A Dual-Function Large Language Model for Correcting Arabic Spelling Mistakes and Adding Diacritics: Bridging Jordanian Dialect and Formal Arabic

By

Rabie Amin Otoum

Supervisor

Prof. Gheith Abandah

Co-Supervisor

Dr. Mohammed Abdel-Majeed

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

School of Graduate Studies

The University of Jordan

May, 2025

COMMITTEE DECISION

This Thesis/Dissertation (A Dual-Function Large Language Model for Correcting Arabic Spelling Mistakes and Adding Diacritics: Bridging Jordanian Dialect and Formal Arabic) was Successfully Defended and Approved on _____

Examination Committee	**Signature**
Dr. Gheith A. Abandah, (Supervisor) Professor of Computer Engineering	----------
Dr. Mohammed R. Abdel-Majeed (Co-supervisor) Associate Professor of Computer Engineering	----------
Dr. Iyad F. Jafar (Member) Professor of Computer Engineering	----------
Dr. Esam A. AlQaralleh, (Member) Professor of Computer Engineering Princess Sumaya University for Technology (PSUT)	----------

DEDICATION

I dedicate this thesis to my beloved parents, my mother Wafa, I ask Allah Almighty to have mercy on her, and my father Amin. They have raised me and patiently endured it; I ask Allah the Almighty to reward them on my behalf with the best reward He has ever given to parents for their child.

To my sisters, Dana, Maha, and Farah, who always encouraged me and believed in me which have always been the wind beneath my wings.

To my brother, Mohammed, the man who I always account on. The man who understands me deeply, the man who understands me on a whole other level.

To my grandmothers, you both just radiate pure tenderness. Your gentle way and the unconditional love I feel from you has been like a guiding star in my life. Your prayers to Allah for my success mean a lot to me.

To the rest of my family, my constant source of love, and friends, who are the chosen of my extended family. Thanks a million, to each one and everyone.

AKNOWLEDGEMENT

I am deeply grateful to my supervisor, Professor Gheith Abandah. His help was truly invaluable, and his insights pointed me in the right direction throughout this research. His deep knowledge and experience were such a great resource to learn from. I especially appreciated his straightforward honesty and the constant support he gave me; it really made a difference. The time he generously spent discussing our findings and offering his truly golden advice was so incredibly helpful. Professor Abandah has been an exceptional mentor, and I am grateful for his guidance.

I also want to extend my sincere thanks to my co-supervisor, Dr. Mohammed Abdel-Majeed. He always kept in touch and followed up on my progress. He was also a magnificent supporter when it came to technical issues, and he directly assisted me with official University tasks. Thanks so much.

Acknowledgement is also due to the Computer and Mechatronics departments at School of Engineering, and School of Graduate Studies in the University of Jordan. Their support throughout my master's degree program journey is appreciated.

Contents

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

DEDICATION ..... **iii**

AKNOWLEDGEMENT ..... **iv**

LIST OF TABLES ..... **viii**

LIST OF FIGURES ..... **ix**

LIST OF ABBREVIATIONS ..... **xi**

ABSTRACT ..... **xiii**

Chapter One **1**

Introduction **1**

1.1 Spelling Errors in the Arabic Language 3

1.2 Diacritics in the Arabic Language 4

1.3 Formal Arabic and Arabic Dialects 6

1.4 Multi-Task Learning in Natural Language Processing 7

1.5 Thesis Problem 8

1.6 Thesis Importance 9

1.7 Thesis Objectives 10

1.8 Thesis Questions and Assumptions 10

1.9 Thesis Contribution 11

1.10 Thesis Methodology 12

1.11 Thesis Outline 13

Chapter Two **14**

Literature Review **14**

2.1 Arabic Dialects-to-MSA Conversion 15

2.2 Arabic Spelling Errors Correction 18

2.3 Automatic Diacritization of Arabic Texts 20

2.4 Multi-Task Learning: Arabic Spelling and Diacritization 23

Chapter Three **27**

Modern Transformer Models **27**

3.1 Transformers 27

3.1.1 Transformer Architecture 28

3.1.2 Attention Mechanisms 30

3.1.3 Feed Forward Networks 31

3.1.4 Positional Encoding 32
3.2 Transfer Learning 32
3.3 Text-to-Text Transfer Transformer 34
3.3.1 T5 Model Architecture 34
3.3.2 T5 Model Approach 35
3.4 A Massively Multilingual Pre-trained Text-to-Text Transformer 36
3.4.1 mT5 Model Architecture 37
3.4.2 mT5 Model Approach 37
3.5 Byte Level Text-to-Text Transfer Transformer Model (ByT5) 38
3.5.1 ByT5 Model Architecture 39
3.5.2 ByT5 Model Approach 40
3.5.3 ByT5 for Arabic Language Tasks 41

Chapter Four 43

Datasets Preparations and Experimental Setup 43

4.1 Data Corpus Development and Preparation 43
4.1.1 Tashkeela Dataset Preparation 44
4.1.1.1 Tashkeela Dataset Overview 44
4.1.1.2 Synthetic Error Injection 46
4.1.2 JODA Dataset Preparation 49
4.1.2.1 JODA Dataset Overview 49
4.1.2.2 JODA Dataset Diacritization 50
4.2 Preparing Datasets for Multi-Task Learning 53
4.3 Experimental Setup 54
4.4.1 Experimental Environment and Computational Resources 55
4.4.2 ByT5 Multi-Task Learning Evaluation 56
4.4.3 ByT5 Multi-Task Learning and Code Validation 58

Chapter Five 60

Experiments and Results 60

5.1 Exploring Model's Architecture Parameters 61
5.2 Exploring Best Hyperparameters 67
5.2.1 Selection of Maximum Epochs Number 68
5.2.2 Exploring Effect of Learning Rate 69
5.2.3 Exploring Effect of Maximum Sequence Length 72
5.2.4 Exploring Effect of Batch Size 76
5.3 Dataset Impact on Model Performance 79
5.3.1 Initial Experiments with JODA Dataset 80
5.3.2 Combining JODA and Clean-50 Datasets 81
5.3.3 Scaling Up: Adding Clean-400 Dataset 83
5.3.4 Investigating REALMS Dataset Integration 86
5.3.5 Final Dataset Selection and Results 88
5.4 Final Model and Evaluation 89
5.4.1 Model Configuration and Dataset 89
5.4.2 Model Performance and Evaluation 91
5.4.3 Manual Linguistic Evaluation 93

Chapter Six 98

Conclusion and Future Work 98

6.1 Conclusion 98

6.2 Future Work 99

References 100

ملخص 104

LIST OF TABLES

Table 1. Primary Arabic Diacritics. 5
Table 2. Comparison of Encoder and Decoder Layer Depth in ByT5 and T5 Architectures. 40
Table 3. Maximum Input/Output Sequence Length in the Prepared Clean-50 and Clean-400 Datasets. 46
Table 4. Replacement Groups for Soft Orthographic Error Injection (Abandah, et al., 2022). 47
Table 5. Examples for Error Injection Patterns of Random Orthographic Distortion. ... 48
Table 6. JODA Dataset Number of Samples per Split Type. 49
Table 7. Samples of Diacritized MSA Sentences from the JODA Dataset. 53
Table 8. System Specifications for MTL Training Machine. 56
Table 9. Examples of Inference Results from N2W / W2N Trial Model. 59
Table 10. Summary Statistics of the Number of Epochs at Which Best Performance was Achieved in Training Experiments. 69
Table 11. Evaluation Metrics of fine-tuning ByT5 model for Varying Learning Rates. 71
Table 12. Evaluation Metrics (CER, BLEU, DER, WER, and Total Score) for Different Input/Output Sequence Length Configurations. Longer sequence lengths correlate with better performance across all metrics. 75
Table 13. Performance Results of Initial Experiments that Utilized JODA Dataset Only. 80
Table 14. Hyperparameters and Architectural Parameters Adopted in our Final Model. 90
Table 15. Final Model: Performance Metrics on JODA and JODA+Clean-400 Test Sets. 93
Table 16. Final Model: Manual Evaluation Results, Including and Excluding Machine-Generated Errors. 95
Table 17. Illustrative Examples from JODA Test Set with Prediction Notes. 97

LIST OF FIGURES

Figure 1. Transformers Structure (Vaswani, et al, 2017). 29
Figure 2. JODA Dataset Statistics per Sample Type. 50
Figure 3. Diacritization Training Progress Curve for Validation Accuracy and Validation Loss. 52
Figure 4. diagram for the ByT5 Model Text-to-Text Framework. 54
Figure 5. Comparison of Total Evaluation Scores between 25% and 50% Trainable Parameter Ratios. 63
Figure 6. Comparison of Total Evaluation Scores between 25% and 75% Trainable Parameter Ratios, Using a Different Set of Experiments. 64
Figure 7. Comparison of Total Evaluation Scores between 25% and 100% Trainable Parameter Ratios. 65
Figure 8. Impact of the Precision parameter on Total Evaluation Score. The chart compares 32-true and bf16-mixed precision 66
Figure 9. Training and Validation Loss Curves Over 10 Epochs (Early Experiments with JODA Dataset). 68
Figure 10. Loss Curve of Fine-Tuning ByT5 for Varying Learning Rates. 70
Figure 11. Accuracy Curve of Fine-Tuning ByT5 for Varying Learning Rates. 71
Figure 12. Validation Loss Curves for Different Input/Output Sequence Length Configurations Over Five Epochs. 73
Figure 13. Accuracy Curves for Different Input/Output Sequence Length Configurations Over Five Epochs. 74
Figure 14. Evaluation Scores for different Sequence Length Configurations. 75
Figure 15. Training Time for Different Sequence Length Configurations. 76
Figure 16. Validation Loss Curves for Batch Sizes 128 and 256 over Three Epochs. 77
Figure 17. Validation Accuracy Curves for Batch Sizes 128 and 256 over Three Epochs. 78
Figure 18. Total Evaluation Scores for Batch Sizes 128 and 256. 79
Figure 19. Performance Enhancements After Adding Clean-50 to JODA Dataset (Evaluated using JODA Test Set). 82
Figure 20. Evaluation Metrics Comparison Between JODA+Clean-50 and JODA+Clean-400 Datasets (Evaluated using JODA Test Set), Showing Percentage Improvements/Reductions in CER, BLEU, DER, and WER. 84

Figure 21. Validation Loss Curves for JODA+Clean-50 and JODA+Clean-400 Datasets over Three Epochs. 85
Figure 22. Validation Accuracy Curves for JODA+Clean-50 and JODA+Clean-400 Datasets over Three Epochs. 86
Figure 23. Evaluation Metrics Comparison Between Repeated (JODA+REALMS)+Clean-400 and JODA+Clean-400 Datasets (Evaluated using JODA Test Set), Showing Percentage Enhancements/Degradations in CER, BLEU, DER, and WER. 87
Figure 24. Total Score Comparison Across Different Dataset Configurations. 89
Figure 25. Training and Validation Loss Curves of the Final Model Across the Training Epochs. 91
Figure 26. Training and Validation Accuracy Curves of the Final Model Across the Training Epochs. 92
Figure 27. Impact of Evaluation Methodology: CER and DER Results from Automatic and Manual Evaluation. 96

LIST OF ABBREVIATIONS

AD	Arabic Dialects
BERT	Bidirectional Encoder Representations from Transformers
bf16	Brain Floating Point 16-bit
BiLSTM	Bidirectional Long Short-Term Memory
BLEU	Bilingual Evaluation Understudy
ByT5	Byte-to-Byte T5
C4	Colossal Clean Crawled Corpus
CA	Classical Arabic
CER	Character Error Rate
CPU	Central Processing Unit
DER	Diacritic Error Rate
FFN	Feed-Forward Network
fp16	Floating Point 16-bit
FP32	Floating Point 32-bit
GPT	Generative Pre-training Transformer
GPU	Graphics Processing Unit
JODA	Jordanian Arabic to Modern Standard Arabic
LLM	Large Language Model
LSTM	Long Short-Term Memory
mC4	Multilingual Colossal Clean Crawled Corpus
MSA	Modern Standard Arabic
mT5	Multilingual T5
MTL	Multi-Task Learning

N2W	Number to Word
NLP	Natural Language Processing
NMT	Neural Machine Translation
RAM	Random Access Memory
REALMS	Real Arabic Language Mistakes in Spelling
RNNs	Recurrent Neural Networks
T5	Text-to-Text Transfer Transformer
W2N	Word to Number
WER	Word Error Rate

A Dual-Function Large Language Model for Correcting Arabic Spelling Mistakes and Adding Diacritics: Bridging Jordanian Dialect and Formal Arabic

By
Rabie Amin Otoum

Supervisor
Dr. Gheith Abandah, Prof.

Co-Supervisor
Dr. Mohammed Abdel-Majeed

ABSTRACT

The Arabic language, with its intricate grammar and diverse dialects, presents unique challenges for natural language processing (NLP), particularly when Modern Standard Arabic (MSA) and regional dialects, such as Jordanian, coexist. Traditional rule-based or statistical methods often struggle to accurately handle such texts.

This research proposes a unified model that performs two main tasks: converting Jordanian dialect or erroneous MSA text into corrected MSA and adding diacritics to the corrected text. The model provides flexibility to perform these tasks either individually or simultaneously, significantly enhancing readability.

We assessed the effectiveness of the token-free ByT5 model by fine-tuning it primarily on the JODA dataset and supported subsets of the Tashkeela dataset: Clean-50 and Clean-400. We systematically explored various hyperparameters and architectural settings to optimize concurrent task performance using a task-specific prefix inspired by the original T5 model. Standard evaluation metrics, including diacritic error rate (DER), character error rate (CER), bilingual evaluation understudy (BLEU), and word error rate

(WER), were applied to comprehensively evaluate the model and aggregated in a derived score (Evaluation Score).

Automatic evaluation yielded notable performance, with the best model achieving an Evaluation Score of 78.06 on the JODA test set and 92.45 on a combined test set (JODA and Clean-400). Recognizing the limitations of automated metrics in capturing linguistic nuances, we also conducted a manual evaluation of 200 randomly selected samples, which resulted in an Evaluation Score of 96.71, highlighting the model's accurate performance. This research underscores the potential of a unified ByT5 approach, paving the way for more sophisticated NLP approaches adept at managing the complexities of both formal Arabic and Arabic dialects.

Chapter One

Introduction

People communicate using intricate systems to share their thoughts, ideas, and emotions, through what we term natural languages. These languages are defined by their structure, sounds, vocabularies, semantics, and pragmatics, forming the basis of human interaction, and understanding.

Within these systems, Arabic language stands out for its complexity and beauty, offering a unique set of challenges and opportunities for natural languages processing (NLP) research. With its rich history, Arabic spans across the Arab nations and further afield, acting as an official or co-official tongue within the Arab League's 22 member countries. It also holds a national language status in Mali, Niger, and Senegal, and is found in specific linguistic communities within Nigeria, Cyprus, Türkiye, Uzbekistan, Iran, and Afghanistan. Moreover, Arabic is utilized as a sacred language across Muslim-majority and Muslim-minority regions throughout Africa and Asia (Bouhdima, 2022). Arabs living in western countries also may maintain their mother language. Al Alili and Hassan's research (2017) in the US revealed that parents, regardless of their Arabic-speaking status, hold positive views on their children acquiring Arabic.

Classical Arabic (CA) is the form of the Arabic language used in the Quran, Islamic literature, and other early Arabic literary works texts. Modern standard Arabic (MSA) is derived from CA and is used today in formal writing and speech across the Arab world. Unlike CA, MSA reflects contemporary usage and is taught in schools. Arabic regional dialects, conversely, are spoken forms of Arabic varying significantly across regions, often with considerable differences in pronunciation, grammar, and vocabulary from both CA and MSA. Zaidan and Callison-Burch (2014) categorized main Arabic dialects into

five; Egyptian, Levantine, Gulf, Iraqi and Maghrebi. However, each county in each region stands out by many terms. Even in the same country, there could be some differences based on the geographical area.

Social media platforms, now ubiquitous and accessible from nearly everywhere, host a vast array of content in regional dialects that lack the formal rules characterizing CA or MSA. This absence of standardized grammar and vocabulary in regional dialects presents unique challenges for text analysis. Processing MSA is equally vital as it remains widely used in social media and is the official language in Arabic-speaking countries. MSA's structured grammar and vocabulary facilitate clear communication in formal settings, educational materials, news outlets, and legal documents. Consequently, the application of deep learning, particularly large language models (LLMs), has become essential for effectively processing text in these dialects or in MSA for a broad spectrum of applications.

The complex nature of Arabic, with its grammar and dialects, poses hurdles for text processing, particularly on social media mixing MSA and dialects. Traditional methods lag pre-trained LLMs in tasks like diacritization and spelling correction, as LLMs learn nuances from vast data without explicit rules. Their adaptability makes LLMs crucial for effective Arabic language processing, offering advantages overrule-based approaches that require manual linguistic encoding.

Arabic NLP tries to solve different kinds of problems such as the lack of diacritics, and spelling errors, that are committed by both native and foreign Arabic speakers. Furthermore, regional dialects introduce additional challenges, as they feature unique variations in vocabulary, pronunciation, and spelling that deviate from MSA. Different deep learning architectures showed strong potential in solving spelling error correction,

including conversion of dialectal texts to MSA, and diacritics addition separately or simultaneously. Moreover, leveraging pre-trained LLMs can result in superior performance compared to other deep learning techniques.

This thesis seeks to capitalize on transfer learning and the ByT5 model to tackle the simultaneous tasks of spelling error correction, encompassing texts in local dialect, and diacritic addition in Arabic text.

1.1 Spelling Errors in the Arabic Language

The Arabic language, with its rich history and complex writing system, presents unique challenges for spelling accuracy. According to Boumaraf, *et al*. (2022), Arabic orthography contains ambiguities, primarily related to long vowels, certain phoneme-grapheme conversions, lexical particularities, and letter connectivity. The cursive nature of Arabic script and the frequent omission of short vowels further exacerbate these ambiguities, potentially causing significant confusion in pronunciation, writing, and interpretation. These complexities lead to considerable difficulties and often result in spelling inaccuracies both for learners and proficient users of Arabic.

Morphological confusion between some Arabic letters, particularly for similar letters like ث (*tha*) and ت (*ta*), can result in incorrect word formation. Also, phonetically similarity, such as (ض) and (ظ), is a major cause in common spelling errors that are made by non-linguists. Additionally, the complex nature of Arabic characters, such as the various forms of Hamza (ء, ؤ, ئ, أ) adds to the difficulty of accurate spelling, especially in digital text input. Typographical errors caused by input methods and keyboard layouts are also a significant source of inaccuracies. Arabic keyboards are often prone to errors due to the proximity of certain letters, such as ب (*ba*) and ي (*ya*), or the difficulty in accessing less commonly used characters like Hamza forms (ء, ؤ, ئ, أ). Such errors can

propagate in digital applications, affecting tasks like search engine queries, text-to-speech systems, and machine translation (Hasan and Abandah, 2023).

Speaking about computational complexity, Arabic text encoding follows standards like UTF-8, a widely adopted encoding system that supports the full range of Arabic characters and symbols. Each Arabic letter is typically encoded in two bytes (16 bits) in UTF-8, compared to the single byte used for English characters in ASCII. This difference affects text storage and processing, as Arabic text requires more memory and computational resources. For example, the various forms of the Hamza (أ, ئ, ؤ, ء) are treated as distinct characters in UTF-8, which increases the encoding complexity.

Consequently, these challenges underscore the need for specialized models and approaches tailored to handle Arabic-specific spelling issues effectively.

1.2 Diacritics in the Arabic Language

Diacritics in Arabic language are small symbols placed above or below letters in to indicate short vowels and specific phonetic and grammatical characteristics essential for accurate pronunciation and comprehension. The primary diacritics of Arabic are shown in Table 1.

In written Arabic, especially MSA, diacritics significantly influence the meaning and grammatical function of words. Arabic is typically written without short vowels, relying on readers' linguistic knowledge to correctly interpret words from context. However, this omission often introduces ambiguity (Abandah, et al., 2022), making the text challenging for learners, non-native speakers, and automated text-processing systems.

Table 1. Primary Arabic Diacritics.

Diacritic Name	Arabic Symbol
Fatha	ـَ
Kasra	ـِ
Damma	ـُ
Sukoon	ـْ
Shadda	ـّ
Tanween Fatha	ـً
Tanween Kasra	ـٍ
Tanween Damma	ـٌ

Diacritics are particularly crucial in formal settings such as religious texts, classical literature, poetry, language learning resources, and computational applications like Text-to-Speech (TTS) systems, where precise pronunciation and interpretation are critical. For instance, two words spelled identically without diacritics can hold entirely different meanings once diacritics are added (Alqudah, *et al*., 2017).

