% and WER at 11.20% and training concluding at step 400. Optimal model weights are achieved at the 43,100th checkpoint step, where evaluation on the DS2 test set yields a DER of 3.28% and a WER of 10.74%.

5.4 Effect of Model Size

we employed both the small and base ByT5 models to train on text and poetry datasets, aiming to assess how variations in model size influence performance.

Our analysis Showed that while the Base ByT5 model demonstrated a slightly lower DER of 0.74% on the Clean-400 dataset compared to the Small ByT5 model's 0.81%, as shown in figure 12, And the small ByT5 model exhibited a lower DER of 3.42% compared to the base ByT5 model's 3.45% for Arabic poetry context. These results emphasize the necessity of considering both model size and dataset characteristics when optimizing text diacritization models for varying linguistic tasks.

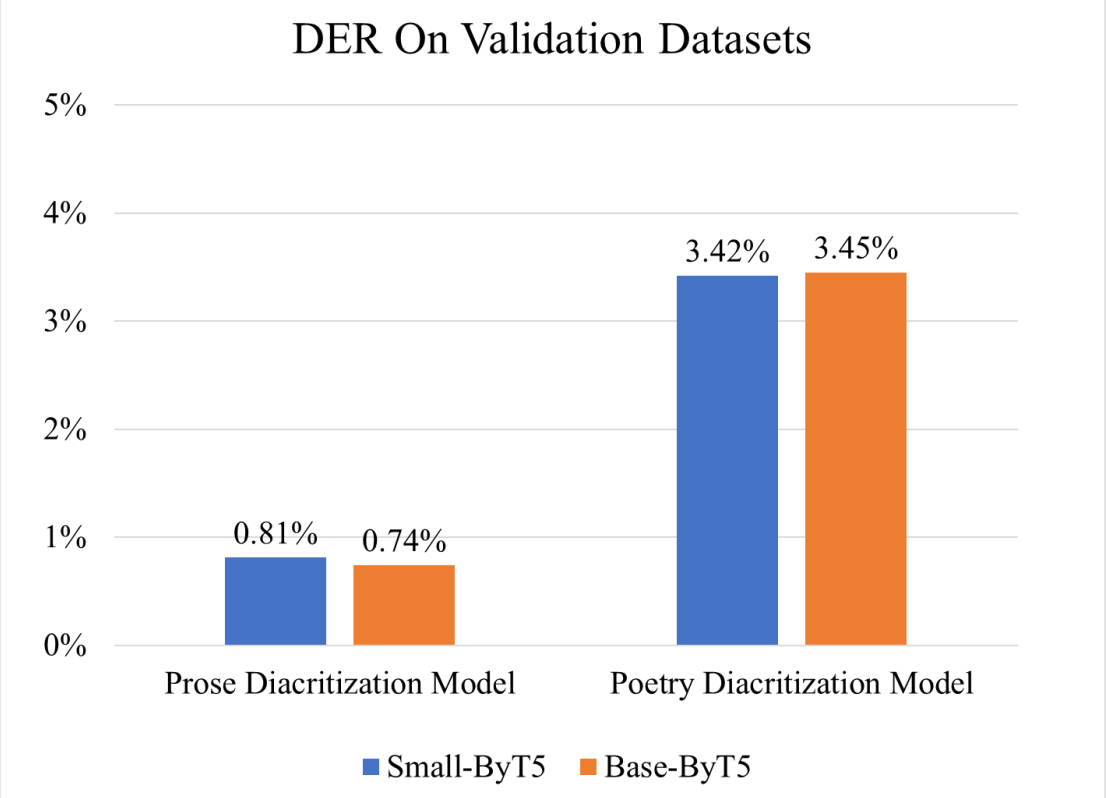

Figure 11. The Effect of Model Size on Different Tasks

5.5 Time Analysis of Models

When selecting the most suitable model for diacritizing Arabic text and poetry, it is crucial to factor in the time required for model training. Our training methodology involved saving training weights at checkpoints every 1000 steps, allowing us to measure the time investment for each training phase. The figure below presents a detailed analysis of the training duration, covering the Small Prose model, the Base Prose model, the Small Poetry model and the Base Poetry model.