Due to these challenges, there is considerable academic and practical interest in developing computational models capable of automatically restoring or adding diacritics to Arabic texts, improving the accuracy of NLP systems, such machine translation and speech recognition tools. Recent advancements in deep learning and transformer-based architecture have shown promising capabilities in achieving accurate and context-sensitive diacritization, underscoring the importance of continued research in this domain.

1.3 Formal Arabic and Arabic Dialects

Modern standard Arabic serves as the standardized linguistic form employed across the Arab world in formal contexts such as literature, media, and official communications. While MSA acts as a unifying medium, the vernacular spoken in everyday life exhibits significant regional diversity, leading to the emergence of numerous Arabic dialects. These dialects are broadly categorized into regional varieties, including Egyptian, Levantine, Gulf, and Maghrebi Arabic, each shaped by distinct historical, cultural, and social influences. For instance, Egyptian Arabic is the predominant spoken form in Egypt, while Levantine Arabic is prevalent in countries such as Jordan, Syria, Lebanon, and Palestine. Gulf Arabic is commonly used in nations like Saudi Arabia, the United Arab Emirates, and Kuwait, and Maghrebi Arabic is spoken in North African countries like Morocco, Algeria, and Tunisia.

Within Jordan, the local dialect, Jordanian Arabic, demonstrates considerable internal variation influenced by the country's diverse geography and demographics. Key dialectal divisions in Jordan include Urban, Rural, Bedouin, and Ghorani varieties (Al-Ali & Arafa, 2010). The Urban dialect is commonly heard in cities like Amman, while the Rural dialect is prevalent in areas surrounding Irbid and other northern regions. The Bedouin dialect is associated with nomadic communities, and the Ghorani dialect is found in the Jordan Valley region. These regional dialects exhibit variations in phonology and vocabulary, reflecting the rich linguistic landscape of Jordan. For example, a comparative study of the dialects of Al-Karak and Amman highlights specific linguistic features that contribute to regional identities and social perceptions within Jordan (Ja'afreh & Al-Saudi, 2024)

Contemporary communication, particularly on social media platforms, reveals a notable trend towards the use of native dialects among Arabic speakers. This preference allows for more authentic and relatable online interactions, as individuals communicate in their daily language. However, the lack of standardized orthography for these dialects often results in diverse spelling conventions and the incorporation of loanwords, notably from English and Turkish, into everyday digital communication. Research investigating the impact of social media on Arabic language use has observed a shift, with increased reliance on colloquial Arabic and foreign vocabulary in online interactions (Al-Jarf, 2019). This phenomenon underscores the dynamic and evolving nature of Arabic language usage in the contemporary digital sphere.

The increasing prevalence of dialectal Arabic in digital communication presents significant issues and challenges for NLP applications. Unlike the relatively standardized MSA, dialectal texts exhibit a high degree of variability in phonological representation, lexical choice, and grammatical structures. The absence of consistent orthographic conventions leads to a wide range of spellings for the same words. The incorporation of code-switching, where speakers blend dialectal Arabic with MSA or foreign languages, adds another layer of complexity. These linguistic characteristics of dialectal texts make tasks such as machine translation to MSA and then diacritics addition, considerably more challenging compared to processing standard Arabic.

1.4 Multi-Task Learning in Natural Language Processing

Multi-task learning (MTL) has gained significant traction as a powerful framework in the field of NLP, offering a way for models to learn from and leverage shared representations across multiple related tasks. Unlike traditional single-task learning, which focuses on optimizing a model for one specific task, MTL enables models to tackle

a variety of objectives simultaneously. This approach allows the model to generalize better by transferring knowledge learned from one task to others, which leads to improved performance on tasks with limited data. For instance, in Arabic NLP, MTL-AraBERT (Fadel, *et al.*, 2024) demonstrated this advantage by jointly training a pre-trained AraBERT model on aspect term extraction and aspect category detection tasks. By sharing linguistic representations across tasks, the model achieved 80.32% F1-score for aspect term extraction and 68.21% F1-score for aspect category detection on the SemEval-2016 dataset, outperforming single-task baselines.

By addressing multiple challenges at once, MTL helps to mitigate issues like overfitting, which often arises when training on a single, task-specific dataset which also was illustrated by the MTL-AraBERT framework that reduced error propagation through simultaneous optimization of interdependent tasks. Furthermore, MTL enhances data efficiency by making better use of available training data, reducing the need for large, task-specific datasets. This makes MTL especially valuable in NLP applications where data scarcity can be a significant bottleneck. Overall, MTL represents a shift towards more flexible and robust learning paradigms that are better suited to real-world NLP challenges, where tasks are often interrelated, and knowledge sharing is critical for optimal performance.

1.5 Thesis Problem

The Arabic language, known for its complexity, rich grammar, and diverse dialects, presents significant challenges for processing text, especially from social media platforms that feature both MSA and regional dialects like the Jordanian dialect. Compared to contemporary techniques, such as leveraging pre-trained LLMs, traditional rule-based or statistical methods often present reduced performance in accurately processing Arabic

text for tasks such as diacritization and spelling error correction. These complexities are more adeptly managed by the advanced capabilities of LLMs, which can better understand and interpret the nuances of Arabic text.

Employing separate models for spelling error correction and diacritization in Arabic text can lead to significant compute and memory demands. This research proposes a compact, unified model utilizing LLMs to streamline these tasks while also converting text from the Jordanian dialect to MSA. This conversion is not just a goal but a strategic approach to ensure effective spelling correction and optional diacritization, addressing the computational efficiency challenges and facilitating a seamless processing of Arabic text into its standardized form.

1.6 Thesis Importance

The significance of this research stems from the pervasive issue of spelling mistakes in Arabic text and the critical role diacritics play in altering the intended meanings. This dual approach addresses critical challenges in Arabic NLP, enhancing readability and comprehension across digital platforms. By improving the accuracy of text-based applications, from chatbots to text-to-speech systems, this model aims to make digital services more accessible and efficient for Arabic speakers, significantly contributing to technological inclusivity and facilitating smoother communication in various sectors.

This research addresses the labor-intensive nature of manual diacritization, spelling error correction, and dialect-to-MSA conversion, all tasks requiring a deep understanding of Arabic grammar. To tackle these challenges, we propose a novel method leveraging pre-trained byte-to-byte language models to enhance the precision and effectiveness of spelling error correction, including the conversion of Jordanian dialectal texts to MSA and automatic diacritics restoration. The significance of this comprehensive approach is

amplified by the fine-tuned model's capacity to process both MSA and the Jordanian dialect, enabling it to navigate the nuances of both formal and colloquial Arabic, which is crucial for a wide array of applications and for developing a solution capable of addressing these tasks either individually or concurrently in Arabic.

1.7 Thesis Objectives

The main objectives of this research are as follows.

- Provide an overview of various approaches that use deep learning architectures along with transfer learning previously utilized in Arabic text processing, especially spelling error correction, dialect-to-MSA conversion, and diacritics addition.
- Explore the effectiveness of unified, token-free models such as ByT5 in simultaneously performing spelling corrections and diacritization for MSA while converting text from the Jordanian dialect to MSA, offering the flexibility to append diacritics in the output as per user preference.
- Leverage transfer learning in utilizing LLMs, such as ByT5, by finetuning it, using different datasets.
- Explore efficient hyper-parameter settings to empower the model to concurrently correct spelling errors and apply diacritization, whether dealing with MSA or the Jordanian dialect.

1.8 Thesis Questions and Assumptions

The main questions are of this research are as follows.

- Given the strategy of converting Jordanian dialect to MSA as a methodological step, how does training the model on datasets containing Jordanian dialect sentences and their MSA translations, alongside MSA and CA texts, influence its accuracy in diacritization and spelling correction tasks?

- What would be the consequences of excluding the diacritization task on the model's performance metrics?

- How does the model's performance in spelling error correction and diacritization tasks compare when processing inputs specifically from the Jordanian dialect versus MSA, especially considering the conversion process to MSA is inherent for Jordanian dialect inputs?

The main research assumptions are:

- The Jordanian Arabic to Modern Standard Arabic (JODA) dataset, which was produced by (Abandah, *et al.*, 2025) will be available and used as a main source of Jordanian dialectal data.

- The new dataset, JODA, already includes some spelling mistakes, and there is a requirement to complement it with other datasets with artificial errors.

- The available text in this new dataset lacks diacritics.

- Most of the existing Arabic datasets for NLP are undiacritized or partially diacritized.

- Adding diacritics to the text is optional, allowing for flexibility in the output to either include or exclude diacritics according to the practitioner’s preference.

1.9 Thesis Contribution

This research contributes to Arabic NLP by addressing the dual challenges of spelling error correction and diacritization, focusing on MSA and the Jordanian dialect. It explores the use of ByT5, a token-free LLM, for performing both tasks simultaneously, eliminating the need for tokenization and accommodating Arabic’s complex morphology.

By incorporating dialect-to-MSA conversion, the study enhances the model's adaptability to real-world linguistic variations. Using suitable training and tuning techniques, this thesis demonstrates that the selected LLM provides accurate solutions for the dual problems.

1.10 Thesis Methodology

The research methodology of this thesis is summarized as follows.

- Carry out literature review about text spelling correction, dialect-to-MSA conversion and diacritization separately or simultaneously.
- Prepare the dataset for both tasks, including spelling errors injection into the training dataset.
- Finetune the LLM, ByT5, and use it to diacritize the JODA datasets as part of the data preparation procedures.
- Explore the ability of the ByT5 model to handle two tasks simultaneously.
- ByT5 model will be used in transfer learning to handle the downstream tasks of Arabic text diacritization and spelling error correction for both MSA and Jordanian dialect.
- Explore the effect of larger dataset. JODA and Tashkeela will be used.
- Finetuning the model's parameters.
- Custom loss functions could be leveraged to enhance the learning process if needed, particularly to counteract the issue of partially diacritized Arabic text available for training.
- DER, CER, WER, and BLEU score metrics will be used to evaluate performance.

- Identify potential future work to enhance generalizability, particularly for under-represented spelling errors and dialect variations, potentially through datasets improvements or exploring different LLMs based on findings.

1.11 Thesis Outline

The remainder of this thesis is structured as follows:

- Chapter Two: literature review about dialect-to-MSA conversion, text spelling errors correction, and diacritic addition.
- Chapter Three: explains key deep learning concepts and the proposed model, particularly the ByT5 which is one of the T5 model variants.
- Chapter Four: details of the data preparation process and the experimental setup.
- Chapter Five: reports on experiments conducted to identify the optimal model for the thesis objective.
- Chapter Six: provides a summary of the conclusions drawn and outlines potential future research.

Chapter Two

Literature Review

Arabic text diacritization and spelling error correction, including dialects-to-formal Arabic, MSA, conversion have undergone a dramatic shift, transitioning from intricate rule-based systems to powerful deep learning techniques. This evolution reflects the need for models that can handle the richness and complexity of Arabic morphology.

Early approaches in Arabic NLP, such as rule-based systems, utilized handcrafted linguistic rules to understand Arabic structure but lacked the adaptability needed for diverse texts. Statistical methods then emerged, offering data-driven improvements by analyzing context, though they still depended on manual feature engineering and struggled with inherent linguistic ambiguity. For instance, Alnefaie and Azmi (2017) used morphological analysis and statistics for selective diacritization to reduce ambiguity in Arabic Treebank data. Hybrid models attempted to combine the strengths of both rule-based and statistical techniques, leading to better predictions, as demonstrated by Darwish, *et al*. (2017) who employed a Viterbi decoder and SVM ranking for diacritization, achieving low error rates on a multi-genre dataset. However, the complex relationship between Arabic spelling and pronunciation continued to present a significant challenge, indicating the need for more adaptable modeling techniques.

The arrival of deep learning techniques, including recurrent neural networks (RNNs), Long Short-Term Memory (LSTM) networks, attention mechanisms, encoder-decoder, and transformers architectures, revolutionized the field. Inspired by advancements in machine translation and speech synthesis, these models possess the ability to learn complex data representations and capture long-range dependencies within text. This shift resulted in a significant leap in performance.

This chapter provides an overview of various approaches that adopted modern deep learning techniques, for addressing conversion of dialects to formal language, common MSA spelling mistakes correction, and diacritics addition.

2.1 Arabic Dialects-to-MSA Conversion

A study accomplished by Farhan, *et al*. (2020) presented pioneering work in unsupervised dialectal neural machine translation (NMT), particularly targeting the translation of Arabic dialects into MSA. The authors proposed two systems: Dialectal to Standard Language Translation (D2SLT) and a Google NMT-based (GNMT) model, utilizing an attentional sequence-to-sequence architecture. The study employs a unique dataset containing 309,000 sentences covering MSA, Jordanian, Saudi, and Egyptian dialects. Specifically, 307,000 sentences were used for training, and 2,000 sentences were reserved for testing.

The research introduces an innovative methodology where dialectal and standard Arabic share embedding spaces, reducing the training complexity and enabling effective translation even without dialect-specific parallel training data. Experiments conducted in supervised and unsupervised settings demonstrated that their proposed models effectively translated Jordanian dialect to MSA. Notably, the D2SLT system achieved a BLEU score of 48.25% in supervised settings and an impressive score of 32.14% in unsupervised scenarios (training on Saudi-MSA), underlining the efficacy of their unsupervised approach. This research significantly contributes to dialectal translation by addressing challenges of resource scarcity and promoting the generalization capability of NMT systems.

The paper by Al-Ibrahim and Duwairi (2020) focused on translating Jordanian dialect into MSA using NMT, specifically employing the RNN encoder-decoder

architecture. The authors created two manually annotated datasets: a word-level dataset (W2W) consisting of 24,200-word pairs and a sentence-level dataset (S2S) containing 500 sentences. Both datasets were collected from diverse local resources and underwent comprehensive preprocessing, including normalization, punctuation removal, diacritics elimination, and character repetitions handling.

The translation experiments were divided into two categories: word-level and sentence-level translations. At the word level, the RNN model achieved an accuracy of 91.3% with a loss of 0.8, demonstrating robust performance. However, at the sentence level, the model's accuracy decreased to 63.2% with a loss of 3.33, reflecting greater complexity inherent in sentence structures and contextual dependencies. Despite the challenges posed by limited training data and resource scarcity typical of low-resource languages like Arabic, these findings underscore the effectiveness of the RNN encoder-decoder approach in dialect-to-standard language translation and highlights the feasibility of applying deep learning techniques to Jordanian dialect translation.

The paper by Slim and Melouah (2024) tackles the challenges associated with NMT for low-resource Arabic dialects through a novel incremental transfer learning strategy specifically tailored to leverage shared linguistic characteristics among various dialects. The methodology involves sequentially transferring learned linguistic features from a general Arabic dialect model (grandparent model) to increasingly specialized models targeting specific Maghrebi dialects, ultimately fine-tuning separate models for Algerian, Tunisian, and Moroccan dialects (child models). This incremental transfer learning process is designed to progressively refine the linguistic knowledge acquired at each step, thereby enhancing model generalization and translation quality.

The researchers employed the multi-dialectal Arabic corpus (MADAR) and Penn Arabic dialect corpus (PADIC) datasets, systematically partitioned into subsets representing general Arabic dialects (48,000 sentences), Maghrebi dialects (23,650 sentences), and specialized dialect subsets for Algerian, Tunisian, and Moroccan (ranging from 7,411 to 11,821 sentences each). The experimental setup included Transformer and Attentional Seq2Seq models, with translation performance rigorously assessed via BLEU scores.

The authors found that incremental transfer learning effectively addresses data scarcity by gradually specializing model knowledge, outperforming standard transfer learning. Their Transformer model showed significant BLEU score gains for Algerian, Tunisian, and Moroccan dialects (5.22, 2.08, and 1.47 increasing points), with Attentional Seq2Seq models also benefiting. The study concludes that sequential knowledge transfer enhances translation quality by capturing dialectal nuances, proving the practicality of incremental learning in low-resource translation.

Alimi et al. (2024) introduced a detailed method for translating various Arabic dialects into MSA using fine-tuned Transformer models, focusing on T5X and the AraT5v2-base-1024 model, which was pre-trained on diverse Arabic data. The researchers tackled the complexities of Arabic dialects by creating translation systems for Levantine (Palestinian, Jordanian, Syrian) and Maghrebi (Tunisian, Algerian, Moroccan, Libyan) dialects, utilizing substantial parallel datasets. For Levantine, they used 31,114 entries, with Jordanian contributing 6,200 training and 2,000 testing examples.

After meticulously fine-tuning hyperparameters, the AraT5v2-base-1024 model achieved a significant BLEU score of 48.38% on Levantine translation, outperforming T5X. The study also explored Zero-Shot Learning on unseen dialects, further validating

the model's generalization ability. The findings emphasize the crucial role of well-prepared parallel data and the power of advanced Transformer architectures in managing Arabic dialectal variations.

2.2 Arabic Spelling Errors Correction

Spelling errors in Arabic arise from orthographic variations, phonetic similarities, and complex spelling rules, such as different forms of *hamza* and the frequent confusion between similar-looking or sounding letters like *teh marbuta* (ة), *heh* (ه), *alef* (ا), and *alef maksura* (ى). Abandah, *et al*. (2022) tackled the issue of soft Arabic spelling at character level using character-level sequence transcription approach using Bidirectional Long Short-Term Memory (BiLSTM) networks, bypassing the heavier encoder-decoder models typically used for variable-length sequences. For training, the authors used two datasets; A subset of the Tashkeela dataset, which consists of high-quality 55,000 CA sequences, and the Arabic Treebank Part 3 (ATB3) v3.2, featuring 26,000 sequences of MSA from news articles. Two main training strategies were investigated. The first, referred to as the transformed input approach, replaces all instances of commonly confused letters with a standardized form across both input and output sequences. The second, more effective strategy involved stochastic error injection, where artificial soft spelling errors were randomly introduced into clean data during training. Various injection rates were tested, with 40% identified as the optimal rate for achieving the best correction performance. The best-performing model was a two-layer BiLSTM with dropout, trained using the Tashkeela subset and the stochastic error injection method. This model achieved CER of 1.28% on the Test200 set, which is a test dataset containing 200 sequences with real soft spelling mistakes at a rate of 6.5 errors per sequence.

Al-Qaraghuli, *et al*. (2021) have investigated Transformer models for correcting soft spelling mistakes in Arabic. Their focus lied on common errors like shape confusion among *hamza*, *alef* at word ends, *alef* insertion/omission after *waw al-jamaea*, and *teh*/*teh marbuta*/*heh* confusion at word ends. The study employed two datasets: Tashkeela, consists of about 75.5 million words, 98.85% of them are formed in CA, and Wiki-40B, which contains 220,885 articles for training, 12,271 articles for testing, and 12,198 articles for validation written in MSA. To train and evaluate their models, the researchers used a stochastic error injection method to generate artificial errors. This approach yields a high correction accuracy of 97.37% on these artificial errors and a CER of 0.86% on Test200 dataset, containing real soft spelling mistakes. These results highlight the effectiveness of transformer models in understanding and rectifying subtle spelling errors within Arabic text.

Hasan & Abandah (2023) employed Transformer-based model to correct a wide range of Arabic spelling errors, including lexical and semantic errors, keyboard-related mistakes, common typing errors, and general random errors. Specifically, the model addresses challenges such as confusion between letters with similar phonetics (e.g., "س" and "ص"), shape-based similarities (e.g., "ة" and "ه"), and proximity-based keyboard errors (e.g., "ب" and "ي"). The errors are categorized and targeted systematically to enhance correction accuracy. The dataset comprises two main sources: Wiki-40B, a large error-free dataset injected with synthetic errors, and REALMS (Real Arabic Language Mistakes in Spelling), a novel dataset collected from social media platforms containing real-world spelling errors. The Wiki-40B dataset facilitates initial training, while REALMS serves to refine and test the model on authentic error patterns. Using the Transformer model with optimized hyperparameters, the system achieves accuracy of 99.12% on the Wiki-40B dataset and CER of 1.14% on the REALMS dataset. The results

highlight the model's robustness in handling diverse error types, demonstrating its potential to improve Arabic spelling correction tasks in real-world applications.