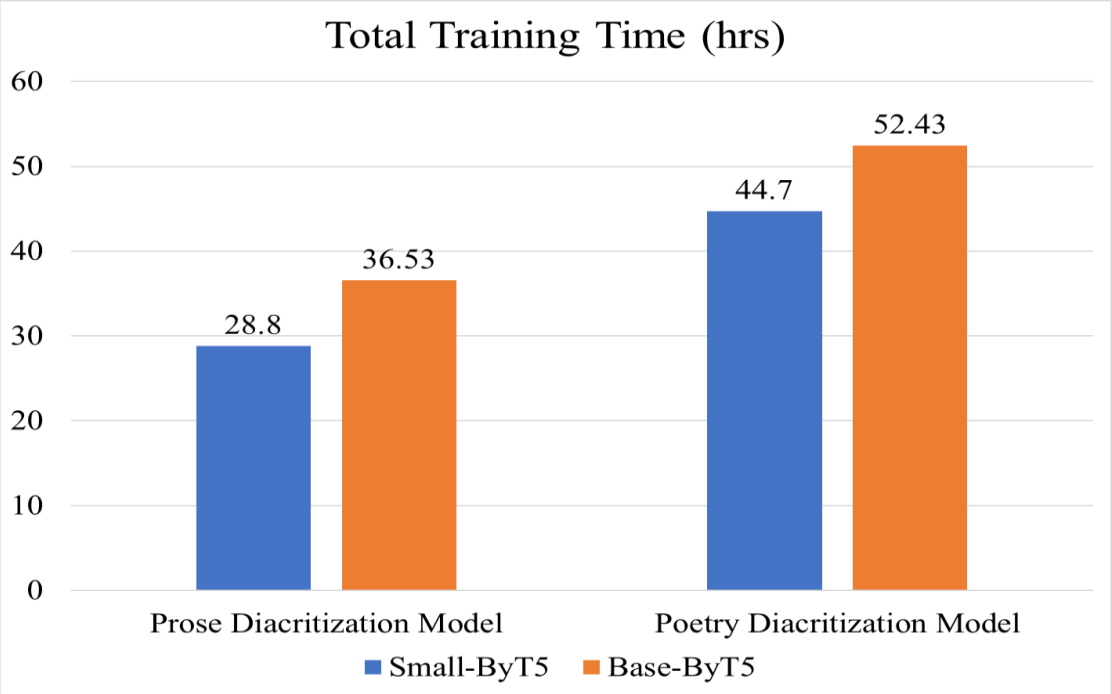

Figure 12. Analysis of Training Time for Diacritization Models

For instance, the small prose diacritization model required 1.36 hours for every 1000 steps. Since the optimal checkpoint is set at 18,000, every step took approximately 96 × 60 × 60 / 1000 = 5.76 seconds per step. To calculate the total training time, we multiply the number of steps by the time per step: 18,000 × 5.76 / 60 / 60, resulting in a total training time of 28.8 hours.

During the prediction phase, the prose diacritization model demonstrates a satisfactory response time, managing to diacritize each test sequence in 0.13 seconds.

Although the base model requires one and a quarter time longer training for prose datasets compared to the small model, it demonstrates a 10.5% improvement in performance. However, for poetry datasets, the small model achieves comparable results to the base model. As a result, we will employ the base model for prose dataset testing and utilize the small model for diacritizing poetry datasets.

5.5 Comparison

We will now compare our results with the state-of-the-art results for diacritical Arabic text and poetry models.

Karim and Abandah (2021) utilized the extensive Tashkeela dataset to build a cleaned corpus with about 550k sequences. In order to convert undiacritized Arabic letter sequences into fully diacritized ones, their baseline model incorporates a deep BiLSTM RNNs. Their method yielded an average WER of 3.89% and a DER of 1.45%.

Al-Rfooh *et al*. (2023) suggested the use of fine-tuning token-free pre-trained multilingual models, notably ByT5, to acquire the ability to predict and add missing diacritics in Arabic text. They conducted two phases of experimentation, utilizing Tashkeela's dataset in the first phase and the Clean-400 dataset in the second phase, with both small and base sizes of the ByT5 and Adafactor optimizers. The methods they provide demonstrate a significant decrease in diacritization error rate when compared to the most successful previous findings, the detailed results are presented in the following table.

Abandah *et al*. (2022) handle the problem of adding diacritics to Arabic poetry verses by leveraging pattern characteristics from a pre-trained classification model via transfer learning. Furthermore, they address the scarcity of training data by training their diacritization model in multi-phase using the second version of the APCD2. When

compared to the most successful previous outcomes, the solutions they propose yield a significant improvement in the diacritization error rate, reducing it from 6.08% to 3.54%, which corresponds to a 42% enhancement.

Our study explores the application of ByT5 for diacritizing two types of Arabic text: prose and poetry. For Arabic prose diacritization, our model includes two sizes, small and base, both employing the Adam weight decay optimizer. The base size model outperformed the small size, yielding a DER of 0.86% and a WER of 2.64%. In comparison to the findings of Karim *et al*. (2021), our approach resulted in a notable reduction in DER from 1.45% to 0.86% and WER from 3.89% to 2.64%, reflecting a significant DER enhancement of 41% when evaluated on the Tashkeela test set. Additionally, in contrast to the outcomes reported by Al-Rfooh *et al*. (2023), when they trained ByT5 base using two phase training approach, our utilization of the Adam weight decay optimizer coupled with a single-phase training strategy led to superior performance for prose diacritization tasks on the test Tashkeela set. Specifically, our approach achieved a reduction of 14% in DER and 10% in WER.

As for the Arabic poetry diacritization model, it was trained using six experiments, each trained at different stages and datasets. Additionally, two sizes of the model, small and base, were utilized. The results obtained from the Clean-400 training phase are used as input for subsequent training stages on the APCD2 with the small size of ByT5. Evaluation of the model's performance was carried out on the DS2 test set, resulting in a DER of 3.28% and a WER of 10.74%. our approach resulted in a slight reduction in DER from 3.54% to 3.28% and WER from 12.34% to 10.74%, reflecting a DER enhancement of 7% and a WER enhancement of 13% when evaluated on the Tashkeela test set.

Table 14. Diacritics Restoration Results Comparison

Authors	Data Used	No Phases	Model Size Used	Evaluation Set	DER	WER
Abandah *et al*. (2022)	DS1, DS2	Two	TL-All	DS2 test	3.54%	12.34%
Karim and Abandah. (2021)	Clean-400	One	BiLSTM	Tashkeela test	1.45%	3.89%
Al-Rfooh *et al.* (2023)	Tashkeela Clean-400	Two	Base	Tashkeela test	1.00%	2.92%
Our Work	**Clean-400**	**One**	**Base Byt5**	**Tashkeela test**	**0.86%**	**2.64%**
	Clean-400, DS1, DS2	**Three**	**Small Byt5**	**DS2 test**	**3.28%**	**10.74%**

5.5 Discussion

As the T5 family typically employs a learning rate of 0.001, we opted for a higher learning rate of 0.003 in our study. To investigate the impact of learning rate on the outcomes, a final experiment was conducted. The Poetry diacritization model was re-trained using learning rates of 0.0003, 0.001, 0.003, and 0.01 with the DS2 dataset, respectively. The results obtained from the DS2 test are illustrated in Figure 13. A learning rate of 0.003 yields superior outcomes compared to others.