The paper by Al-Qaraghuli and Jaafar (2024) presents a transformer-based approach for correcting Arabic soft spelling mistakes using a customized T5 model. The work focuses on soft spelling issues that frequently arise in Arabic due to the phonological and orthographic complexity of characters such as different forms of *hamza*, *alef*, *teh*, and *waw*. These mistakes can significantly alter the meaning or clarity of a sentence and are common in informal and formal texts alike. The authors employed the T5 (Text-to-Text Transfer Transformer) architecture, a unified encoder-decoder model initially proposed by Raffel, *et al*. (2020) and fine-tuned it for the soft spelling correction task. They developed three configurations of the T5 model with varying model depths and parameters, trained using the Wiki-40B Arabic subset. This dataset was cleaned, preprocessed, and injected with artificial errors using a stochastic error injection method. For evaluation, the authors used both the processed Wiki-40B test subset (2,000 sequences) and the real-error dataset Test200.

The best model, which consisted of a four-layer Transformer trained with 90% error injection and 300-character input, achieved 0.77% CER on Test200 and corrected 97.8% of artificial errors on Wiki-40B with a 95.79% BLEU score.

2.3 Automatic Diacritization of Arabic Texts

Arabic diacritics are small symbols placed above or below letters to indicate short vowels, pronunciation, and grammatical inflections. While essential for disambiguating meaning and improving readability, they are often omitted in everyday writing. This omission creates challenges for learners and NLP applications, prompting significant research efforts to automatically restore diacritics to Arabic texts.

Madhfar & Qamar (2020) have explored three deep learning models to address the challenging task of automatic diacritization of Arabic text, including a baseline BiLSTM, an encoder-decoder model adapted from Tacotron (an end-to-end text-to-speech model known for its strong sequence encoding capabilities), and a CBHG-based encoder-only model (utilizing the Convolutional Bank Highway Network and Gated Recurrent Unit layers for robust feature extraction from the text). The baseline BiLSTM model serves as a benchmark. The encoder-decoder model, inspired by Tacotron, has been adapted for diacritization instead of generating speech frames. And lastly, the CBHG Model (Convolution Bank + Highway Networks + Gated Recurrent Unit) is an encoder-based architecture which was initially proposed as part of Tacotron and has been adapted to handle diacritization without needing a separate decoder. The models were trained and evaluated on the Tashkeela corpus. The study also examined the interplay between CA and MSA, leveraging distinct corpora for each. Preprocessing was carried out over the datasets including cleaning partially diacritized sentences, removing noisy data, and splitting long sentences into manageable chunks. Experiments revealed the CBHG model as the most effective, achieving results with DER of 1.13% and WER of 4.43% on CA test data, while they were 5.65 and 21.28, respectively, on MSA test set. While, the encoder-decoder model offered competitive performance, but was slower and less efficient.

Al-Rfooh, *et al*. (2023) explored fine-tuning pre-trained token-free multilingual models (ByT5) to predict and insert missing diacritics. The approach leverages ByT5's character-level processing, modeling the task as a sequence-to-sequence generation problem where input sequences lack diacritics, and output sequences include them. The Tashkeela corpus serves as the primary dataset, featuring CA and MSA texts. Variants of the dataset, such as Clean-400 and Clean-50, were used for training and evaluation to

mitigate noise and inconsistencies in the original corpus as they are high-quality subsets of the Tashkeela dataset. The study employed curriculum learning, starting with larger datasets (full Tashkeela dataset) for generalization and refining performance using cleaner subsets (Clean-400 dataset). Experiments were conducted with ByT5 models of varying sizes, and optimal hyperparameters are identified through initial runs. The model achieves state-of-the-art results, reducing WER by 40% and achieving DER as low as 0.74% when evaluated on filtered Tashkeela test splits. These results outperform prior approaches, demonstrating the effectiveness of leveraging large-scale pre-trained models and high-quality training data.

Skiredj and Berrada (2024) presented PTCAD (Pre-FineTuned Token Classification for Arabic Diacritization), a two-phase approach to improve Arabic text diacritization (ATD) using pre-trained language models. They framed ATD as token classification and used a multi-stage training strategy. The first phase pre-finetunes a BERT-like model on multiple tasks (CA language modeling, part-of-speech (POS) tagging, segmentation, and diacritization as Masked Language Modeling (MLM). The second phase fine-tunes it specifically for ATD as a token classification task. They used OpenITI for pre-finetuning tasks and the cleaned Tashkeela introduced by Abbad & Xiong (2020) for diacritization pre-finetuning. For fine-tuning, they benchmarked on two Tashkeela versions that were introduced by Abbad & Xiong (2020), and Fadel, et al. (2019).

PTCAD achieved DER of 1.1% and WER of 4.19% on Fadel’s Tashkeela, and DER of 0.87% and WER of 3.53% on Abbad’s version. The study highlighted the benefit of multi-task pre-finetuning for ATD and the potential of adapting BERT-like models for Arabic.

Arabic poetry diacritization is more difficult than other text types as diacritics need to be placed in a way that preserves the meter and rhyme pattern. Also, the lack of datasets that are specialists in poetry makes it a big deal as stated by Abandah, *et al.* (2022) works. The study models diacritization as a sequence-to-sequence task, leveraging the structural patterns in poetry to refine predictions. They used a transfer learning model which integrated a pretrained classification model, to predict the verse meter which helped in better prediction of the diacritics, with a new model for the diacritics prediction (base model). The datasets include the Arabic Poem Comprehensive Dataset (APCD2), containing verses with varying levels of diacritization, and its subsets, DS1 and DS2, filtered for fully diacritized verses. The transfer learning approach utilizes a pretrained poetry classification model, and the training process incorporates three phases: initial training on DS1 (large dataset), fine-tuning on DS1, and final refinement on DS2 (high-diacritics dataset). Experimental results showed significant improvements, with the best model, TL-All, achieving a DER of 3.54%, resulting in 42% improvement over prior results. Key contributions include leveraging meter-based classifications, integrating features across BiLSTM layers, and refining predictions with context-aware post-processing.

2.4 Multi-Task Learning: Arabic Spelling and Diacritization

Multi-Task Learning has emerged as an effective strategy in NLP for enabling a single model to perform multiple related tasks simultaneously. By sharing representations across tasks, MTL often leads to improved generalization and efficiency compared to training separate models. Recent studies have explored unified frameworks that address multiple linguistic tasks within a single architecture. This section reviews key works that demonstrate the potential and challenges of applying multi-task learning for text processing.

Almajdoubah, *et al.* (2021) introduced an approach for simultaneously tackling Arabic text diacritization and correcting soft spelling errors using a single model framework. They focused on leveraging encoder-decoder recurrent neural network models, particularly emphasizing a model that combines an embedding layer, an LSTM layer for both encoder and decoder, augmented with Luong attention (Luong, *et al*., 2015), and concluding with a dense output layer equipped with softmax activation. For their experimentation, they utilized the Tashkeela dataset. To simulate soft spelling mistakes, they applied stochastic error injection on this dataset, creating a robust environment for training and evaluating their models. Additionally, the Wiki-40B dataset, initially error-free, was used for the initial stages of model training. The encoder-decoder model with embedding and attention, referred to as E-D-Emb-Att was adopted to solve the spelling mistakes and diacritization problems simultaneously and achieved accuracy of 99.33% and 99.56% on ATB3 and Tashkeela datasets respectively.

The task of correcting spelling errors and restoring diacritics simultaneously has received less attention in research compared to studies focusing on each problem individually. While significant progress has been made in addressing spelling correction and diacritic restoration as separate tasks, there is a growing interest in exploring their combined challenge. It is useful to expand this perspective by examining related research across various languages, as insights gained from one language can often inform solutions for others.

For instance, the work by Stankevičius, *et al*. (2022) presented an investigation that delves into the capabilities of the ByT5 model for tackling diacritics restoration and typo correction across a diverse range of 13 languages. The study reveals ByT5's remarkable performance, achieving an average accuracy of 98.3% in restoring diacritics, placing it

within striking distance of top-performing models despite requiring less training data and time which underscores ByT5's. The inclusion of a basic statistical unigram model (Dict.+ByT5) demonstrates a marginal improvement in performance under restricted training timeframes. In the combined task of diacritic restoration and typo correction, ByT5 achieves an average accuracy of 94.6%. This highlights the inherent challenge of this dual task. Despite achieving promising results, the Dict.+ByT5 configuration indicates potential for greater improvement with more extensive training. A key limitation was the model's difficulty with uncommon words in the training set. However, ByT5 demonstrated strong generalization, correctly restoring diacritics on over 76% of previously unseen words across the examined languages.

Multi-task frameworks for correcting spelling mistakes, including dialectal text conversion to formal language, and diacritization are relatively uncommon. However, a related study by Baniata, *et al*. (2018), 'A Neural Machine Translation Model for Arabic Dialects That Utilizes Multitask Learning (MTL),' presented a multi-task learning approach for translating Arabic dialects (AD) to MSA, alongside a second task of translating MSA to English. The study aims to address challenges in dialectal Arabic machine translation, especially the scarcity of parallel resources and the lack of standardized orthography across dialects. To tackle this, the authors proposed NMT model built on a recurrent encoder-decoder architecture using Bi-LSTM units. The core of their approach is a multitask learning framework in which separate encoders are used for each source language (AD and MSA), but a single decoder is shared across all translation tasks. This setup allows the model to benefit from task interdependencies, with the MSA-to-English task serving as an auxiliary task to enhance the low-resource AD-to-MSA task. The training data consisted of sentence pairs extracted from the PADIC and MPCA corpora. PADIC (Parallel Arabic Dialect Corpus) provides parallel data in several

dialects aligned with MSA, while MPCA (Multidialectal Parallel Corpus of Arabic) includes parallel texts across multiple Arabic dialects and MSA. These resources covered Levantine dialects (Jordanian, Syrian, and Palestinian) and Maghrebi dialects (Moroccan, Algerian, and Tunisian). The dataset included 13,805 sentence pairs for Levantine dialects and 17,736 for Maghrebi dialects, with 2,000 sentence pairs used for testing each. Preprocessing involved orthographic normalization, tokenization, and removal of non-Arabic characters and diacritics. The multitask model was trained using the Adam optimizer with dropout and gradient clipping. Training alternated between tasks, using mini batches of 256 tokens. The authors reported that the multitask learning model outperformed single-task models significantly in terms of BLEU scores. For Levantine-to-MSA translation, the multitask model achieved a BLEU score of 41% compared to 17% for the single-task model. For Maghrebi-to-MSA, the multitask model scored 30% versus 16%, and for MSA-to-English, 27% versus 10%. The results demonstrate the benefits of multitask learning in low-resource settings. The shared decoder improved generalization and reduced overfitting, and the model was able to capture complex linguistic structures and variations across dialects.

Chapter Three

Modern Transformer Models

The ability of machines to understand and generate human language has been a long-standing goal of artificial intelligence. RNNs revolutionized sequence processing, including texts, by processing information sequentially and maintaining a hidden state to capture past context. However, RNNs faced limitations, particularly in capturing long-range dependencies due to the vanishing and exploding gradient problems. LSTMs and Gated Recurrent Units (GRUs) were introduced to mitigate these issues by incorporating memory cells and gating mechanisms, allowing them to better retain information over longer sequences. Despite these improvements, LSTMs and GRUs still struggled with very long sequences and inherent sequential processing limitations.

Emergence of the encoder-decoder architecture along with attention mechanisms paved the way for an architecture that dispenses with any recurrence. This architecture is called Transformer, and it is now forming the foundation of the advanced LLMs which highly accelerated the advancement of NLP. This chapter explores the inner workings of Transformers, and discusses several types of the T5 family models, which are using Transformer in their architectures, like mT5 and ByT5 that we have adopted in this research.

3.1 Transformers

Attention mechanism revolutionized how models process and learn from data. It allows models to selectively focus on relevant information, as humans unconsciously prioritize key words during reading. While LSTMs and GRUs enhanced memory retention over traditional RNNs, they remain limited in handling extremely long

sequences, attention mechanism addressed this limitation by allowing the models to focus on related information of the sequence regardless of its length. Also, in traditional encoder-decoder models, the encoder summarizes all the input sequence words into one fixed-length vector, leading to loss of information. Equipping this type of model by attention mechanism provides a weighted combination of all encoder states which allow the decoder to access all relevant pieces of information. In addition, attention mechanism allows for parallel processing, making the training much faster and much more scalable.

All these advantages prepared the ground for another powerful architecture called Transformer which was introduced by Vaswani, *et al*. (2017). Transformers eliminate the need for any recurrence (RNNs, LSTMs, GRU) and rely only on attention mechanism. Their work achieved state-of-the-art results on translation tasks particularly German-to-English and French-to-English.

3.1.1 Transformer Architecture

The Transformer kept the encoder-decoder concept but without any recurrence or convolutional units. It uses stacked self-attention and fully connected layers, in addition to embedding layers, positional encoding, and residual connections. Figure 1 shows the architecture of the Transformer.

Each of the encoder and decoder consists of six identical layers, the encoder represents the input sequence in a matrix based on the context. Each matrix has a dimension of $T \times d_{model}$, where T is the number of tokens and d_{model} is the embedding size. This matrix is used then by the decoder to regressively generate the output.

The Transformer encoder comprises two basic components: a multi-head self-attention mechanism and a position-wise feed-forward network. The self-attention

mechanism allows the model to concentrate on the similarity between each token and all other tokens in the sequence which adds contextual information. The model includes multiple attention heads, each analyzes different aspects of the input, such as grammar and semantics. Then, the feed-forward network performs a sequence of modifications on each token separately to refine its representation. The model then uses residual connections and implements layer normalization to maintain stability and efficiency during training.

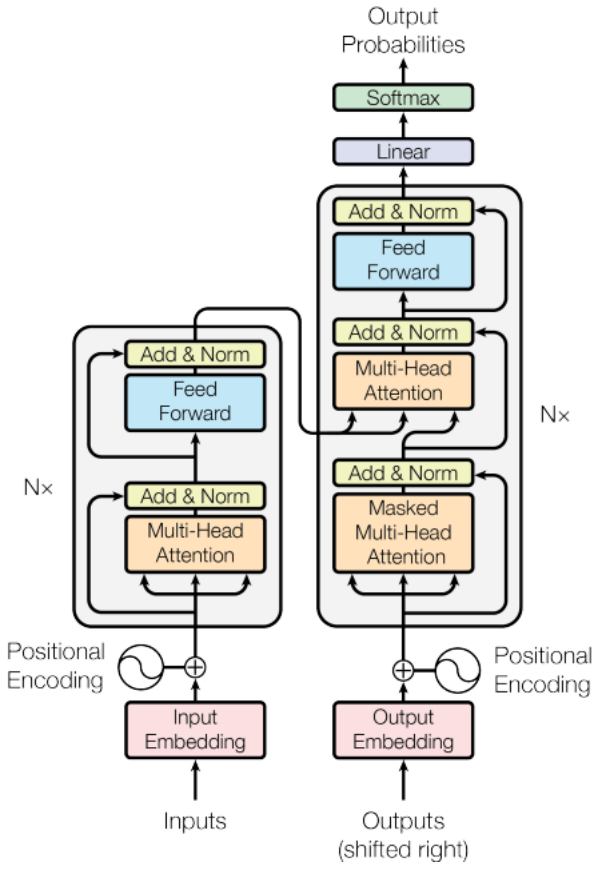

Figure 1. Transformers Structure (Vaswani, et al, 2017).

The decoder comprises three basic components. Starting out, it utilizes a masked self-attention process, like the encoder's self-attention, but with a modification that prevents the model from accessing subsequent words. This ensures that the model

predicted the text based on already generated words. Next, the decoder utilizes an additional attention mechanism to reference the encoder's output and retrieve related data from the starting sequence. Ultimately, like the encoder, it utilizes a feed-forward network to enhance its representation. Like the encoder, residual connections and layer normalization are employed to ensure stable and effective training process. Together, these levels enable the decoder to provide output sequences that are both precise and contextually significant.

3.1.2 Attention Mechanisms

Attention mechanisms have a major role in advancing models, particularly in NLP. Additive Attention mechanism is one of the earliest mechanisms introduced by Bahdanau, *et al*. (2015). This mechanism computes attention scores using a feed-forward neural network with a single hidden layer which allows the model to learn complex alignments between input and output sequences. However, its reliance on a neural network makes it computationally expensive.

To address this, Multiplicative Attention, proposed by Luong, *et al*. (2015), simplified the process by replacing the neural network by dot product or a learned matrix multiplication. This reduced computational costs and is still achieving competitive performance. This advancement in attention mechanisms improved tasks like machine translation and text summarization.

Based on these advancements, the Transformer architecture introduced by Vaswani, *et al*. (2017) adopted a highly efficient and scalable attention mechanism called Scaled Dot-Product Attention. Unlike previous mechanisms, Scaled Dot-Product Attention computes attention scores by applying the dot product of queries (Q) and keys (K) , scaling the result by the square root of the dimensionality of the keys $\left(\sqrt{d_k}\right)$ which

mitigate growing of the dot products to very large values, leading to diminished gradients and unstable training, followed by applying a softmax function to obtain attention weights. These weights are then used to compute a weighted sum of the values (V). Mathematically, it is defined as:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (3.1)$$

This mechanism is used in both the encoder and decoder of the Transformer, enabling the model to capture long-range dependencies and contextual relationships. Combining advantages of the Scaled Dot-Product Attention, it became the in core of the Transformer architecture, improving performance in tasks like machine translation and text generation.

3.1.3 Feed Forward Networks

The location-wise Feed-Forward Network (FFN) functions separately on each token position within the sequence, providing an identical set of modification processes for each token. The FFN consists of two linear transformations with a ReLU (Rectified Linear Unit) activation function in between, defined as:

$$FFN(x) = max(0, xW_1 + b_1)W_2 + b_2 \quad (3.2)$$

where x represent the input to the FNN, W_1 and W_2 represent the learnable weight matrices, and b_1 and b_2 represent the bias terms. This structure enables the network to incorporate non-linearity converting the input into a higher-dimensional space before reverting it to the original dimensionality. The FFN serves a key part in further improving the representations generated by the self-attention process, ensuring the model recognizes intricate patterns and correlations within the data. The Transformer uses the identical

FNN to each position, ensuring consistent processing across the sequence while maintaining the potential to learn position-specific features.

3.1.4 Positional Encoding

Order of tokens is important information in sequences, particularly sentences. However, there is no more recurrence or convolution in the Transformer models. To accommodate this loss, a positional encoding was introduced. Positional encoding can be learned or fixed. For example, Gehring, *et al*. (2017) used learned position embeddings. Meanwhile, they are learned during training. Vaswani, *et al*. (2017) used fixed positional encoding which provides a predefined structure for positional information using the following sinusoidal functions each with different frequency.

$$PE_{(pos,2i)} = sin\left(pos/10000^{2i/d_{model}}\right) \quad (3.3)$$

$$PE_{(pos,2i+1)} = cos\left(pos/10000^{2i/d_{model}}\right) \quad (3.4)$$

Where (*pos)* is the position of the token in the sequence, (*i)* is the dimension of the positional encoding (0, 1, 2, …., $d_{model} - 1$), and (d_{model}) is the dimensionality of the embeddings.

Vaswani, *et al*. (2017) also experimented with a learned positional encoding and found that they both, fixed and learned, have similar performance. However, they chose the sinusoidal function because it is simpler and can be more generalized to loner sequences.

3.2 Transfer Learning

Transfer learning is a powerful technique that enables fast advancement in many downstream tasks, as it enables models to leverage knowledge from one task or domain to improve performance on another downstream task. Traditional machine learning

approaches are based on training the model from scratch, while transfer learning enables models to handle other tasks building on pre-trained representations. This can significantly decrease the need for huge data sets, computational resources, and training time.

Transfer learning enables fine-tuning a pretrained model, which was trained with huge datasets, on a downstream task with less data. That is, the knowledge gained from the original task (e.g., language modeling) can be transferred to the target downstream task (e.g., sentiment analysis). This approach can be very useful in NLP where the source models can be trained using the abundant unlabeled data, then fine-tuned using the limited labeled data.

Development of pre-trained language models, such as BERT (Bidirectional Encoder Representations from Transformers), that was introduced by Devlin, *et al*. (2019), GPT (Generative Pre-trained Transformer), that was introduced by Radford, *et al*. (2018), and T5 (Text-to-Text Transfer Transformer), that was introduced by Raffel, *et al*. (2020), accelerated the current advancement in NLP. For example, BERT, which is based on bidirectional Transformers architecture, was pre-trained on masked language modeling and next sentence prediction. Then it achieved state-of-the-art results on downstream tasks like question answering and sentiment analysis. GPT, which is based on unidirectional Transformers architecture, was trained to predict the next word in a sequence. It then has been successful generative tasks such as dialogue generation. T5 is a text-to-text model, it was pre-trained on a diverse set of tasks, making it capable of excelling in a wide range of downstream tasks.