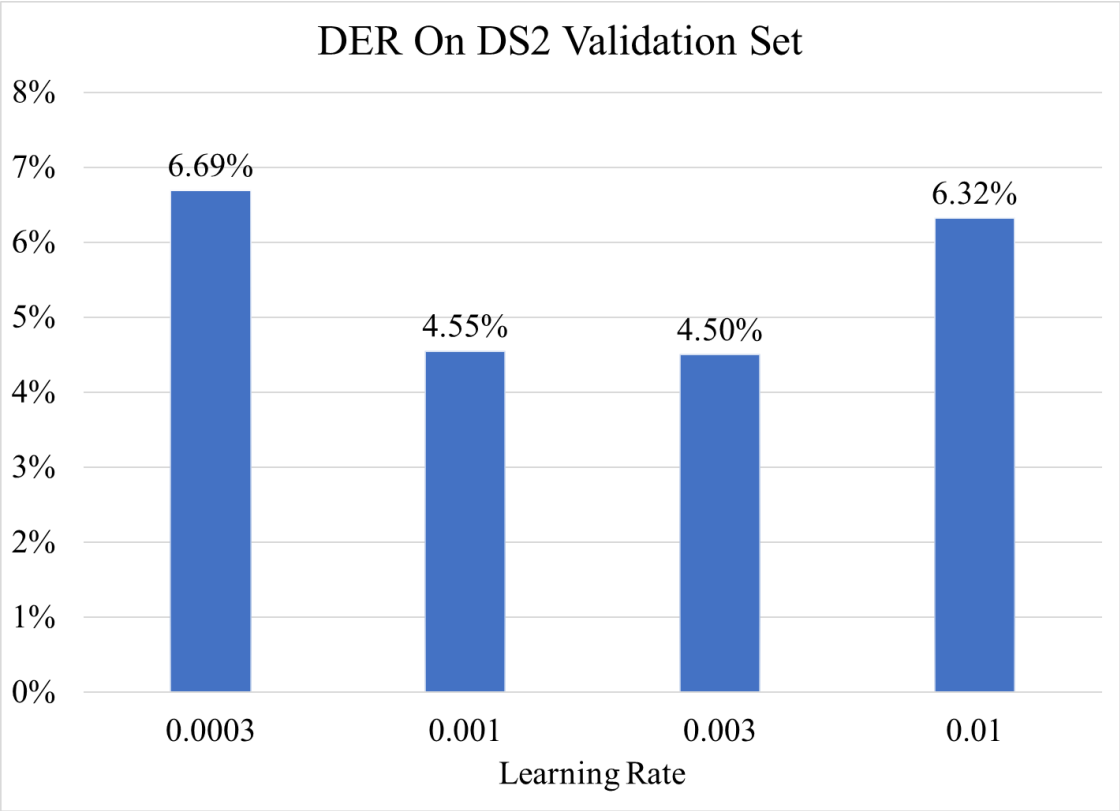

Figure 13. Impact of Different Learning Rates on DER During DS2 Training

As the T5 family typically employs a train batch size of 128, we opted for a higher train batch size of 256 in our study. To investigate the impact of batch size on outcomes, a final experiment was conducted. The poetry diacritization model was re-trained using batch sizes of 128 and 256 with the DS2 dataset, respectively. The results obtained from the DS2 test are illustrated in Figure 15. A train batch size of 256 yields superior outcomes compared to others.

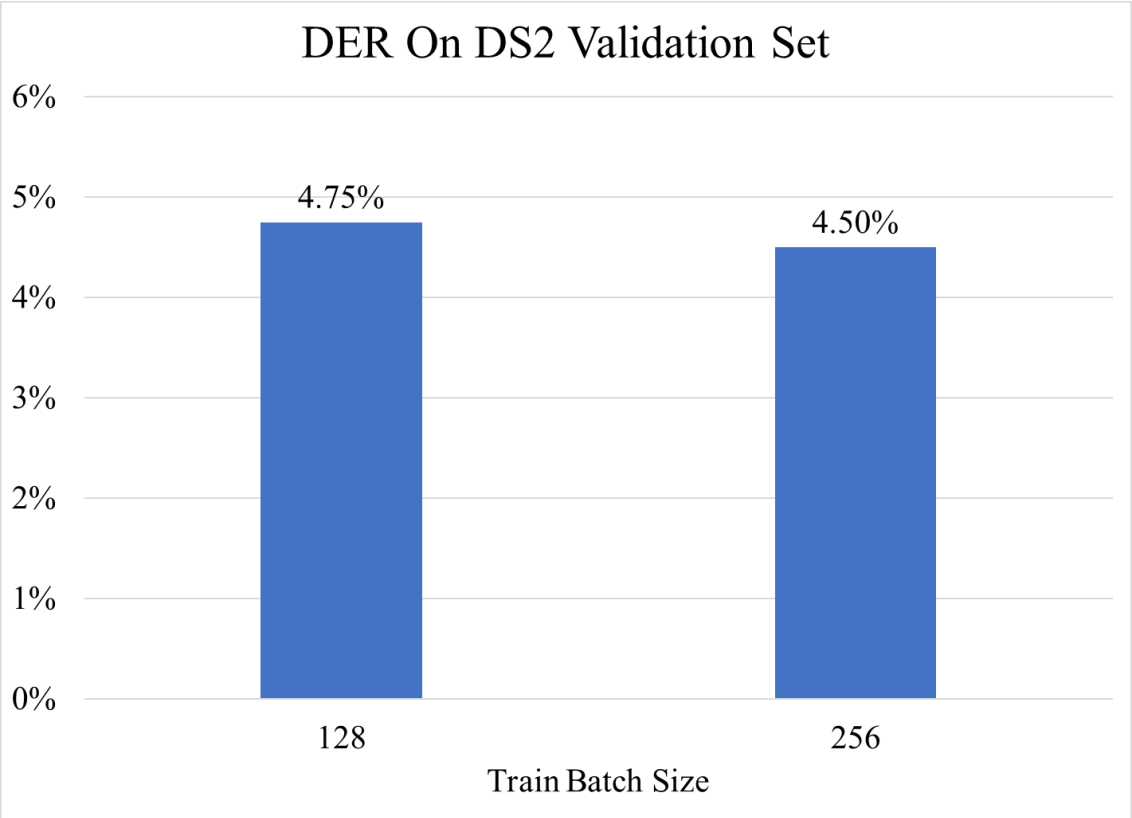

Figure 14. Impact of Different Train Batch Size on DER During DS2 Training

Considering that Abandah *et al*. (2022) utilize a maximum sequence length of 256, whereas in our study, we employ a sequence length of 400 for the output, we conducted a concluding experiment to assess the impact of sequence length on the results. We proceeded to re-train the poetry diacritization model using maximum sequence lengths of 256 and 400. The outcomes of the DS2 test are illustrated in Figure 15. a sequence length of 400 yields superior outcomes compared to another.

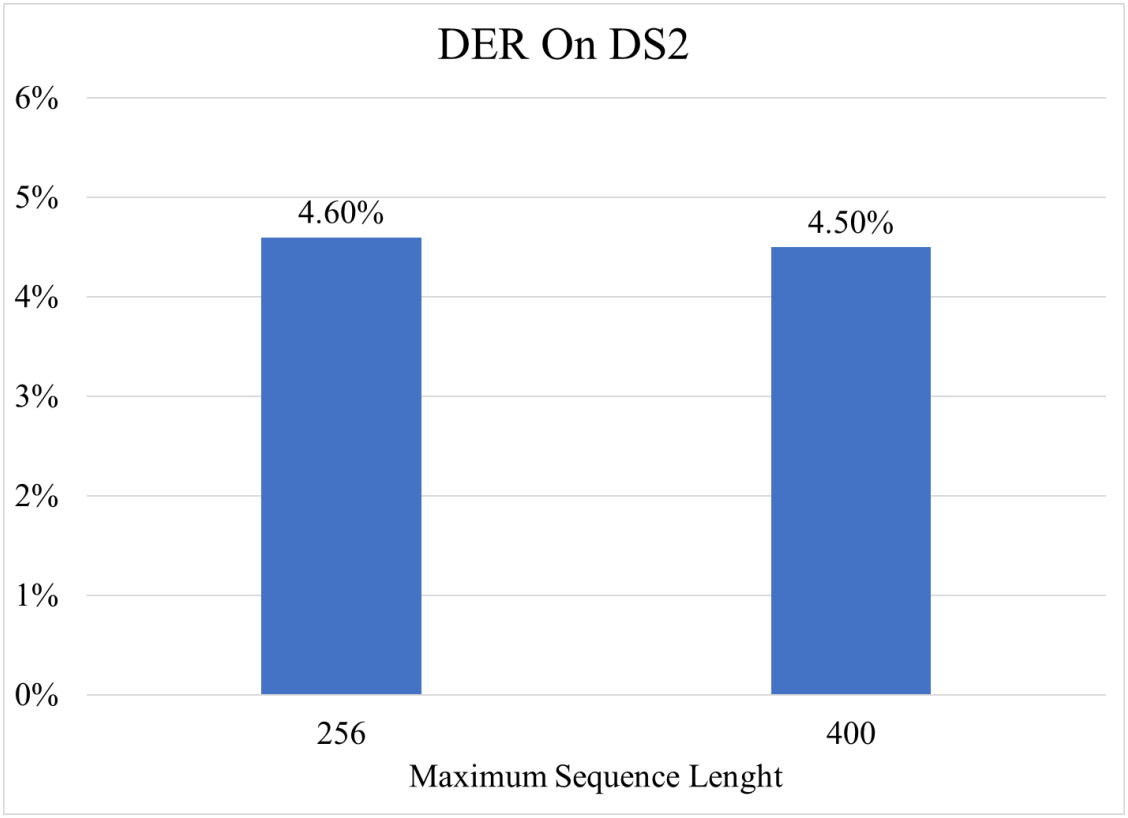

Figure 15. Impact of Different Maximum Sequence Lengths on DER During DS2 Training

In the context of diacritizing Arabic poetry, the superior performance of the small model size over the base model size lies in the relatively smaller and less complex nature of the dataset. This reduced dataset size enables the small model to more effectively capture and interpret the unique linguistic nuances and structures prevalent in Arabic poetry. With fewer data points to process, the small model can allocate its learning capacity more efficiently, allowing it to better handle the complexities and subtleties of poetic forms.

5.6 Predictions' Errors

In this section, we analyze the performance of our diacritization model in predicting Arabic diacritics for both prose and poetry sentences. We compare the model's predictions to the target sentences to assess its accuracy and effectiveness.

5.6.1 Arabic Prose Diacritization

When comparing the predicted word "يَعُدْ" in the table below with the target word "يُعِدْ," it confirms a prior inference by researchers: The ByT5 model relies on understanding texts and understanding the relationship between the word's position and the diacritic mark. So, if the proposed model understands the word differently, it will also add diacritics differently, leading to errors in more than one diacritic mark in the same word.

Table 15. An Example of Prose Model's Predictions

	Example
Arabic prose sentence	ولم يعد وجب عليه لكل شوط نصف صاع
Target	وَلَمْ يُعِدْ وَجَبَ عَلَيْهِ لِكُلِّ شَوْطٍ نِصْفُ صَاعٍ
Our model prediction	وَلَمْ يَعُدْ وَجَبَ عَلَيْهِ لِكُلِّ شَوْطٍ نِصْفُ صَاعٍ

5.6.2 Arabic poetry diacritization

When comparing the predicted word "بَنِيهِ" with the target word "بَنِيهِ ", and the predicted word "يَهْفُو" with the target word" يَهْفو ", there are additional diacritics that the model predicted, so we use custom DER and WER evaluation metrics as we mentioned in chapter 4.