Transfer Learning still has limitations such as domain adaptation, computational cost needed in pre-training, and ethical concerns including bias and fairness. Also, it is still not perfectly generalized for a specific domain with their own linguistic attributes like

telecom engineering or legal texts. Additionally, pre-training and fine-tuning of LLMs, such as GPT-3, still need high computational resources.

3.3 Text-to-Text Transfer Transformer

Raffel, *et al*. (2020) introduced the Text-to-Text Transfer Transformer (T5) model which adopted a unified text-to-text framework, meaning the model always receives text as input and produces text as output. Unlike previous models, which were designed to handle a specific task, this model was designed to handle a wide range of tasks, including translation, question answering, and classification, all formulated as a text generation problem. This unified format allows single model architecture, training objective, and decoding strategy across multiple tasks.

3.3.1 T5 Model Architecture

T5 model utilizes the encoder-decoder Transformer introduced by Vaswani *et.al* (2017). It is almost like the original Transformer with some modifications. Both the encoder and decoder consist of multiple stacked layers (blocks), the encoder takes an input sequence, and the decoder autoregressively generates the output sequence.

Encoder:

The encoder consists of multiple stacked layers (blocks), each one contains self-attention, which is needed to calculate the relative relation for all token in the input and ultimately capture the contextual relationships. FFN is also a component in each block. Relative position embeddings were adopted, instead of the fixed positional to enhance the ability of generalization. The bias in the normalization layer was removed, and the normalization layer was placed outside the residual path.

Decoder:

The decoder also consists of multiple stacked layers (blocks), which still utilizes self-attention mechanism, but with masking, allowing each position to attend to earlier one, maintaining the autoregressive property. In addition to the masked self-attention, the decoder utilizes cross-attention with the encoder, allowing it to attend to the encoder's output, using the input's context during texts generation. FNN and relative position embeddings are also utilized in the decoder. Like the encoder, the bias in the normalization layer is absent, and the normalization layer is placed outside the residual path.

The number of layers vary across T5 variants (T5-Small, T5-Base, T5-Large, T5-3B, and T5-11B), ranging from 6 to 24 layers. T5 variants differ in not only the number of layers, but also the attention heads, and hidden sizes which directly affect the model's capacity to capture linguistic nuances. Also, Dropout is widely used across the network to prevent overfitting.

3.3.2 T5 Model Approach

The T5 model was pre-trained using a huge dataset called colossal clean crawled corpus (C4), which is a cleaned version of the publicly available Common Crawl dataset. Cleaning approach ensured inclusion of complete, natural language sentences exclusively in English. The authors used SentencePiece tokenizer which was introduced by Kudo & Richardson (2018) with a size of 32,000 tokens. The denoising objective was adopted in the pre-training process, where the model is trained to predict the masked or corrupted tokens. In addition, T5 was pre-trained on several auxiliary tasks, such as translation, summarization, and question answering. This approach enables the model to learn task-specific patterns and improve its generalization capabilities.

T5 model treats both input and output as texts, without requiring different types of tokenization or specific architecture in the output layer. The input text is prefixed with a description of the required task. For example, in translation task, the input text might be “Translate English to German: My name is Rabie” and the output would be “Ich heiße Rabie”. Same for summarization task, the input might be “summarize: Long time ago, the boy, who lived in the jungle with the animals, became a hero” and the output might be “The jungle boy became a hero”. This approach streamlines the usage of transfer learning in a specific task.

The T5 authors targeted to push the boundaries of what transfer learning could do, so they carried out several experiments. They basically tried several combinations, tweaking the model’s hyperparameters, trying different training goals, and playing with how they fine-tuned it. They concluded that larger models usually did better, but they noticed the improvements weren’t as dramatic as the models got huge. The biggest one, T5-11B, outperformed a lot of benchmarks, but it also needed huge computing power, which is something they had to consider. In addition, they discovered that training the model on a bunch of different tasks at once really helped in generalization on a different downstream task.

3.4 A Massively Multilingual Pre-trained Text-to-Text Transformer

The T5 model was pre-trained on English-centric tasks only and outperform several benchmarks. However, this monolingual focus, rooted in English-language pre-training, left room for exploration in multilingual contexts. Addressing this gap, Xue, *et al*. (2020) introduced the Massively Multilingual Pre-trained Text-to-Text Transformer (mT5), a significant extension of the T5 framework designed to handle the complexities of over 100 languages simultaneously.

3.4.1 mT5 Model Architecture

Keeping the paradigm of the T5 mode, mT5 still deals with all tasks as a text-to-text problem. Unlike T5 pre-training dataset, mT5 was trained using datasets that compromise more than 100 languages, including Arabic, which makes it more versatile. Also, it utilizes tokenizer trained on a multilingual corpus with 250,000 word-piece vocabulary, making it more suitable for tasks involving complex natural languages.

mT5 model authors persisted the architecture of the T5 model including the encoder-decoder transformer's design, the multi-head self-attention, the feedforward network, and the relative positional encoding. Like the T5 model, mT5 model has variants, each represent a different scale with alert in numbers of layers, attention heads, and hidden dimensions in some variants. For example, the largest T5 model comprises 11B parameters, while mT5 expanded this number to 13B.

3.4.2 mT5 Model Approach

mT5 adopted the same denoising approach, where random tokens are masked or corrupted and the model learns to reconstruct them, but applies it over a multilingual corpus, making it able to be used in downstream tasks of different languages.

A dataset, derived from the Common Crawl web data, called mC4 (multilingual colossal clean crawled corpus), is used in the training process. It follows same filtering approach that was used in C4 except that it included 101 languages, instead of only English, which produced texts with more than 10,000 pages. To handle this vast dataset, they pre-trained the model for 10^6 steps using a batch size of 1024 and length-1024 input sequences. mC4 is not balanced dataset. Meanwhile, deviation in each language proportion is relatively high. To address this, the authors used a language-agnostic-

tokenizer that treats all languages equally, which ensures that the model is not biased toward languages with high-proportion.

To see how well mT5 performed, the authors evaluated it on a variety of multilingual tasks, including machine translation, text classification, question answering, and summarization. The results were quite compelling, with mT5 consistently outperforming other leading multilingual models like mBERT (Devlin, *et al*., 2019) and XLM-R (Conneau, *et al*., 2019) across numerous benchmarks. What's particularly noteworthy is how the text-to-text framework allows mT5 to seamlessly handle a broad spectrum of tasks without needing any task-specific tweaks which reflect a real testament to its versatility in multilingual NLP. Furthermore, its ability to generalize across languages and tasks makes it incredibly effective for situations where cross-lingual transfer is key. Like any model, mT5 isn't without limitations. It still faces challenges, especially when it comes to the sheer computational resources required and the potential for bias towards languages with abundant training data.

3.5 Byte Level Text-to-Text Transfer Transformer Model (ByT5)

Building upon the foundation of T5 architecture, Xue, *et al*. (2022) introduced the Byte-Level Text-to-Text Transfer Transformer (ByT5). It operates texts at the byte level, rather than word or sub-word level. This design choice fundamentally alters the input processing, enabling ByT5 to handle text as a direct sequence of bytes, effectively removing the requirement for explicit tokenization.

In early grades in school, children learn the alphabet first, then they use them in building words and sentences. This approach of learning is mimicked in token-free models. Token-free models offer several advantages compared to NLP models that need traditional tokenization methods. As they are language-agnostic, they can handle texts in any language with no need for a specific tokenization process, which represents a

significant shift in how text is processed in transformer-based models. Particularly speaking about the ByT5 model, like its peer models T5 and mT5, it still treats all tasks as a text generation problem. Meanwhile, it retains the text-to-text framework of T5, but its byte-level processing enables it to generalize better across diverse inputs.

T5 and mT5, process input texts as sub-words, which limits their ability in handling noisy texts, and languages with complex structures like Arabic while ByT5 addresses these limitations by operating directly on byte level, treating each byte as a token. This approach eliminates the need for fixed vocabulary, making ByT5 highly flexible and capable of handling noisy texts that include special characters or emojis, and it stamps out the out-of-vocabulary issue. However, training a token-free model is more complex and more computationally expensive as the model needs to capture patterns on byte level. In addition, characters in different languages may be represented in more than one byte, like Arabic, which increases sequences length. Despite these drawbacks, the flexibility and robustness in such models make them a powerful tool for a wide range of NLP tasks especially for languages with complex structures.

3.5.1 ByT5 Model Architecture

ByT5 is a direct descendant of the T5 and mT5 models, sharing their core architecture and text-to-text framework, particularly, it was based on mT5 model, which was trained using mC4 dataset. The aim of ByT5 was to address the same use cases as mT5, functioning as a general-purpose multilingual text-to-text model. It inherits the multilingual capabilities of mT5 but extends them further by handling any language or script without requiring language-specific tokenization rules.

Transformer-based encoder-decoder architecture is still persistent in the ByT5 model just like T5 and mT5. The major difference is in the tokenization technique. ByT5 tokenizer breaks input text into bytes (8 bits), each byte is considered as token. This

sequence of bytes is then processed through the self-attention blocks and feedforward networks just like T5 and mT5. A byte can represent 256 distinct values which results in a vocabulary size of 256.

Byte-level processing produces a longer input sequence, this issue requires a modification in the design to handle it. The authors used deeper layers compared to T5, for example, ByT5 base uses 24 layers in encoder compared to 12 layers in the T5 base model. Another adjustment in the design is that, unlike T5 and mT5, ByT5 uses inconsistent number of layers in the encoder and decoder. Table 2 shows the number of layers in both encoder and decoder for ByT5 and T5. In most cases, the encoder's layer number was increased to better capture and contextualize the longer input sequences, while the decoder was kept the same, except in ByT5 small, to keep it efficient during output generation.

Table 2. Comparison of Encoder and Decoder Layer Depth in ByT5 and T5 Architectures.

Model Variant	ByT5		T5	
	Encoder	Decoder	Encoder	Decoder
Small	12	8	6	6
Base	24	12	12	12
Large	24	24	24	24
XL	36	24	24	24
XXL (11B)	36	24	24	24

3.5.2 ByT5 Model Approach

ByT5 was pre-trained using the same dataset used in mT5, with different tokenization approach, which process UTF-8 bytes directly into the model without any preprocessing. It also follows the same denoising objective, but with longer byte-spans masking. Specifically, they used mean mask span length of 20 bytes, while it was 3

subword tokens in mT5. Like mT5, sequence length was set to 1024 bytes (rather than tokens) and trained for 10^6 steps, using Adafactor optimizer, and batch size of 2^{20}.

3.5.3 ByT5 for Arabic Language Tasks

Arabic is a complex-structured language, its connected letters, optional diacritics, unruly dialectal shifts, and word forms that spring up from a single root makes it a language that necessitates models capable of capturing fine-grained morphological and orthographic details. Systems that are leaning on word or subword tokenization and fixed vocabularies, often stumble over these hurdles.

Arabic script is marked by its complex and nuanced structure. Diacritics are little marks like فَتْحَة (fatha), كَسْرَة (kasra), or ضَمَّة (damma), that can change a word's meaning entirely. In casual texts, they are often skipped, leaving ambiguity that humans might guess but machines struggle to resolve. Traditional tokenizers, chopping words into subword pieces, can't always keep up, mangling diacritized and bare text alike. ByT5, processing input as raw byte sequences, exhibits insensitivity to the presence or absence of diacritics. This makes it a natural fit for tasks like diacritization restoration, where it can predict and place those marks with precision. Then there's the connected script letters morphing depending on their spot in a word. That's a headache for subword systems, but ByT5 just sees a string of bytes, unbothered by shape-shifting letters, letting it glide through Arabic's orthography effortlessly.

Dialects add another layer of complexity. Dialects like Egyptian, Levantine, or Gulf stray away from MSA in words, grammar, even how they sound. Jordanian dialect, for instance, might toss out "شو" for "what" instead of MSA's "ماذا," or sprinkle in phonetic spellings and shorthand that defy standardization. Models that utilize traditional tokenization methods choke on this. ByT5, though, thrives on messiness. Its knack for

handling noisy informal text honed by training with random byte dropouts means it can take dialectal chaos and turn it into polished MSA, even if the input's a jumble of quirks.

Arabic's morphological maze is also another challenge. A single root can sprout in many forms. For example, "كتب" spinning into "كاتب" (writer), "مكتوب" (written), and beyond. Traditional tokenizers often miss these patterns, especially with rare words that don't fit their vocabulary. ByT5 overcome this issue as it works at the byte level, so it can handle oddball forms or new derivations.

Given its natural ability to manage the complexity of translating Jordanian dialect to MSA with the additional dimension of possible diacritization, ByT5 is a suitable model for the research. Its byte-level processing proves strong against the noisy character of dialectal text and helps it to properly manage the variants and informalities in Jordanian dialect, therefore avoiding the requirement for a predetermined vocabulary. Moreover, ByT5's flawless handling of diacritics, treating them as ordinary bytes, ensures that the model can precisely predict and insert them, independent of their presence in the initial input, hence fitting a multi-task aim. ByT5's text-to-text framework easily fits the multi-task approach, allowing to create both dialect-to-MSA transformation and diacritization as text generation tasks, hence simplifying the training process and boosting performance by shared learning. Furthermore, major difficulties in Arabic NLP include ByT5's treatment of Arabic's connected script and its abolition of out-of-vocabulary problems.

Chapter Four

Datasets Preparations and Experimental Setup

Datasets used in training of any model is a milestone in its performance, especially when training deep transformer-based models like ByT5. This thesis research utilized two primary datasets, JODA and the filtered Tashkeela datasets (Clean-50 and Clean-400), for fine-tuning the ByT5 model.

JODA is a dataset of paired sentences, each containing a Jordanian dialect and an MSA version. It was developed by Abandah, *et al*. (2025) to be used in NLP specially for texts in Jordanian dialect. Clean-50 and Clean-400 constitute high-quality text subsets of the larger Tashkeela dataset, a corpus widely employed in the development of Arabic NLP. Both datasets were prepared to enable the model to perform the two tasks: correcting orthographic errors in dialect or MSA to standard MSA and then adding diacritics to the corrected MSA based on practitioner preference.

To accurately gauge the model's proficiency in both orthographic correction and diacritization, a fair evaluation was performed using appropriate metrics, ensuring that the results reflected the model's true ability to handle each task.

This chapter covers the datasets that were used, and the preparation process. Each of them was used to spotlight specific aspects of the final objective of multi-tasking. It also discusses the experiment environments used for both data preparation and model training. In addition, we discussed the approach to evaluating this multi-task model.

4.1 Data Corpus Development and Preparation

This section details the crucial preprocessing steps undertaken to prepare the Tashkeela and JODA datasets for multi-task learning with the ByT5 model. Given the absence of diacritics in the MSA version of the JODA sentences and the fact that the

Tashkeela dataset (specifically Clean-50 and Clean-400) consists solely of diacritized sentences without input-output pairs or orthographic errors, proper preparation was essential. This section will elaborate on the specific procedures applied to the Tashkeela dataset, specifically the Clean-50 and Clean-400 subsets, and the JODA dataset, highlighting how each was adapted to facilitate orthographic correction and diacritization tasks.

4.1.1 Tashkeela Dataset Preparation

The Tashkeela dataset is a widely used resource for Arabic text diacritization, comprising a large collection of diacritized texts that serve as a foundation for training and evaluating diacritization models. The following subsections will provide an overview of the dataset and its subsets, Clean-50 and Clean-400. Subsequently, we will describe the process of synthetic error injection, a crucial step in preparing the dataset for our multi-task learning objectives, which involved simulating orthographic errors to train the model's correction capabilities.

4.1.1.1 Tashkeela Dataset Overview

The Tashkeela dataset was collected by Zerrouki and Balla (2017) from diacritized texts that were crawled from the internet. They are mostly retrieved from free published books, particularly Islamic classical books. Tashkeela dataset consists of about 75.5 million words, 98.85% of them are formed in CA. Two subsets of the Tashkeela dataset were introduced, Clean-50 and Clean-400. These subsets provide high-quality texts achieved through the application of a series of filters to ensure purity, cleanliness, and overall quality.

Fadel, *et al*. (2019) extracted 55,000 sequences from the Tashkeela dataset with a diacritization percentage of 80%, meaning that 80% of the characters within the dataset

included diacritical marks. This dataset, referred to as Clean-50, was split into training, validation, and test sets consisting of 50,000, 2,500, and 2,500 samples respectively. Cleaning procedure included correction of some diacritics issues, removing English letters and HTML tags, removing the special Arabic character *Tatweel*, which does not alter the meaning or add any information, separate numbers from words, and removing multiple white spaces.

Karim and Abandah. (2021) used the same approach that was used by Fadel, *et al*. (2019) in filtering and cleaning Tashkeela dataset to extract a larger training dataset consisting of 400,000 sequences, referred to as Clean-400, keeping same validation and test sets.

In our research, several maximum input and output sequence lengths were tested to balance performance and computational cost. The approach used to cope with the sequences relies on processing full sentences to preserve full contextual information. Full stop, comma, and dotted comma were adopted as the marks to identify full sentences. For example, with a maximum sequence length of 208 bytes (104 characters), the sentence is truncated at the first mark starting from left (end of the sequence). The choice of lengths was inspired by the results from Al-Rfooh, *et al*. (2023) work, which revealed that longer sequence length (512 bytes) yielded the best performance. Table 3 shows the maximum input and output sequence lengths tested in this research. The maximum output sequence length was capped to a specified value, and the corresponding maximum input sequence length was determined accordingly. When combining Tashkeela dataset with other datasets (JODA or REALMS), the maximum sequence length after the combination was adopted, regardless of whether it exceeded that of the Tashkeela dataset.

Table 3. Maximum Input/Output Sequence Length in the Prepared Clean-50 and Clean-400 Datasets.

Max Input Sequence Length (bytes)	Max Output Sequence Length (bytes)
208	256
366	512
700	1,024

4.1.1.2 Synthetic Error Injection

The Tashkeela dataset, specifically its Clean-50 and Clean-400 subsets, was combined with the JODA dataset to expand the overall training data and help the model capture more linguistic nuances. Since the JODA dataset contains sentences written either in the Jordanian dialect or in MSA with varying degrees of human errors, with dialectal texts sometimes including MSA words with or without mistakes, an effort was made to align the characteristics of the Tashkeela dataset with those of JODA. To achieve this, synthetic errors were injected into the Tashkeela dataset.

Orthographic mistakes are common among native and foreign Arabic speakers, either in MSA or in dialectal form. This stems from intricacies of the language's rules related to grammar and morphology which dictate the words formatting to be altered based on their syntactic position in the sentence contextual factors. Also, the dialectal variations often introduce major or minor changes that could be considered as errors in the context of transferring to MAS.

Claiming that integrating Clean-50 or Clean-400 would improve performance after fine-tuning the model on the JODA dataset alone, error injections were integrated into Clean-50 and Clean-400 to simulate real errors and enhance the model's generalization capabilities. Two different error techniques were stochastically injected into the input, undiacritized, sequences with specific error probabilities. The first error injection

technique includes replacement of specific letters and the second introduces random letters modification such as exaggeration and deletion.

Soft Orthographic Error Injection

Soft spelling errors are a form of lexical and semantic errors, arising from orthographic variations of some letters that often confusing even native speakers, like the *hamza*, that have different forms based on its position, its diacritic and surrounding diacritics. Other widespread errors are related to *alef* variations, and *teh marbuta*. Using artificial errors injection may not fully capture the nuances of real-world mistakes. However, simulating such mistakes may help models to be more robust against real errors.

We adopted the methodology presented by Abandah, *et al*. (2022) to inject such type of errors into the input of Clean-50 and Clean-400 datasets. This approach targeted letters that are exposed to be confused, such as variations of the *alef*, *hamza*, and *teh marbuta*, by replacing them with other members of predefined letter groups. With an error injection rate, random letters are selected from these groups and are substituted with alternative forms from the same group. For instance, an *alef maksura* could be replaced with an *alef*, or vice versa, mimicking typical confusions. The probability of these substitutions was controlled by the error injection rate, which was fixed to 10% in our research. Table 4 shows letters that are altered with a corresponding possible replacement.