In the second example, when comparing the predicted word "تَنِبِذُهُ" with the target word" تُنَبِذُهُ ", the model commits similar errors to those observed in Arabic prose, but with a greater degree. Arabic poetry composition presents a challenge for linguistic and computational models due to its linguistic complexity and poetic diversity, requiring extensive study and a profound understanding of the Arabic language and culture, as

previously discussed. The model requires a sizable dataset with a high occurrence of diacritics to achieve significant enhancements.

Table 16. Examples of Poetry Model's Predictions

	Examples
Arabic poetry input	ودهرك دع بنيه إليك يهفو بطاعة مستبين الرق عبد هلا غضبت لرحل جا رك إذ تنبذه حضاجر
Arabic poetry target	ودَهْرُكَ دَعْ بَنِيهِ إليكَ يَهْفو بِطاعَةِ مُسْتَبِينِ الرِّقِّ عَبْدِ هَلّا غَضِبتَ لِرَحلِ جا رِكَ إذ تُنَبِّذُهُ حَضاجِر
Our model prediction	وَدَهْرُكَ دَعْ بَنِيهِ إِلَيْكَ يَهْفُو بِطَاعَةِ مُسْتَبِينِ الرِّقِّ عَبْدِ هَلّا غَضِبتَ لِرَحلِ جا رِكَ إذ تَنبِذُهُ حَضاجِر

Chapter Six

Conclusion and Future Work

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

6.1 Conclusion

We conducted a study aiming to develop a precise solution for automatic diacritization of Arabic text. Our investigation involved the utilization of pre-trained models, notably the ByT5 model. We examined two variations of the ByT5 model size, both of which were trained on the Clean-400 dataset. We achieved the best result using the base ByT5 model, with a DER of 0.86% and a WER of 2.64%.

Despite the proficiency of the base ByT5 model in diacritizing Arabic text, it encounters challenges when tasked with diacritizing Arabic poetry. This difficulty stems from the linguistic and morphological complexity inherent in poetry, as well as the tendency of poets to deviate from conventional Arabic grammar and morphology when adding diacritics to their verses. Given the artistic nature of poetry, it may be more effective to utilize models specifically trained on poetic datasets rather than those trained on general Arabic prose. However, a significant difficulty arises from the scarcity of high-quality datasets specifically designed for Arabic poetry, making it challenging to train such specialized models.

We have presented another precise solution for automatically diacritizing Arabic poetry. We utilized the pre-trained ByT5 model, specifically a model employing transfer learning. This model benefits from a prose diacritization model trained on a large and clean dataset. Additionally, we explored training this model through multiple training phases on two datasets of diacritized poetry. Firstly, the model is trained on the DS1 dataset, which is larger than the DS2 dataset and contains a lower diacritic ratio.

Secondly, the model is trained on the DS2 dataset, which has a higher diacritics ratio, to enhance its ability to predict fully diacritized poetry. These solutions provide a model with a DER on the test set of 3.28%, and a WER of 10.74%.

6.2 Future Work

To enhance our diacritization models, future work could focus on several paths. Firstly, exploring larger and more diacritizing training datasets may help improve model performance, especially the datasets of general Arabic poetry. Moreover, considering the integration of advanced training techniques such as custom loss functions, this may contribute to further advancements in diacritization accuracy. Finally, efforts to extract our best-performing model into smaller, more efficient models suitable for deployment in edge-compute applications. This aims to broaden the practical applicability of our approach, allowing these models to be used in resource-constrained environments such as smart devices, sensors, and internet-connected devices that rely on processing data locally at the application site rather than sending it to the cloud.

References

Abandah, G. A., Graves, A., Al-Shagoor, B., Arabiyat, A., Jamour, F., & Al-Taee, M. (2015). Automatic diacritization of Arabic text using recurrent neural networks. **International Journal on Document Analysis and Recognition (IJDAR)**, 18, 183-197.

Abandah, G. A., Khedher, M. Z., Abdel-Majeed, M. R., Mansour, H. M., Hulliel, S. F., & Bisharat, L. M. (2022). Classifying and diacritizing Arabic poems using deep recurrent neural networks. **Journal of King Saud University-Computer and Information Sciences**, 34(6), 37753788.

Abandah, G. A., Suyyagh, A. E., & Abdel-Majeed, M. R. (2022). Transfer learning and multiphase training for accurate diacritization of Arabic poetry. **Journal of King Saud UniversityComputer and Information Sciences**, 34(6), 3744-3757.

Abandah, G., & Arabiyat, A. (2017, October). Investigating hybrid approaches for Arabic text diacritization with recurrent neural networks. **In 2017 IEEE Jordan conference on applied electrical engineering and computing technologies (AEECT)** (pp. 1-6). IEEE.

Al-Musawi, M. J. (2006). Arabic poetry: Trajectories of modernity and tradition. **Routledge**.

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

Boyle, D., & Kalita, J. (2023). Spatiotemporal Transformer for Stock Movement Prediction. **arXiv preprint arXiv**:2305.03835.

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. **arXiv preprint arXiv**:1810.04805.

El-Imam, Y. A. (2004). Phonetization of Arabic: rules and algorithms. **Computer Speech & Language**, 18(4), 339-373.

Fadel, A., Tuffaha, I., Al-Jawarneh, B., & Al-Ayyoub, M. (2019). Neural Arabic text diacritization: State of the art results and a novel approach for machine translation. **arXiv preprint arXiv**:1911.03531.

Gal, Y. A. (2002, July). An HMM approach to vowel restoration in Arabic and Hebrew**. In Proceedings of the ACL-02 workshop on Computational approaches to semitic languages**.

Levstik, S. and Keith C. (2005), Doing History: Investigating with Children in Elementary and Middle Schools, (1st ed.). New York: **Routledge**.

Graves, A., Mohamed, A. R., & Hinton, G. (2013, May). Speech recognition with deep recurrent neural networks. **In 2013 IEEE international conference on acoustics, speech and signal processing** (pp. 6645-6649). Ieee.

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).

Kharsa, R., Elnagar, A., & Yagi, S. (2024). BERT-Based Arabic Diacritization: A state-of-the-art approach for improving text accuracy and pronunciation. **Expert Systems with Applications**, 123416.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. **arXiv preprint arXiv**:1412.6980.

Kiran, S. K., Shashi, M., & Madhuri, K. (2022). Multi-stage Transfer Learning for Fake News Detection Using AWD-LSTM Network. **International Journal of Information Technology and Computer Science (IJITCS)**, 14(5), 58-69.

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

Lee, Y. S., Papineni, K., Roukos, S., Emam, O., & Awadalla, H. H. (2003, July). Language model based Arabic word segmentation. **In Proceedings of the 41st annual meeting of the association for computational linguistics** (pp. 399-406).

Madhfar, M. A. H., & Qamar, A. M. (2020). Effective deep learning models for automatic diacritization of Arabic text. **IEEE Access**, 9, 273-288

Madhfar, M. A. H., and Qamar, A. M. (2020), Effective Deep Learning Models for Automatic Diacritization of Arabic Text. **IEEE Access**, 9, 273-288.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. **arXiv preprint arXiv**:1802.05365, 2018.

Mubarak, H., Abdelali, A., Sajjad, H., Samih, Y., & Darwish, K. (2019, June). Highly effective Arabic diacritization using sequence to sequence modeling. **In Proceedings of the 2019 Conference of the North American Chapter of the**

Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers) (pp. 2390-2395).

Nassiri, K., & Akhloufi, M. (2023). Transformer models used for text-based question answering systems. **Applied Intelligence**, 53(9), 10602-10635.

Nassiri, K., & Akhloufi, M. (2023). Transformer models used for text-based question answering systems. **Applied Intelligence**, 53(9), 10602-10635.

Parikh, A. P., Täckström, O., Das, D., & Uszkoreit, J. (2016). A decomposable attention model for natural language inference. **arXiv preprint arXiv**:1606.01933.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., ... & Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. **Journal of machine learning research**, 21(140), 1-67.

Ristea, N. C., Anghel, A., & Ionescu, R. T. (2024). Cascaded Cross-Modal Transformer for Audio-Textual Classification**. arXiv preprint arXiv**:2401.07575.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986), Learning Representations by Back-propagating Errors. **Nature**, 323(6088), 533-536.

Shaalan, K. (2010). Rule-based approach in Arabic natural language processing. **The International Journal on Information and Communication Technologies** (IJICT), 3(3), 11-19. Heugel, M. (2010). U.S. Patent No. 7,665,979. Washington, DC: U.S. Patent and Trademark Office.

Shahzadi, I., Tang, T. B., Meriadeau, F., & Quyyum, A. (2018, December). CNN-LSTM: Cascaded framework for brain tumour classification. **In 2018 IEEE-EMBS Conference on Biomedical Engineering and Sciences (IECBES)** (pp. 633-637). IEEE.

Shazeer, N. (2020). Glu variants improve transformer. **ArXiv preprint arXiv**:2002.05202.

Stankevičius, L., Lukoševičius, M., Kapočiūtė-Dzikienė, J., Briedienė, M., & Krilavičius, T. (2022). Correcting diacritics and typos with a ByT5 transformer model. **Applied Sciences**, 12(5), 2636.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. **Advances in neural information processing systems**, 30.