Table 4. Replacement Groups for Soft Orthographic Error Injection (Abandah, et al., 2022).

Original Letters	Possible Replacements
ء ، آ ، أ ، ؤ ، إ ، ئ	ا ، ء ، آ ، أ ، ؤ ، إ ، ئ
ا	ى ، ء ، آ ، أ ، ؤ ، إ ، ئ ، ا
ة	ة ، ت ، ه
ى	ى ، ا
(at end of word) اء	ا ، ء ، آ ، أ ، ؤ ، إ ، ئ ، ا
(at end of word) و ، وا	و ، وا
(at end of word) ه	ة ، ه
(at end of word) ت	ة ، ت

Random Orthographic Distortion

This research implemented a comprehensive stochastic error injection technique, denoted as Random Orthographic Distortion, which expands upon the methodology proposed by Khaleel and Abandah (2025). This approach transcends the simulation of soft-spelling mistakes, aiming to replicate a broader spectrum of common error patterns encountered in Arabic texts. Specifically, artificial errors were introduced through letter exaggeration (randomly repeating a letter two to four times), deletion, insertion, swapping, and replacement.

During dataset preparation, input sequences were processed, and the aforementioned error injection patterns were stochastically applied to Arabic letters based on a pre-defined probability distribution. A fixed error injection rate of 5% was consistently utilized across the dataset. The selection of specific error injection patterns (exaggeration, deletion, insertion, swapping, or replacement) was determined according to a predetermined probability distribution. Each error form, except for random letter replacement, was assigned a probability of 10%. Random letter replacement was assigned a 60% probability. This methodological approach was designed to enhance the model's capacity to correct realistic orthographic distortions, thereby improving its robustness and generalization capabilities in handling authentic Arabic text. Table 5 clarify how this methodological approach could modify words by showing example for each error form.

Table 5. Examples for Error Injection Patterns of Random Orthographic Distortion.

Error Form	Original word	Modified word
Exaggeration	السلام عليكم	السلاام عليكم
Deletion	الأرض	الرض
Insertion	خضراء	خضبراء
Replacement by a random letter	سبانخ	سبانغ

4.1.2 JODA Dataset Preparation

JODA is a specialized dataset, specifically providing parallel sentences in Jordanian dialect and MSA. This dataset was compiled from several sources, including social media platforms such as Facebook, YouTube, Instagram, and Twitter, as well as existing online datasets like the Shami-Corpora and DART-Dataset. The following section presents a description of the JODA dataset, detailing the diacritization work carried out to facilitate its utilization within this research thesis.

4.1.2.1 JODA Dataset Overview

JODA dataset contains Jordanian dialect sequences, and MSA sequences with minor spelling discrepancies. The total number of samples is 59,135 divided into training, validation, and test subsets using stratified sampling to maintain a balanced distribution of data from different sources and language types (Jordanian dialect and MSA).

Table 6 shows number of samples per split type and Figure 2 shows statistics about samples per type (Jordanian dialect and MSA) for each split.

Table 6. JODA Dataset Number of Samples per Split Type.

Split	Train Set	Validation Set	Test Set
Number of Samples	54,135	2,500	2,500
Percentage	92%	4%	4%

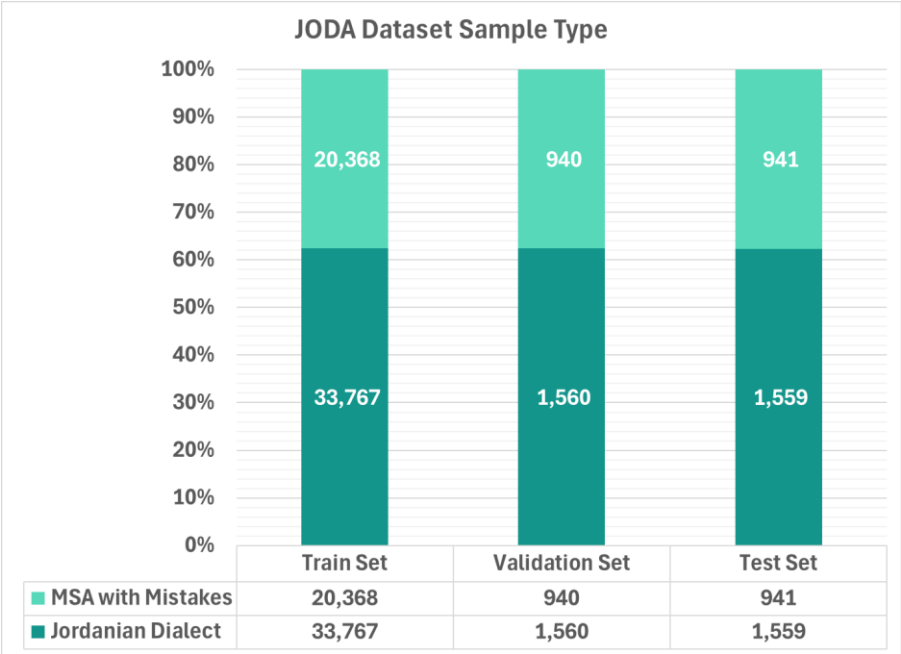

Figure 2. JODA Dataset Statistics per Sample Type.

4.1.2.2 JODA Dataset Diacritization

The JODA dataset is valuable resource, offering parallel texts in both Jordanian dialect and MSA. However, we encountered a significant hurdle: the MSA texts lacked diacritics. These small markings are quite important for reading and understanding Arabic accurately, especially in more formal contexts. Since the research aimed to create a reliable way to convert dialect to MSA, ensuring the MSA text was diacritized became a necessary first step. Although adding these diacritics wasn't the focus, it did require considerable effort and was essential for everything else we wanted to achieve. To address this, we employed the ByT5 model, which demonstrates a strong capacity to capture the nuances of Arabic orthography, including complex diacritics that depend on various linguistic factors.

The Clean-400 dataset served as the primary resource for fine-tuning of ByT5 model to act as our Arabic diacritization model. This dataset is characterized by its high-quality text with diacritization ratio of at least 80%, making it an asset for training models that

aim to accurately predict diacritical marks. To prepare the Clean-400 dataset for the fine-tuning process, we initially removed all diacritics from the text. This step resulted in a set of un-diacritized input sequences that were then paired with their corresponding fully diacritized versions from the original dataset. These pairs of un-diacritized input and diacritized target sequences formed the training data for the ByT5 model, which was implemented using the PyTorch Lightning framework.

Due to limitations in available computational resources, we utilized the ByT5-small model for fine-tuning. This model was trained on the Clean-400 dataset, and hyperparameters such as learning rates and maximum input and output sequence lengths were tuned to optimize performance.

The model was trained using the AdamW optimizer, a standard choice for transformer-based models, and the loss function cross-entropy, the default loss function, which is well-suited for sequence generation tasks. We also implemented early stopping, monitoring the validation DER to prevent overfitting. This approach allowed us to identify the optimal training epoch, preventing unnecessary computational overhead and maintaining a balance between model performance and training efficiency. After careful evaluation, we selected the configuration that yielded the best balance of performance and computational cost.

The experiments were conducted on a system equipped with an Intel Xeon CPU, a Quadro RTX 5000 GPU, and 188 GB of RAM. The operating system used was Ubuntu 22.04.3 LTS, and the experiments were implemented using Python with relevant deep learning libraries. For development, Jupyter Notebook was used for interactive experimentation, while PyCharm facilitated structured code development and debugging. The training process leveraged PyTorch Lightning to manage model training efficiently.

Our best model had maximum input sequence length of 1024, maximum output sequence length of 2048, learning rate of 0.003, batch size of 256, dropout ratio of 0.1 and due to computing resources limitations, half of the model's parameters were frozen. The model was trained for 1885 training steps, which took about 67 hours. Figure 3 shows the training progress curve for both validation accuracy and validation loss.

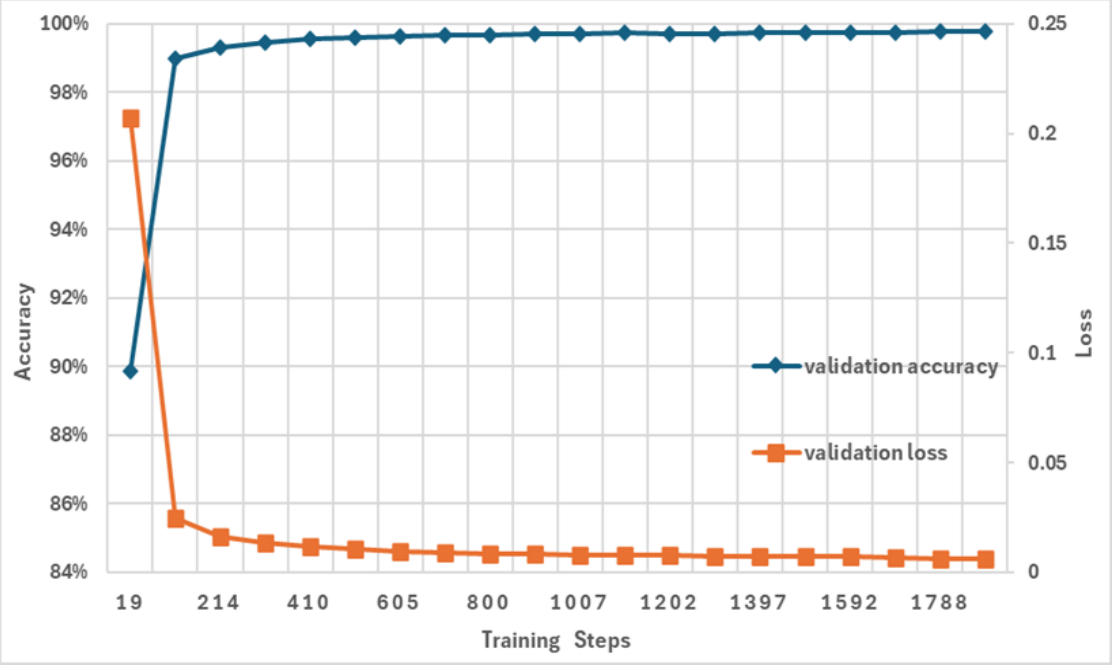

Figure 3. Diacritization Training Progress Curve for Validation Accuracy and Validation Loss.

The performance of the diacritization model was evaluated on Clean-400 test set using DER, which focused specifically on the accuracy of diacritic placement. We achieved a DER of 1.54%, including the diacritic of the last letter, which is the most frequently changed diacritic and is affected by the grammar, on the test set. Excluding the diacritic of the last letter, DER recorded a value of 1.1%. These results indicate that the model effectively learned to predict and insert diacritics, demonstrating its suitability for diacritizing the JODA dataset.

Because DER is a sensitive metric, as any extra or missing character from the sentence largely affects the value, we carried out further analysis of the predicted texts in

the test set. We identified 28 sequences where the model produced mismatches, indicating letter and word alterations in the original sentences. After re-evaluating the model's performance using only matched sequences, DER, including the last letter's diacritic enhanced to 1.25% and DER excluding the last letter's diacritic enhanced to 0.74%.

Based on these results, we applied the trained ByT5 model to diacritize the MSA column of the JODA dataset and considered only the matched sequences which were 52,948 out of 54,135. Each undiacritized MSA sentence was passed through the model, and the resulting diacritized output was integrated into the dataset. Table 7 shows samples of the diacritized JODA dataset.

Table 7. Samples of Diacritized MSA Sentences from the JODA Dataset.

Original JODA Text	Diacritized JODA Text
أو الذي يطمئن على أحوالي بين فترة وأخرى	أَوَ الَّذِي يَطْمَئِنُّ عَلَى أَحْوَالِي بَيْنَ فِتْرَةٍ وَأُخْرَى
أي بهذا المنظور؛ ماذا يكون التعليم؟	أَيْ بِهَذَا الْمَنْظُورِ؛ مَاذَا يَكُونُ التَّعْلِيمُ؟
لأن الحياة ليست فقط أكلا وشربا	لِأَنَّ الْحَيَاةَ لَيْسَتْ فَقَطْ أَكْلًا وَشُرْبًا

4.2 Preparing Datasets for Multi-Task Learning

To leverage the versatility of the ByT5 for our research objectives, we adopted a multi-task learning (MTL) framework. This approach, inspired by the original T5 model's design, allows a single model to address multiple related tasks by framing them all as text-to-text problems. Like the T5 authors' methodology, we utilized task-specific prefixes to delineate between different tasks, enabling the model to discern and generate the appropriate output for each input sequence.

Our MTL setup incorporated two primary tasks: The first task is orthographic correction, which involved transforming text from Jordanian dialect or MSA with errors into corrected MSA text, and the second one is correction and diacritization, which expanded on the first task by generating corrected MSA text with diacritics. To

distinguish between these tasks, we introduced prefixes to the input sequences. Specifically, the prefix "correct: " was used for the first task, and "diacritize: " was used for the second.

The selection of these prefixes was deliberate, prioritizing brevity to minimize the impact on input sequence length and, consequently, computational cost. Moreover, the prefixes were chosen to be distinct, ensuring that the model could effectively differentiate between tasks during training and avoid potential confusion. Figure 4 shows a diagram of our text-to-text framework with input/output example for each task. Each training sample for each task is started by a specific prefix, which allows to use same model for both tasks. These examples show the outputs produced by our best model given the corresponding inputs.

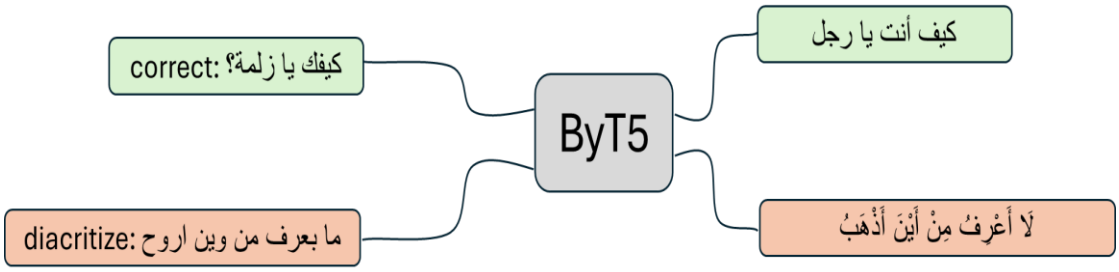

Figure 4. diagram for the ByT5 Model Text-to-Text Framework.

4.3 Experimental Setup

This section details the experimental environment and evaluation methodology employed to assess the performance of our ByT5 multi-task learning model. We outline the hardware and software resources utilized, as well as the evaluation metrics and procedures used to evaluate the model's effectiveness in handling orthographic correction and diacritization tasks.

4.4.1 Experimental Environment and Computational Resources

The training of the ByT5 model required substantial computational resources due to the byte-level granularity it works with. Initially, the diacritization experiments on the JODA dataset were conducted on a relatively low-resource machine, which may imposed constraints on the number of experiments and extended the training duration for fine-tuning of our main objective model. To address these limitations, we later transitioned to a more powerful machine to train the model efficiently.

We migrated to a more capable system equipped with an AMD EPYC CPU with 64 cores/128 thread and 256 MB L3 cache, 360 GB of RAM, and 2 NVIDIA H100 PCIe GPUs with 80 GB of VRAM. This upgrade to our training infrastructure enabled much better scale training with lower gradient accumulation steps and reduced training time significantly. Another machine with an additional 2 H100 PCIe GPUs was used in a limited number of experiments to further reduce the training time.

Both machines operated on Ubuntu 22.04 LTS. The software environment was managed using Anaconda, which was integrated into the system. A virtual environment was configured via the Ubuntu command line interface (CLI) to install the necessary libraries. Model training and evaluation were conducted using Python scripts executed directly through CLI. A summary of the computational setup is provided in

Table 8.

Table 8. System Specifications for MTL Training Machine.

Component	Specifications
CPU	AMD EPYC 9554 @3.1GHz, 60 cores, 997.6 MB Total Cache Memory
RAM	360 GB
GPU	2 × NVIDIA H100 PCIe, 80 GB VRAM, CUDA Version: 12.2, Kernel Version Installed on Image: 6.5 (HWE)
Operating System	Ubuntu 22.04 LTS
Deep Learning Framework	PyTorch (Lightning)
Libraries	lightning, pandas, torch, transformers, sklearn, torchmetrics, t5, pyarabic
Development Environments	Command Line Interface (CLI)

4.4.2 ByT5 Multi-Task Learning Evaluation

To comprehensively evaluate the ByT5 multi-task learning model, we employed distinct evaluation metrics tailored to the specific nature of each task: orthographic correction, by converting erroneous MSA and Jordanian dialect text into corrected MSA, and diacritization. The following metrics were utilized to evaluate the performance of the model throughout the conducted experiments:

BLEU (Bilingual Evaluation Understudy) Score:

The BLEU score is a widely used metric for evaluating the quality of machine-translated text. In our context, we used it to assess the fluency and accuracy of the model's

orthographic correction output, comparing it to the reference MSA text. It was introduced by Papineni, *et al*. (2002) as an inexpensive and language-agnostic method that largely correlates with human evaluation.

BLEU measures the n-gram overlap between the generated text and the reference text. N-grams are contiguous sequences of n words used to measure text overlap in NLP tasks. Lower n-grams (e.g., unigrams) focus on word-level accuracy but do not guarantee proper word order or sentence structure, while higher n-grams (e.g., 4-grams) capture sentence-level structure and fluency which ensures the correct word order and sentence structure. The BLEU score combines precision scores for multiple n-grams (typically up to 4-grams) to evaluate text quality, ensuring a balance between word choice and sentence structure. Higher n-grams contribute to a stricter evaluation, penalizing mismatches in word order and context.

We utilized a 4-gram setting for BLEU score calculation, which is commonly used in machine translation evaluation, meaning we considered sequences of up to 4 consecutive words. The score ranges from 0 to 1, with higher scores indicating better performance. A score of 1 indicates a perfect match between the generated and reference texts. It was exclusively used for the evaluation of orthographic correction.

Character Error Rate (CER):

Character error rate measures the number of character-level edits required to transform the model's output into the reference text. These edits include insertions, deletions, and substitutions. It is calculated by dividing the number of edits by the total number of characters in the reference text. A lower CER indicates better performance, with a score of 0 indicating a perfect match. CER was used to assess the model's precision in the orthographic correction task.

Diacritic Error Rate (DER):

Diacritic error rate specifically measures the accuracy of diacritic placement in the model's output. We used the DER calculation script introduced by Fadel, *et al*. (2019) to ensure consistency and comparability with previous research. Because DER is a sensitive metric to any wrong letter, DER was calculated for matched sequences only, because the model had already been penalized for errors in sequence evaluation through CER and WER. A lower DER indicates better performance, with a score of 0 indicating perfect diacritization. It was exclusively used for the evaluation of the diacritization task.

Word Error Rate (WER):

Word error rate measures the number of word-level edits required to transform the model's output into the reference text. These edits include insertions, deletions, and substitutions of words. It is calculated by dividing the number of edits by the total number of words in the reference text. A lower WER indicates better performance, with a score of 0 indicating a perfect match. WER was used to assess the overall accuracy of the model, considering both orthographic correction and diacritization. Therefore, it was measured for all sequences and is affected by wrong diacritics as well.

To provide a comprehensive assessment of the final model's performance, we aligned the metrics. Then, we calculated the average of all metrics, after transforming them to percentages, resulting in the *Evaluation Score* presented in Equation (4.1). This consolidated metric provided a holistic view of the model's effectiveness across both orthographic correction and diacritization tasks.

$$Evaluation\ Score = \frac{BLEU_{Score} + (100 - CER_{Score}) + (100 - DER_{Score}) + (100 - WER_{Score})}{4} \qquad (4.1)$$

4.4.3 ByT5 Multi-Task Learning and Code Validation

To validate the multi-task learning capabilities of the ByT5 model and to verify the integrity of our fine-tuning implementation using the aforementioned libraries, we conducted a numbers-to-words (N2W) and words-to-numbers (W2N) trial. This experiment served as a preliminary assessment of the model's ability to handle distinct text transformation tasks concurrently.

The trial was executed within the Lightning framework to ensure consistency and to validate the implementation's robustness. Model performance was evaluated based on inference loss and accuracy. The results demonstrated that ByT5 successfully learned and performed both N2W and W2N tasks, leveraging task-specific prefixes with high precision. Specifically, the model achieved a test loss of 0.0006 and a test accuracy of 0.9998. Table 9 shows examples of inference results, clarifying the expected input and output for the trial.

Table 9. Examples of Inference Results from N2W / W2N Trial Model.

Input	Prediction
N2W:8531	eight five three one
N2W:11111	one one one one one
W2N:seven nine six five	7965
W2N:five one one one	5111

These results confirm the efficacy of ByT5 in learning and executing multi-task transformations, providing empirical support for its applicability to more intricate linguistic tasks. The high accuracy and low loss values indicate that the model effectively utilized the task-specific prefixes to differentiate between N2W and W2N tasks,

validating our implementation and setting a solid foundation for subsequent experiments with more complex linguistic transformations.

Chapter Five

Experiments and Results

Chapter 5 presents our experimental journey to evaluate the ByT5 model's ability to perform two tasks: orthographic correction and correction with diacritization. Our objective is to create a single model that transforms Jordanian dialect or erroneous MSA into corrected MSA, offering users a choice between plain MSA output or MSA with diacritics. The first task corrects errors and translates to MSA, while the second restores diacritics in addition to the translation to corrected MSA. We implemented a prefix-based approach, using distinct prefixes in training and inference to signal the desired task. This chapter explores our experiments, aiming to validate the effectiveness of multi-task learning for Arabic text processing.

Our initial experiments focused on training with the JODA dataset alone. However, we hypothesized that incorporating the Tashkeela dataset would enhance the model's generalization capabilities. To explore this, we conducted a series of experiments using the JODA and Clean-50 datasets. Although we recognized that larger datasets generally improve model generalization, we opted for Clean-50 due to its lower computational demands, allowing us to efficiently trial various hyperparameter configurations.

This chapter outlines the experimental process for optimizing and evaluating our model for Arabic text processing. It begins by exploring different model architecture settings and then systematically investigates optimal training configurations, including key hyperparameters like the number of training cycles, learning rate, sequence length, and batch size. The chapter then analyzes the impact of various training datasets, starting with initial experiments and progressively incorporating larger and different data sources to determine the most effective training data. Finally, the chapter presents the configuration and performance evaluation of our best-performing model, including

quantitative metrics and a qualitative linguistic assessment of its capabilities in addressing Arabic text processing tasks.

5.1 Exploring Model's Architecture Parameters

Choosing the appropriate architecture parameters for the ByT5 model is an important step that we want to take to balance performance and efficiency. Our approach began with the determination of the model size, a decision informed by prior research. Notably, the authors of the ByT5 model, Xue, *et al*. (2022), reported that larger models generally exhibit superior performance. Consequently, we opted for the ByT5-Base model, which, with its 600 million parameters, offers a judicious trade-off between computational resources and model capacity.

Larger models, while capable of learning intricate patterns, are inherently susceptible to overfitting, particularly in scenarios with limited training data. This phenomenon, wherein the model memorizes the training set rather than generalizing to unseen data, poses a significant challenge. However, pre-training on extensive text corpora, as is the case with ByT5, can mitigate this risk, allowing larger models to benefit from smaller fine-tuning datasets. With 600 million parameters, ByT5-Base offers a good equilibrium of capacity and efficiency, aligning with our resource constraints and performance objectives.

Furthermore, we explored the impact of varying the trainable parameters Ratio, trialing ratios of 25%, 50%, 75%, and 100%. Our findings indicated that a 75% ratio outperformed the 25% and 50% ratios, although at the cost of increased computational burden. Notably, the performance difference between 75% and 100% was marginal, suggesting that the ByT5 model, having been pre-trained on a multilingual corpus including Arabic, had already captured essential characteristics of the Arabic language.

This observation underscores the potential for pre-trained models to leverage existing linguistic knowledge, reducing the necessity for extensive fine-tuning of all parameters.

To assess the impact of varying trainable parameter ratios, we established the 25% ratio as a benchmark. It is crucial to mention here that we adopted to change multiple parameters at a time in some experiments, enabling us to recognize a wider range of combinations. So, for example, results of 25% trainable parameters ratio may have different values as each one contains different hyperparameters. Also, it is important to mention that at early experiments we used JODA test set only, which is our benchmark and was used for the final verdict, while in later experiments, merging Tashkeela subsets, we added its test set for the sake of accurate parameter values evaluation, so there would be significant difference in the Evaluation Score. However, we evaluated the impact as ratio compared to equivalent experiments results. Each experiment with a different ratio (50%, 75%, and 100%) was then compared to a corresponding 25% experiment with otherwise identical settings. Figure 5 compares the 25% trainable parameter ratio benchmark with 50%, demonstrating a 0.72% increase in the total evaluation score, as calculated by Equation (4.1).

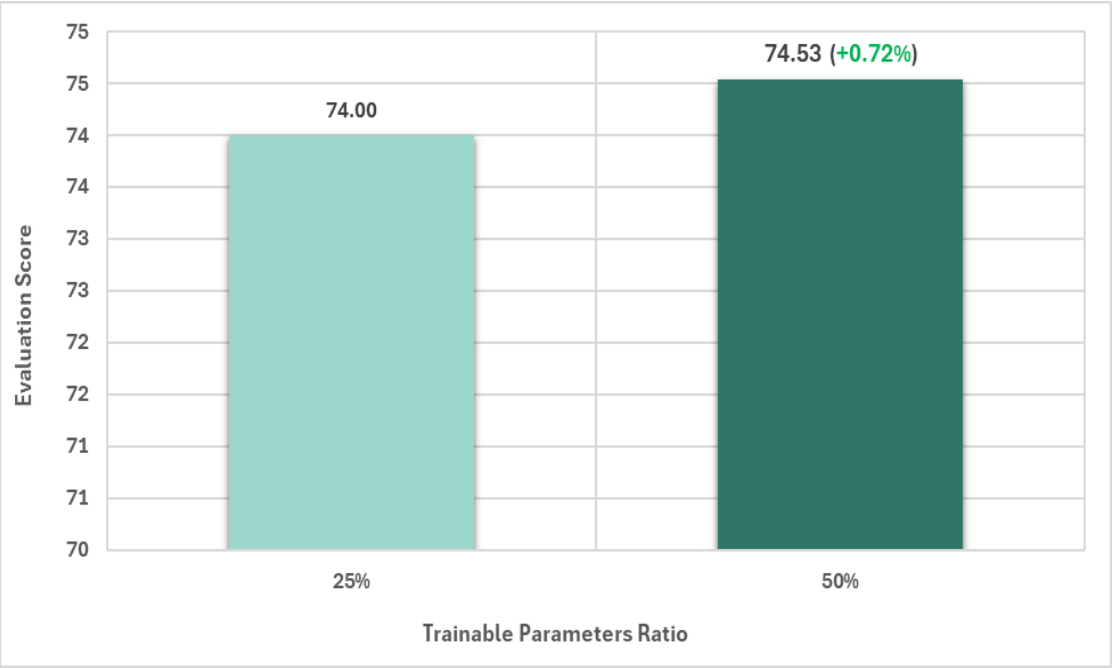

Figure 5. Comparison of Total Evaluation Scores between 25% and 50% Trainable Parameter Ratios.

As mentioned, different sets of experiments were used to simultaneously test multiple parameters. Different parameter values were evaluated using ratios to enable fair comparison in different sets of experiments. Using a different set of experiments than the one used to evaluate the 50% trainable parameters ratio, Figure 6 demonstrates that the 75% ratio achieved a 1.94% improvement over the 25% baseline.

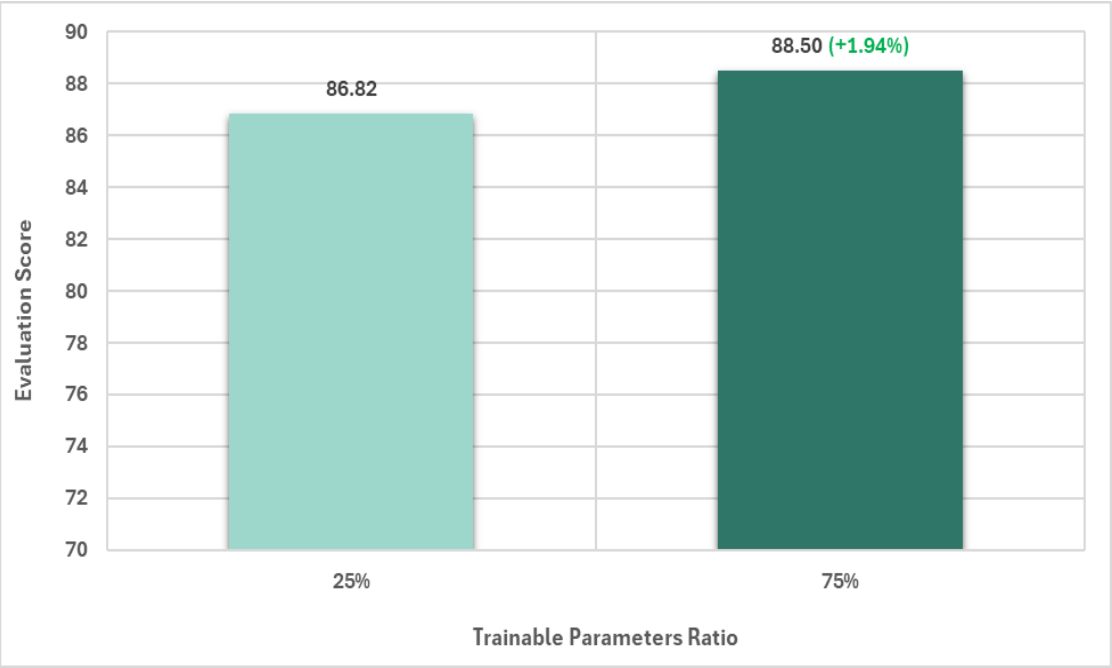

Figure 6. Comparison of Total Evaluation Scores between 25% and 75% Trainable Parameter Ratios, Using a Different Set of Experiments.

Trainable parameters ratio of 100% has an enhancement of 2.09% compared to 25% (Figure 7). Nevertheless, the performance gain of the 100% ratio over the 75% ratio was minimal, accompanied by a 12.45% increase in training duration. Given this marginal improvement in performance coupled with a significant increase in training time, we concluded that the 75% trainable parameter ratio offered a more favorable balance between performance and computational efficiency. The 75% ratio demonstrated a substantial improvement over the 25% and 50% ratios, while avoiding the excessive computational cost associated with the 100% ratio.

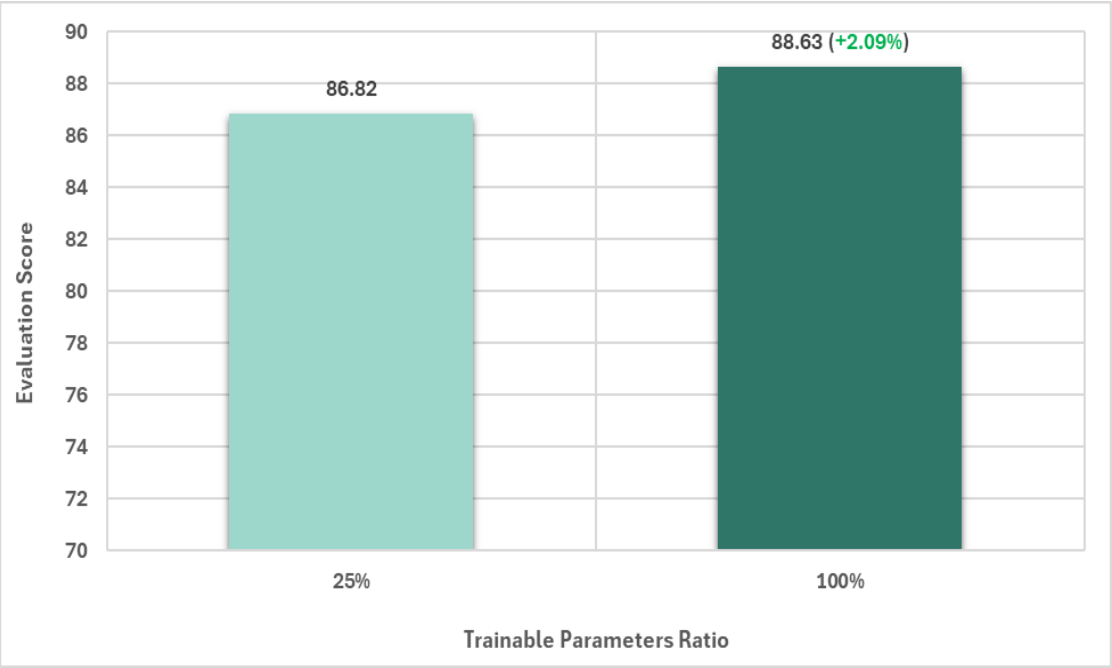

Figure 7. Comparison of Total Evaluation Scores between 25% and 100% Trainable Parameter Ratios.

Finally, we examined the influence of precision, a parameter that determines the numerical representation and accuracy of floating-point values during model training, on training stability and resource consumption. This involved comparing 32-bit true precision (FP32), fp16 precision (Floating Point 16-bit), and bf16-mixed precision (Brain Floating Point 16-bit).

The choice of FP32, or standard single-precision floating-point arithmetic, offers high numerical accuracy, ensuring precise calculations throughout the training process. However, this precision comes at the cost of substantial memory usage and increased computational requirements, leading to longer training times.

While fp16 also aims to reduce memory footprint and accelerate training, offering a significant speedup compared to FP32, it suffers from a significantly reduced dynamic range. This narrower range can lead to underflow issues, where very small numbers are rounded to zero.

To bridge the gap between the high accuracy of FP32 and the computational efficiency of fp16, bf16-mixed precision (Brain Floating Point 16-bit) was used. This mixed-precision technique, designed by Google researchers, utilizes 16-bit floating-point numbers for certain operations, while maintaining a dynamic range comparable to FP32. By selectively applying 16-bit precision, bf16-mixed significantly reduces memory usage and accelerates training, offering a balance between numerical accuracy and computational efficiency. We have examined both FP32 and bf16-mixed, and results showed that performance, measured by the total evaluation score based on equation 4.1, was marginally degraded by 0.15%, as demonstrated by Figure 8. However, model training using FB32 precision required about double the time compared to training using bf16-mixed.

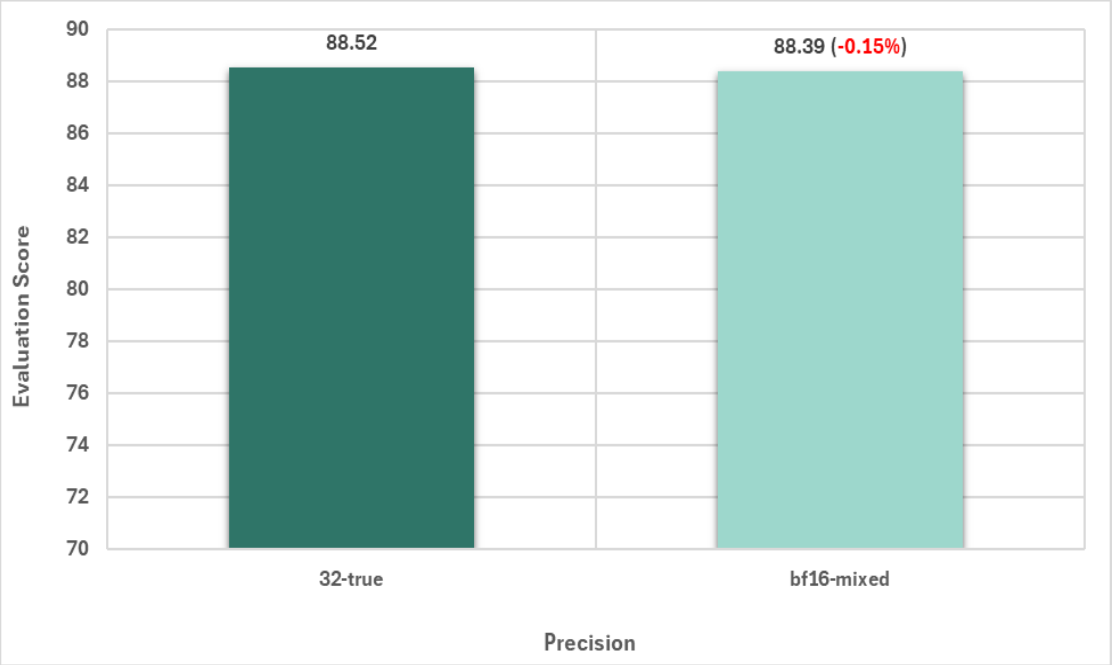

Figure 8. Impact of the Precision parameter on Total Evaluation Score. The chart compares 32-true and bf16-mixed precision

In conclusion, we determined that the ByT5-Base model, a 75% trainable parameter ratio, and bf16-mixed precision offered the best balance of performance and efficiency. The 75% ratio outperformed lower ratios without the 100% ratio's cost, and bf16-mixed

closely matched FP32's performance with reduced training time. These optimized values were adopted for our final model, ensuring robust performance and efficient resource use.

5.2 Exploring Best Hyperparameters

The optimization of hyperparameters is paramount to achieving enhanced performance and efficiency in the ByT5 model. This section details the hyperparameter tuning process, emphasizing the critical parameters that significantly influence model training and generalization. A systematic experimental approach was employed to identify the optimal configuration, balancing training stability, convergence velocity, and model accuracy.

To fully explore the impact of individual hyperparameters, we sometimes adopted a strategy of varying one parameter at a time. This allowed us to isolate the effects of each adjustment and gain a clear understanding of its influence on model performance. However, recognizing the potential for interactions between hyperparameters and the need for efficient exploration of the parameter space, we also conducted experiments where multiple parameters were adjusted simultaneously. This approach enabled us to explore a broader range of configurations and potentially uncover optimal combinations that would have been missed by single-parameter adjustments.

The hyperparameters subjected to adjustment included the maximum input sequence length and maximum output sequence length, which define the token processing and generation capacity of the model, respectively which affect the contextual information that is necessary for our model to capture. The maximum epochs number was refined to mitigate underfitting and overfitting phenomena. Furthermore, the learning rate was optimized to ensure smooth convergence, while the batch size was tuned to achieve equilibrium between memory utilization and training stability.

The subsequent subsections will elucidate the tuning methodology, present the experimental findings, and analyze the impact of these hyperparameters on the overall performance of the ByT5 model.

5.2.1 Selection of Maximum Epochs Number

The determination of an appropriate maximum number of epochs is crucial for effective model training, as it directly impacts both convergence and the prevention of overfitting. While dedicated trials focusing solely on the maximum epoch number were not conducted, our initial experiments provided valuable insights that guided our decision-making process.

The first experiment, utilizing a JODA dataset, was executed with a maximum of 10 epochs. Early stopping was not implemented to get Insights about the suitable maximum epochs number and number of patience epochs for the early stopping. As depicted in Figure 9, the validation loss exhibited a plateau and then started to increase after the fourth epoch, suggesting that further training beyond this point would yield overfitting.

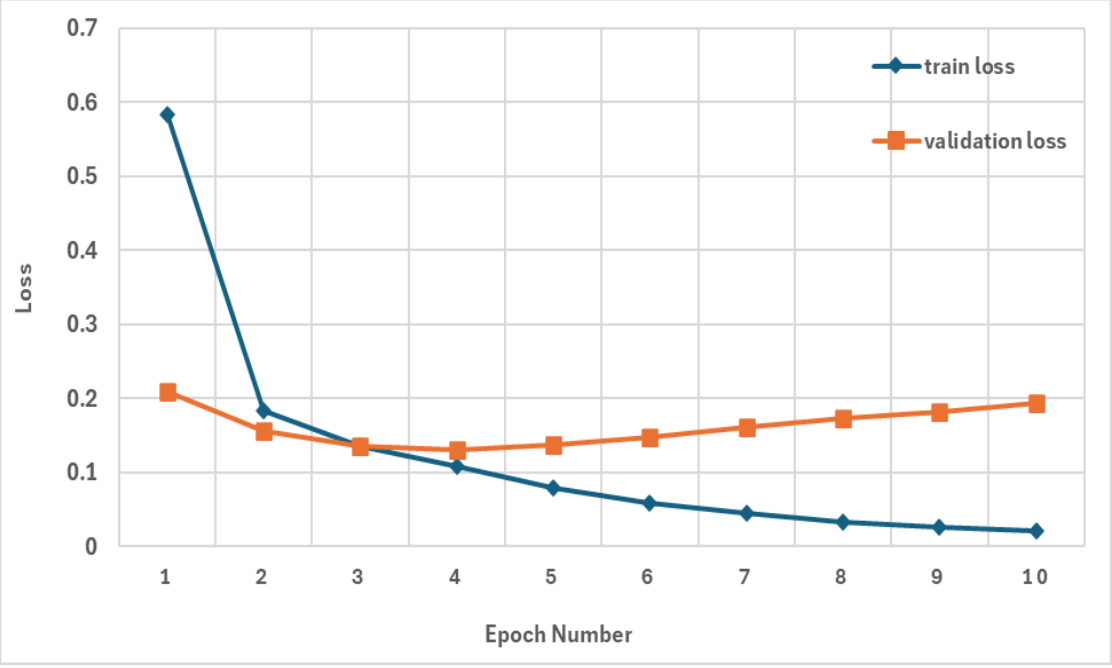

Figure 9. Training and Validation Loss Curves Over 10 Epochs (Early Experiments with JODA Dataset).