Xue, L., Barua, A., Constant, N., Al-Rfou, R., Narang, S., Kale, M., ... & 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.

Xue, L., Constant, N., Roberts, A., Kale, M., Al-Rfou, R., Siddhant, A., ... & Raffel, C. (2020). mT5: A massively multilingual pre-trained text-to-text transformer. **arXiv preprint arXiv**:2010.11934.

Yousef, W. A., Ibrahime, O. M., Madbouly, T. M., & Mahmoud, M. A. (2019). Learning meters of Arabic and English poems with Recurrent Neural Networks: a step forward for language understanding and synthesis. **arXiv preprint arXiv**:1905.05700.

Zerrouki, T. (2014). Arabic corpora resources, Tashkila collection from the Arabic Al-Shamela library. **Google Scholar Google Scholar Reference**.

Zerrouki, T., & Balla, A. (2017). Tashkeela: Novel corpus of Arabic vocalized texts, data for auto-diacritization systems. **Data in brief**, 11, 147-151.

Zhang, D., Peng, Q., Lin, J., Wang, D., Liu, X., & Zhuang, J. (2019). Simulating reservoir operation using a recurrent neural network algorithm. **Water**, 11(4), 865.

Azim, A. S., Wang, X., & Sim, K. C. (2012, September). A Weighted Combination of Speech with Text-based Models for Arabic Diacritization. In **INTERSPEECH** (pp. 2334-2337).

Hifny, Y. (2012). Smoothing techniques for Arabic diacritics restoration. In **Proceedings of the 12th Conference Lang. Eng.(ESOLEC'12)** (No. 1, pp. 6-12).

Habash, N., & Rambow, O. (2007, April). Arabic diacritization through full morphological tagging. In **Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume**, Short Papers (pp. 53-56).

التشكيل الآلي للنصوص والأشعار العربية باستخدام نماذج اللغة من حرف إلى حرف والتدريب متعدد المراحل

إعداد

روزان علي يونس

المشرف

الأستاذ الدكتور غيث علي عبندة

ملخص

تقتضي عملية التشكيل اليدوية للنصوص العربية جهدًا كبيرًا وفهمًا عميقًا للبنية الصرفية في اللغة العربية، في حين أن الأنظمة الآلية الحالية غالبًا ما تعاني من مشاكل الدقة والكفاءة، متطلبة بيانات تدريب وموارد حسابية كبيرة. يظهر نموذج البايت إلى البايت، وهو نوع من المحولات النصية إلى نصوص، كحلا واعدا لتشكيل النصوص العربية والشعر ضمن حقل معالجة اللغات الطبيعية. تهدف هذه الرسالة إلى معالجة هذا التحدي، مركزة بشكل خاص على الشعر، حيث تشكل قواعد بيانات الشعر العربي المتوفرة غير المشكولة أو المشكولة جزئيًا تحديًا كبيرًا. الاستفادة من معالجة النموذج على مستوى البايت وميزات تقسيم النصوص تلقائيًا، جنبًا إلى جنب مع استراتيجية التدريب متعددة المراحل، أمر ضروري للتغلب على نقص بيانات التدريب.

تبدأ الدراسة بالتدريب على مجموعة بيانات Clean-400 لإنشاء نموذج لتشكيل النصوص النثرية، ثم تقييمه على مجموعة بيانات الاختبار Tashkeela بعد ذلك، ثم تدريب نموذج تشكيل الشعر على مجموعات البيانات DS1، و DS2، باستغلال التمثيلات والأوزان المتعلمة من نموذج النثر. مراحل التدريب على مجموعات البيانات DS1وDS2 ، تهدف إلى تحسين دقة التنبؤ للشعر المشكول بالكامل.

قمنا بفحص نماذج من الحرف الى حرف (ByT5) ، بأحجام الصغير والأساسي، واختبار المحسنات Adam و Adafactor لتعزيز أداء النموذج. تظهر النتائج أن النموذج الأساسي مع المحسن Adam يقدم أفضل أداء لتشكيل النصوص، بينما يتفوق

النموذج الصغير مع المحسن Adam في التشكيل الشعري. تحقق الحلول المقترحة معدل خطأ في العلامات التشكيلية (DER) مقداره 0.86٪ ومعدل خطأ الكلمات (WER) مقداره 2.64٪ لتشكيل النصوص النثرية العربية. في حين يواجه تشكيل الشعر العربي بوجه خاص تحديات إضافية، مما يؤدي إلى معدل DER أقل قدره 3.28٪ ومعدل WER قدره 10.74٪.