Consequently, we opted to reduce the maximum epoch number to 5 for subsequent experiments, even as the dataset size expanded significantly, after incorporating Clean-50 and Clean-400 datasets. We also employed early stopping callback with patience of two epochs in most of the experiments. Table 10 shows statistics about the number of epochs needed for all our experiments that were needed to record the best performance.

Table 10. Summary Statistics of the Number of Epochs at Which Best Performance was Achieved in Training Experiments.

Median	Minimum	Maximum	Range	Standard Deviation	Mode
2	1	5	4	1.19	2

This consistent early stopping across diverse dataset sizes and model configurations validated our choice, indicating that a maximum of 5 epochs provided sufficient training for convergence without unnecessarily prolonging the training process. In summary, the decision to limit training to a maximum of 5 epochs, coupled with early stopping, proved to be an effective strategy for balancing model performance and training efficiency. This approach ensured that our models achieved optimal results without incurring the risk of overfitting, while also minimizing computational resource consumption.

5.2.2 Exploring Effect of Learning Rate

The learning rate is a critical hyperparameter that significantly influences the training dynamics and convergence of the ByT5 model. To investigate its impact on performance, we conducted a series of experiments, systematically varying the maximum learning rate while maintaining consistent settings for all other parameters.

Our initial learning rate of 0.003 was based on results from Al-Rfooh, *et al*. (2023) work, that demonstrated its effectiveness in similar tasks. However, to ensure optimal performance for our specific dataset and objectives, we explored a broader range of

values. Since we noticed that, in our initial trials, the model becomes overfit after the first two or three epochs, we decided to try other lower maximum learning rates. Particularly, in addition to the learning rate of 0.003, we trialed learning rates of 0.001, and 0.0001.

We employed the AdamW optimizer, which incorporates weight decay, a technique that helps in preventing overfitting by penalizing large weights. Additionally, we implemented a warm-up phase, setting it to 10% of the total training steps. During this phase, the learning rate increased linearly, reaching its maximum value at the end of the warm-up period. Following the warm-up, the learning rate decayed proportionally to the inverse square root of the step number, resulting in a gradual decrease throughout the remaining training epochs, preventing overshooting and ensuring smooth convergence for the model.

Figure 10 and Figure 11 illustrate the sensitivity of model performance to the learning rate. Experiments with learning rates of 0.003 and 0.001 recorded their best performance in the second epoch while it was in the third epoch when experimenting with learning rates of 0.0001.

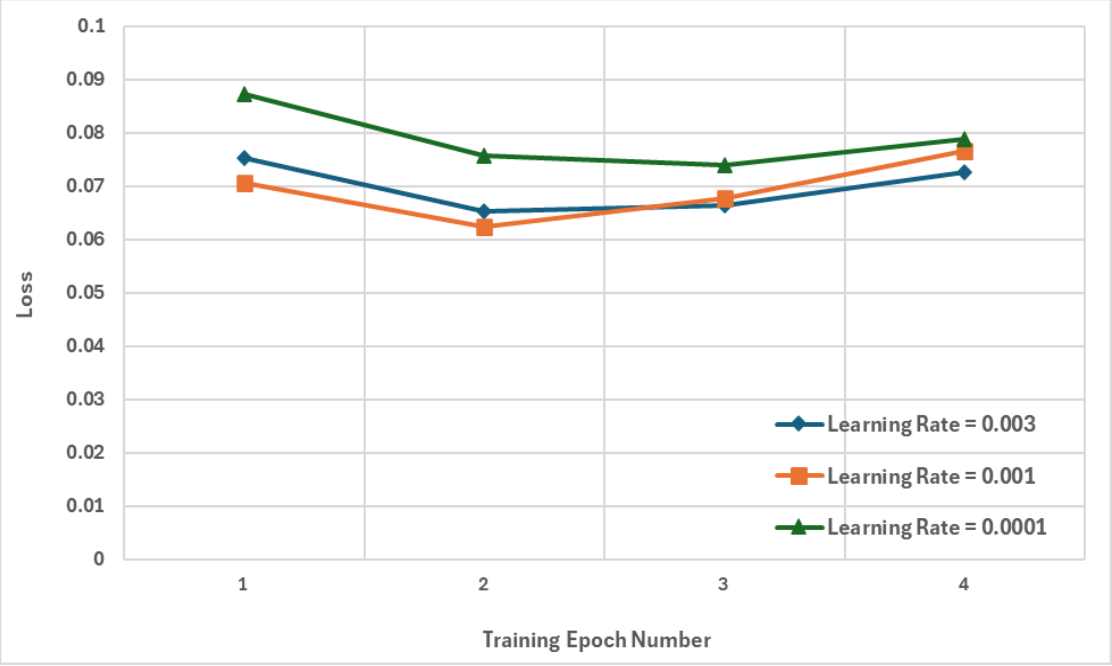

Figure 10. Loss Curve of Fine-Tuning ByT5 for Varying Learning Rates.

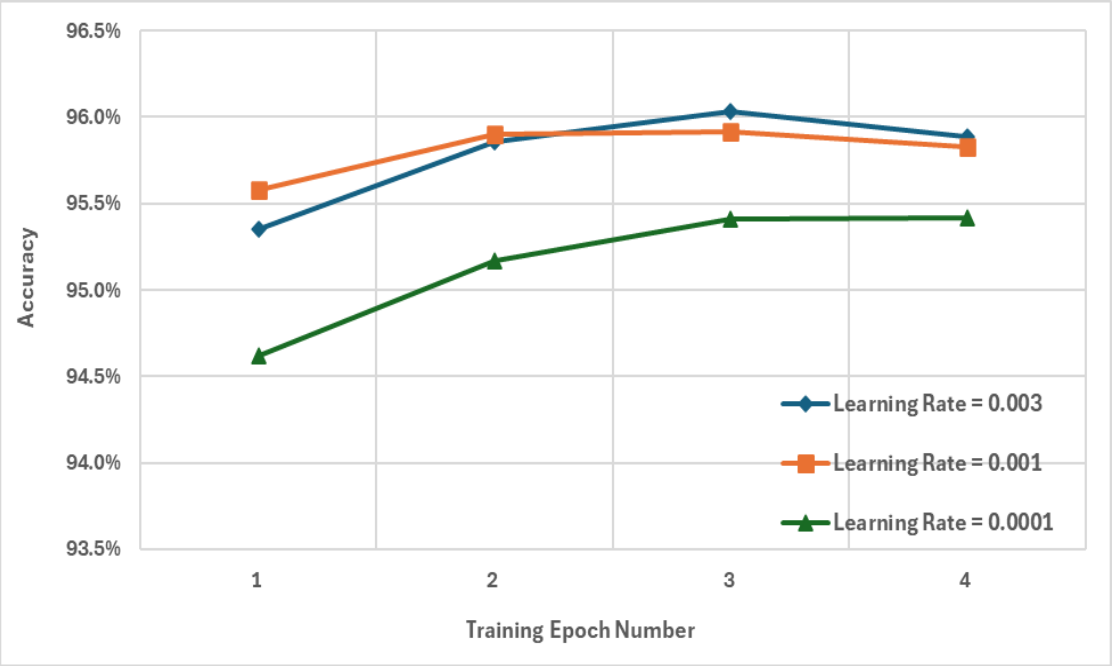

Figure 11. Accuracy Curve of Fine-Tuning ByT5 for Varying Learning Rates.

From those curves, we can notice that models with learning rates of 0.003 and 0.001 converge faster than that with 0.0001 and achieve better loss and accuracy values. The models were also evaluated using CER, BLEU, DER, and WER and then the total evaluation score was calculated based on equation (4.1). Table 11 presents evaluation results for the different learning rates. A learning rate of 0.001 yields the highest score (88.495), outperforming the other learning rates. Therefore, based on the evaluation scores and the observed convergence patterns, we concluded that a learning rate of 0.001 was the most effective for our model.

Table 11. Evaluation Metrics of fine-tuning ByT5 model for Varying Learning Rates.

Learning Rate	CER (%)	BLEU (%)	DER (%)	WER (%)	Evaluation Score
0.003	5.91	76.01	**2.53**	14.92	88.16
0.001	**5.74**	**76.73**	2.57	**14.44**	**88.50**
0.0001	6.39	74.50	2.81	16.19	87.28

5.2.3 Exploring Effect of Maximum Sequence Length

The maximum sequence length, which determines the number of tokens processed by the model at once, is a crucial parameter that can significantly impact both performance and computational efficiency. To investigate this effect, we conducted a series of experiments using the combined JODA and Clean-50 datasets, varying the input and output maximum sequence lengths. Building upon the findings of Al-Rfooh, *et al*. (2023), which demonstrated that longer sequence lengths enhance DER and WER metrics in diacritization tasks, our primary objective extended beyond pinpointing the absolute optimal sequence length. Instead, we aimed to investigate how varying sequence lengths affect the performance of our multi-task model, particularly in terms of CER and BLEU improvements, while also examining overall model performance and computational burden. Specifically, we investigated whether longer sequence lengths enabled the model to capture more contextual information. Additionally, we examined the associated computational costs.

We experimented with three distinct input/output sequence length configurations [(208, 256), (366, 512), and (700, 1,024)]. The validation loss and accuracy curves, for each configuration are presented in Figure 12 and Figure 13, respectively.

As depicted in Figure 12, the validation loss curves demonstrate a clear trend: longer sequence lengths result in lower validation loss. The (700, 1,024) configuration consistently achieved the lowest loss, indicating superior generalization. Conversely, the (208, 256) configuration exhibited the highest loss, suggesting limited contextual understanding. These results suggest that increased sequence length enables the model to capture more contextual information, leading to improved performance.

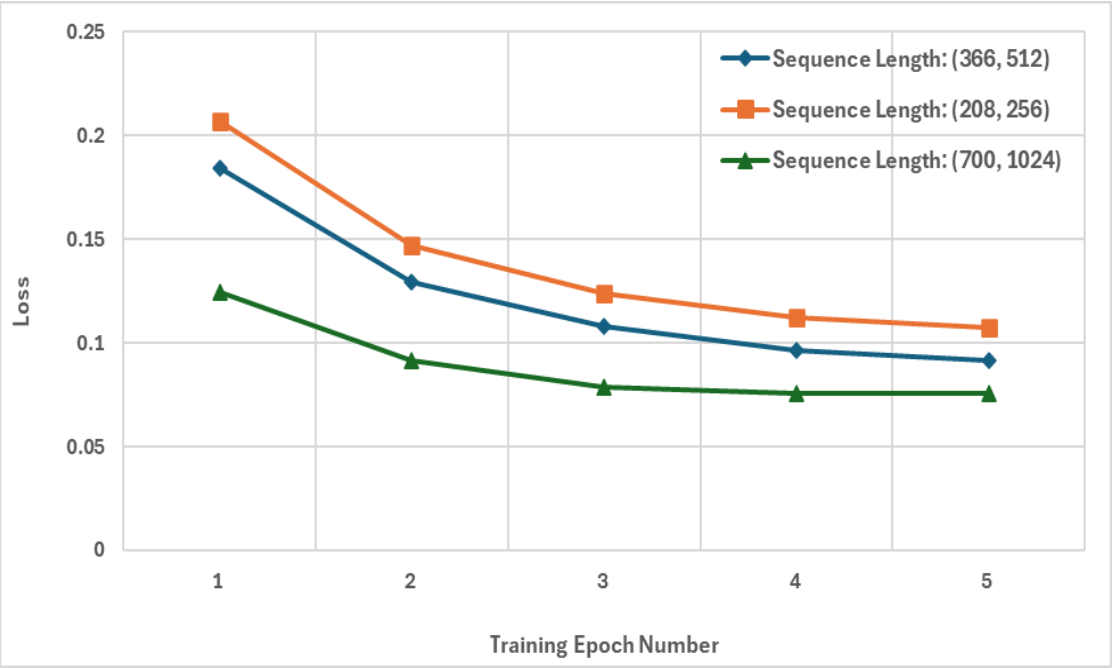

Figure 12. Validation Loss Curves for Different Input/Output Sequence Length Configurations Over Five Epochs.

As depicted in Figure 13, the accuracy curves reinforce the trend observed in the loss curves: longer sequence lengths result in higher accuracy. The (700, 1,024) configuration consistently achieved the highest accuracy, indicating superior generalization. Conversely, the (208,256) configuration exhibited the lowest accuracy, suggesting limited contextual understanding. These results further demonstrate that increased sequence length enables the model to capture more contextual information, leading to improved performance.

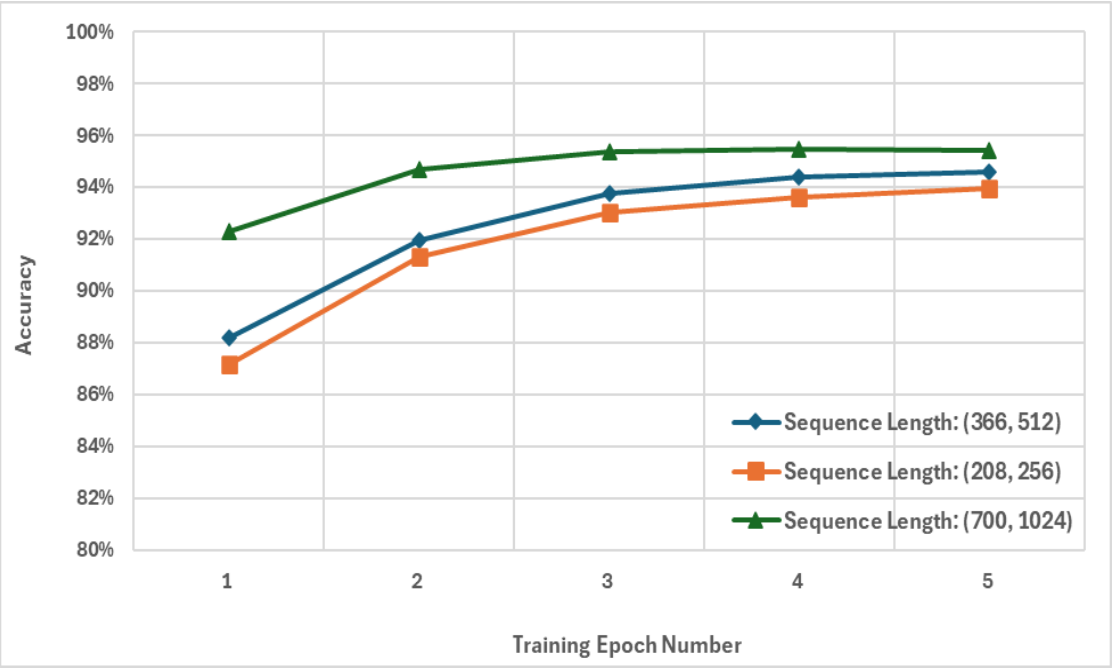

Figure 13. Accuracy Curves for Different Input/Output Sequence Length Configurations Over Five Epochs.

The impact of maximum sequence length on model performance was further quantified using evaluation metrics, as presented in Table 12. The results consistently aligned with the trends observed in the loss and accuracy curves. The longest sequence length configuration, (700, 1,024), achieved the highest score in each of the metrics, indicating superior performance. Specifically, it yielded the lowest CER (7.17%), DER (3.2%), and WER (17.36%), and the highest BLEU score (74.32%). Conversely, the shortest sequence length configuration, (208, 256), resulted in the lowest scores, with the highest error rates and lowest BLEU score. The medium sequence length configuration, (366, 512), achieved a moderate score, demonstrating performance between the other two configurations. These findings reinforce the conclusion that longer sequence lengths enable the model to capture more contextual information, leading to improved performance across all evaluation metrics

Table 12. Evaluation Metrics (CER, BLEU, DER, WER, and Total Score) for Different Input/Output Sequence Length Configurations. Longer sequence lengths correlate with better performance across all metrics.

Max Input Sequence Length (bytes)	Max Output Sequence Length (bytes)	CER (%)	BLEU (%)	DER (%)	WER (%)
208	256	10.85	64.90	4.00	21.63
366	512	7.69	70.20	3.88	19.49
700	**1,024**	**7.17**	**74.32**	**3.20**	**17.36**

Figure 14 emphasize the preceding results by comparing the total evaluation score for each input/output sequence length. However, this performance gain comes at a significant computational cost. As depicted in Figure 15, the training time needed for five epochs increased substantially with longer sequence lengths. The (700, 1,024) configuration required significantly more training time compared to the other configurations. This demonstrates the trade-off between performance and computational burden.

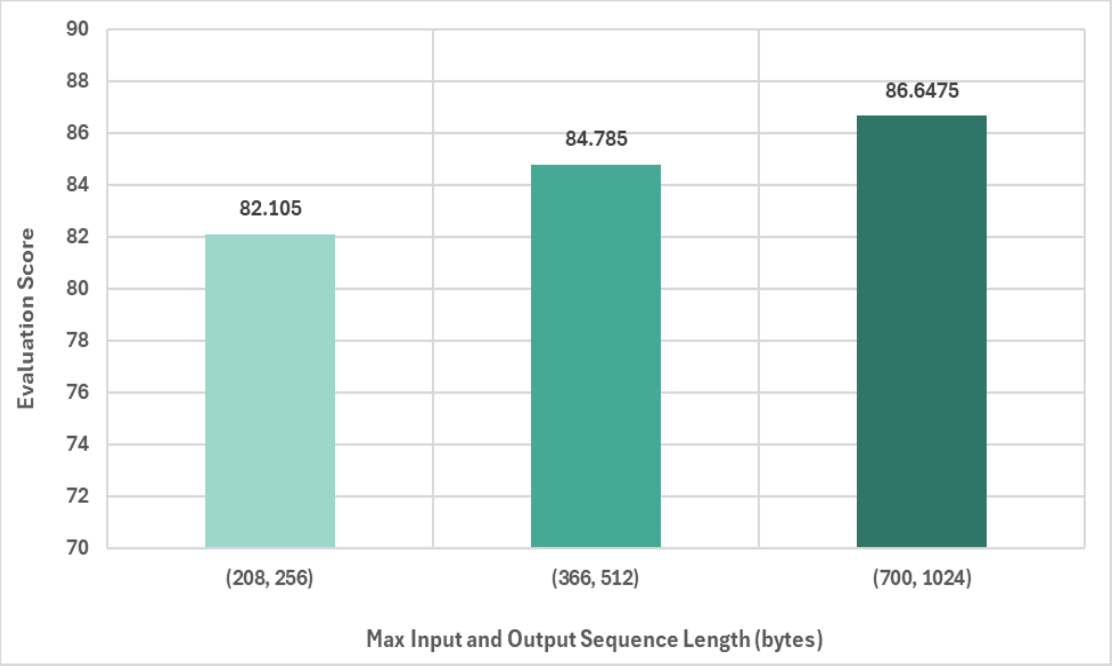

Figure 14. Evaluation Scores for different Sequence Length Configurations.

Based on these results, we opted for a maximum sequence length configuration that significantly improves performance despite the higher computational demand. The exact sequence length chosen varies depending on the specific dataset, batch size, number of samples, and trainable parameters ratio used in subsequent experiments. This adaptive approach allowed us to optimize model performance while maintaining reasonable training times.

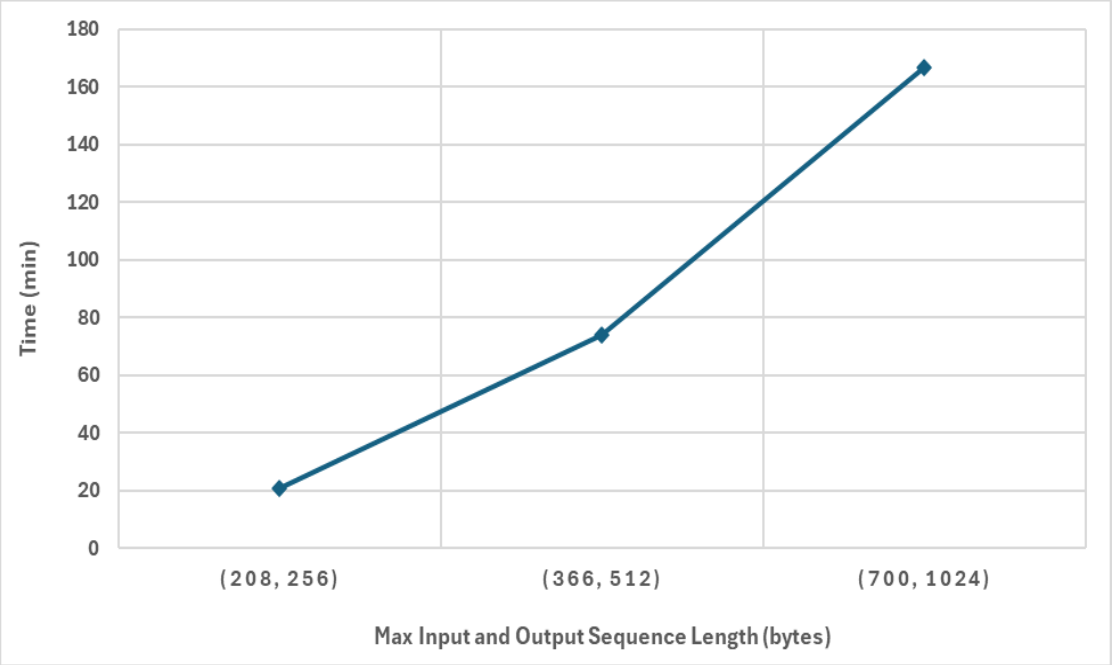

Figure 15. Training Time for Different Sequence Length Configurations.

5.2.4 Exploring Effect of Batch Size

The batch size, a crucial hyperparameter that determines the number of samples processed in each training iteration, can significantly influence model training dynamics and performance. To investigate its effect, we conducted a controlled experiment comparing two batch sizes: 128 and 256 (which was adopted by Al-Rfooh, *et al*., 2023). All other hyperparameters and architecture parameters were held constant between these two experiments, allowing us to isolate the impact of batch size.

Figure 16 illustrates the validation loss curves for both batch sizes over three epochs. As observed, the 128-batch size exhibits a lower loss trend while the 256-batch size shows a higher loss, with an increase after the second epoch for both batch sizes.

Figure 16. Validation Loss Curves for Batch Sizes 128 and 256 over Three Epochs.

Figure 17 depicts the validation accuracy curves for the same experiments. The 256-batch size achieves a slightly higher accuracy at epoch 2 compared to the 128-batch size. However, it also shows a decrease in accuracy at epoch 3, aligning with the divergence observed in the loss curve. The 128-batch size, on the other hand, shows a relatively more stable accuracy.

Figure 17. Validation Accuracy Curves for Batch Sizes 128 and 256 over Three Epochs.

The total evaluation scores for both batch sizes are presented in Figure 18. The 128-batch size achieves a slightly higher total score of 88.495 compared to the 256-batch size, which scores 88.385. This aligns with the observed trends in the loss and accuracy curves, where the 128-batch size demonstrates more stable performance.

Based on these results, we conclude that a batch size of 128 yields better overall performance in terms of both stability and evaluation score for this specific experimental setup. The 256-batch size, while showing a slight advantage in accuracy at epoch 2, exhibits signs of divergence at epoch 3, leading to a lower final score. Therefore, we adopted using a batch size of 128 for the subsequent experiments with this specific configuration.

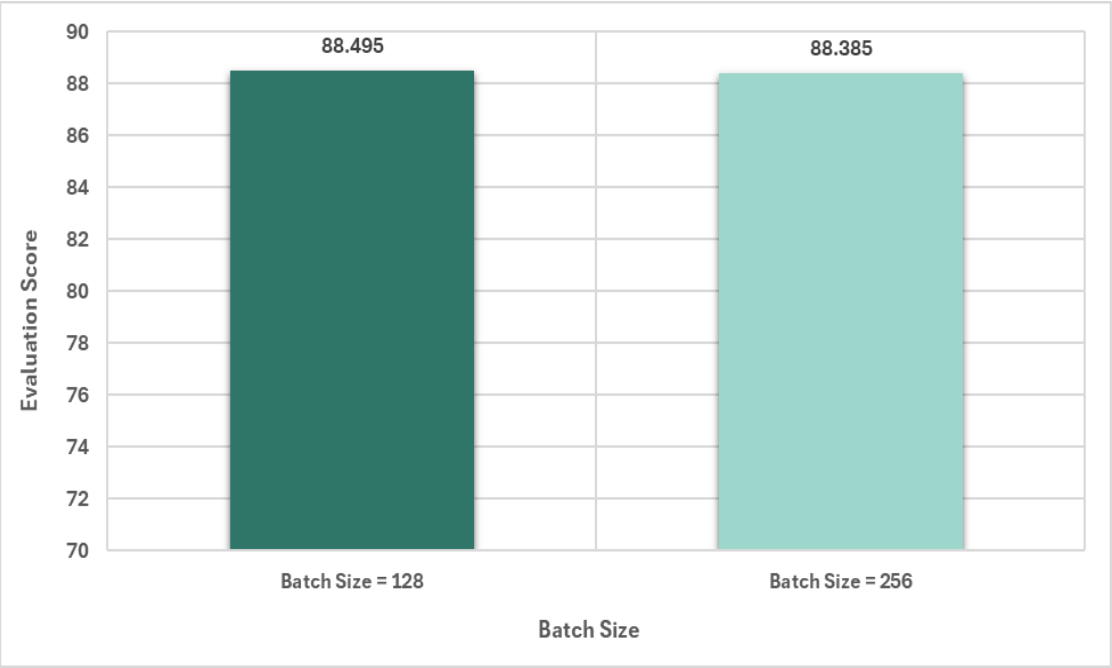

Figure 18. Total Evaluation Scores for Batch Sizes 128 and 256.

5.3 Dataset Impact on Model Performance

This section details a series of experiments conducted to evaluate how different dataset combinations affect our model's ability to correct errors and diacritize text from Jordanian dialect and MSA sources containing common mistakes. Given our objective to accurately process real-world text, we initially focused on the JODA dataset. However, initial experiments revealed suboptimal performance on the JODA test set, indicating the need for dataset enhancement. To address this, we explored several strategies, including the addition of other datasets, specifically Clean-50 and Clean-400 from the Tashkeela dataset. These datasets are characterized by a high diacritics ratio and rich contextual information, which are beneficial for our objective. We also tested the impact of a smaller dataset, REALMS, to assess its potential contribution. Throughout these experiments, we utilized various test sets, including the JODA-only test set and combinations of JODA with other datasets, to thoroughly evaluate model performance and generalization. This

section presents a systematic analysis of these experiments, highlighting the impact of different dataset combinations on model performance and ultimately leading to the selection of the most effective dataset strategy.

5.3.1 Initial Experiments with JODA Dataset

JODA dataset, which was described in Chapter 4, contains pure Jordanian dialect sentences, in addition to MSA that contain real world spelling mistakes with their corresponding corrected MSA version, making it an ideal starting point to fine-tune our ByT5 model. Initially, we have fine-tuned ByT5-Base model in two different configurations, varying the trainable parameters ratio. The results, shown in Table 13, were suboptimal. Consequently, we decided to add extra dataset before further fine-tuning, as the JODA dataset sample size is under the model's capacity required for MTL. In addition, it contains machine-generated diacritics which impose a certain percentage of error. Based on this, we hypothesized that introducing an Additional dataset with a high accuracy diacritization ratio would noticeably enhance the performance, allowing us then to tweak some soft parameters.

Table 13. Performance Results of Initial Experiments that Utilized JODA Dataset Only.

Experiment Number	Trainable Parameters Ratio %	CER (%)	BLEU (%)	DER (%)	WER (%)	Score (%)
1	25%	15.99	49.09	4.05	33.06	73.9975
2	50%	15.5	49.98	4.09	32.26	74.5325

5.3.2 Combining JODA and Clean-50 Datasets

As previously mentioned, to address the suboptimal results of models that are fine-tuned using JODA dataset only, we sought to enrich the training data with an additional dataset that could provide complementary information and enhance the model's generalization capabilities. The Clean-50 dataset was chosen for this purpose. However, to ensure that Clean-50 better aligned with our objective of correcting real-world spelling errors, we applied synthetic noise injection to simulate common spelling mistakes, as described in Chapter 4.

The rationale for selecting Clean-50 was twofold. First, it contains MSA texts, which aligns with a significant portion of the JODA dataset that also includes MSA with spelling mistakes. Second, Clean-50 is characterized by a high diacritics ratio and rich contextual information, both of which are crucial for improving the model's ability to accurately diacritize and correct text.

After integrating Clean-50 into the training set, we re-ran Experiment 1, which had a lower trainable parameter ratio and required less training time, as a proof of concept to verify whether adding Clean-50 could improve performance. We observed clear and consistent enhancement across all evaluation metrics, as illustrated in Figure 19, which compares the performance of training on JODA-only versus JODA+Clean-50, both of the models were evaluated using same test set, JODA test set. Specifically, we observed a significant reduction in DER by 10.86%, aligning with the expectation that Clean-50, a high-quality diacritized dataset, would substantially improve diacritization accuracy. Additionally, CER decreased by 8.19%, indicating improved character-level corrections, and WER saw a 5.05% reduction, showcasing better word-level accuracy. The BLEU score also increased by 2.85%, suggesting an improvement in the overall similarity of the model's output to the reference translations.

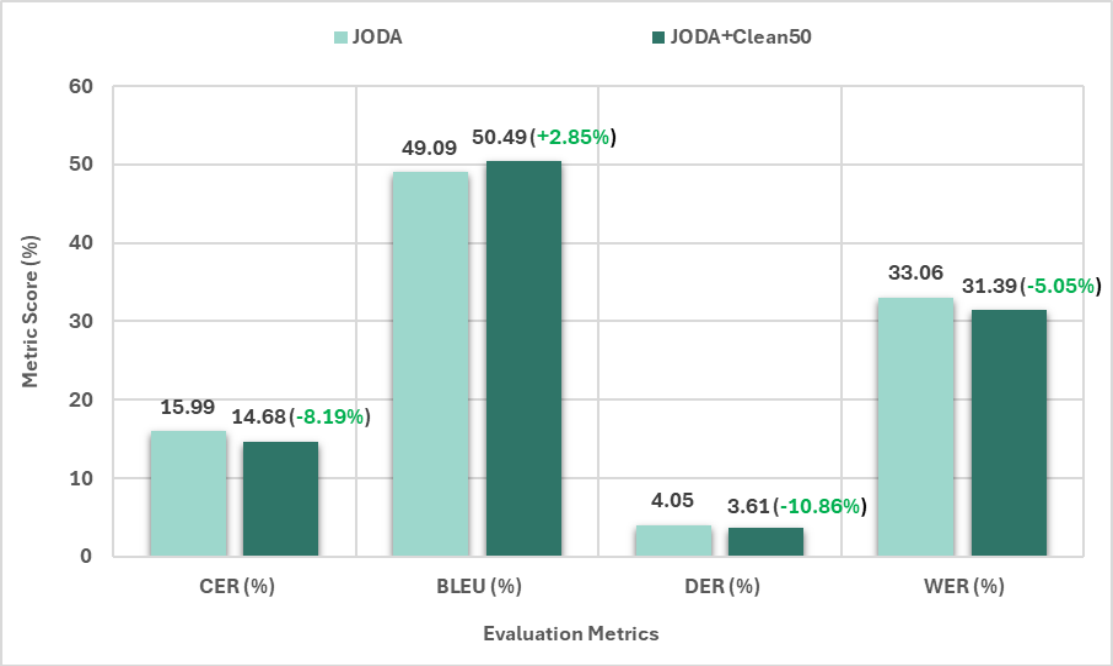

Figure 19. Performance Enhancements After Adding Clean-50 to JODA Dataset (Evaluated using JODA Test Set).

The results confirm that the additional dataset provided more useful learning signals, leading to improved performance. This aligns with our expectation that introducing more MSA data with controlled errors would enhance the model's ability to generalize. Based on these findings, we hypothesized that a larger high-quality dataset could enhance the model's performance, so we meant to use a larger dataset particularly, Clean-400. However, the substantial size of Clean-400 poses significant training time challenges, hindering the iterative experimentation needed for hyperparameter and architectural optimization. Therefore, to balance dataset richness with computational efficiency for broader experimentation, we strategically focused on the combined JODA+Clean-50 dataset for much of our work, evaluating our models on a merged test set from JODA and Clean-50 to ensure robust assessment across dialectal and standard MSA text, which help in exploring different parameters configurations.

5.3.3 Scaling Up: Adding Clean-400 Dataset

Following the successful integration of the Clean-50 dataset, which resulted in noticeable performance improvements, we explored the potential benefits of scaling up our training data with the Clean-400 dataset. Clean-400, like Clean-50, is a high-quality subset of the Tashkeela dataset, but it offers a significantly larger volume of texts with accurate diacritization and rich contextual information. We hypothesized that leveraging this larger dataset could further enhance the model's generalization capabilities and overall performance.

Despite the primary focus on JODA+Clean-50, since it requires less computational resources and enables us to explore a wider range of hyperparameters and architectural variations, we did conduct a comparative experiment with the JODA+Clean-400 dataset to gauge its potential impact. The results of this comparison are presented in Figure 20, which outlines the performance metrics for both dataset combinations.

The comparative analysis between the JODA+Clean-50 and JODA+Clean-400 datasets reveals subtle yet notable differences in performance. While the JODA+Clean-400 configuration demonstrates marginal improvements in CER and BLEU scores, the most significant enhancement is observed in DER, which exhibits a 6.31% reduction. This indicates that the larger Clean-400 dataset contributes to improved diacritization accuracy, likely due to its richer and more extensive training data. This suggests that the Clean-400 dataset is particularly effective in improving the model's ability to accurately diacritize Arabic text. WER, which is affected by errors in all characters including letters and diacritics, also demonstrates a measurable decrease, suggesting an improvement in the overall word-level accuracy.

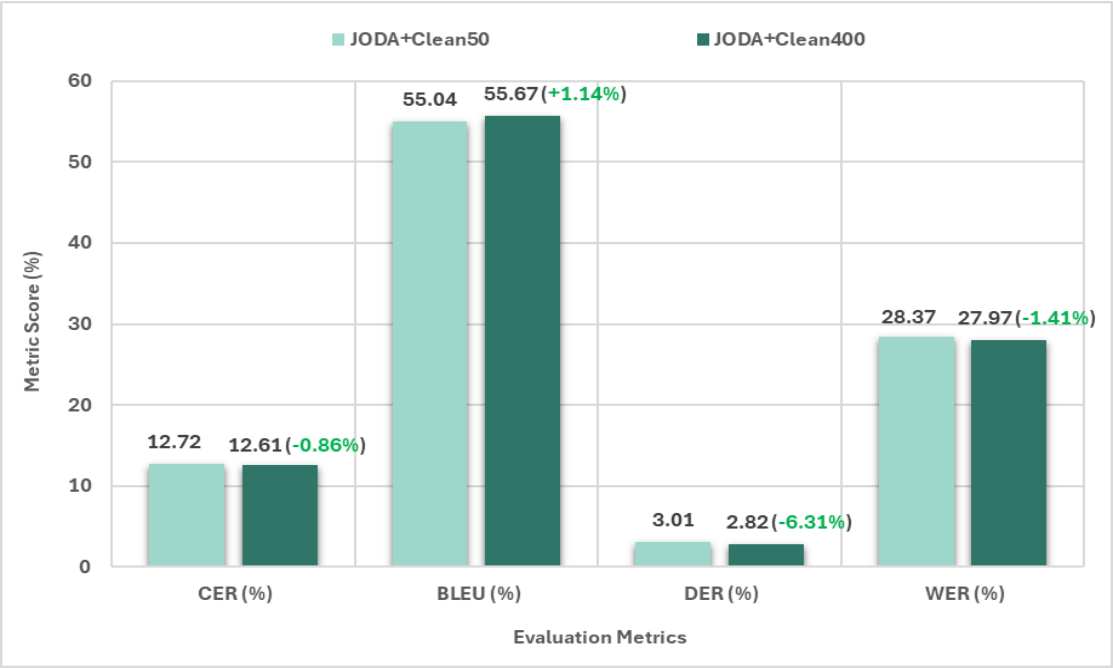

Figure 20. Evaluation Metrics Comparison Between JODA+Clean-50 and JODA+Clean-400 Datasets (Evaluated using JODA Test Set), Showing Percentage Improvements/Reductions in CER, BLEU, DER, and WER.

To further illustrate the impact of the Clean-400 dataset, we analyzed the validation loss and accuracy curves (Figure 21 and Figure 22). The validation loss curve for the JODA+Clean-400 model showed a consistent decrease across all three epochs, with loss values approximately 20% lower than those observed in the JODA+Clean-50 model. Importantly, the JODA+Clean-50 model began to exhibit signs of overfitting after the second epoch, as indicated by an increase in loss. In contrast, the JODA+Clean-400 model maintained a decreasing loss trend until the third epoch.

Similarly, the validation accuracy curves revealed that the JODA+Clean-400 model achieved a consistently higher accuracy, approximately 0.34% across all epochs, compared to the JODA+Clean-50 model. Furthermore, the JODA+Clean-50 model exhibited a decrease in accuracy after the second epoch, indicating potential overfitting, while the JODA+Clean-400 model continued to show an increase in accuracy throughout the three epochs.

These findings support the decision to adopt the JODA+Clean-400 dataset for our best model, as it demonstrates superior performance and improved training dynamics compared to the JODA+Clean-50 combination.

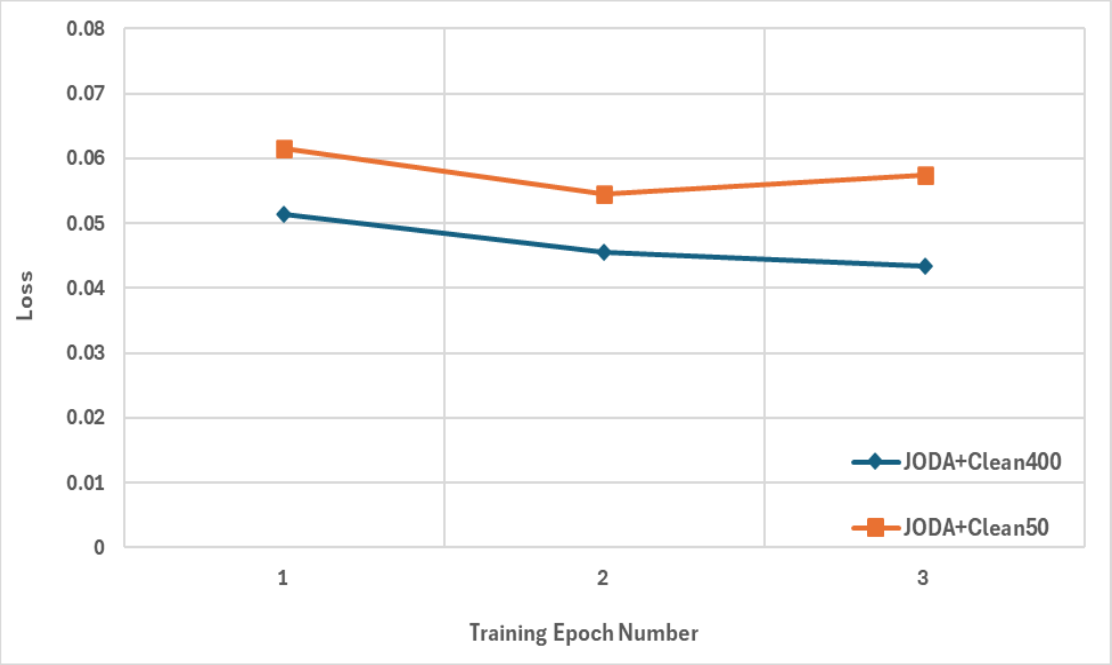

Figure 21. Validation Loss Curves for JODA+Clean-50 and JODA+Clean-400 Datasets over Three Epochs.

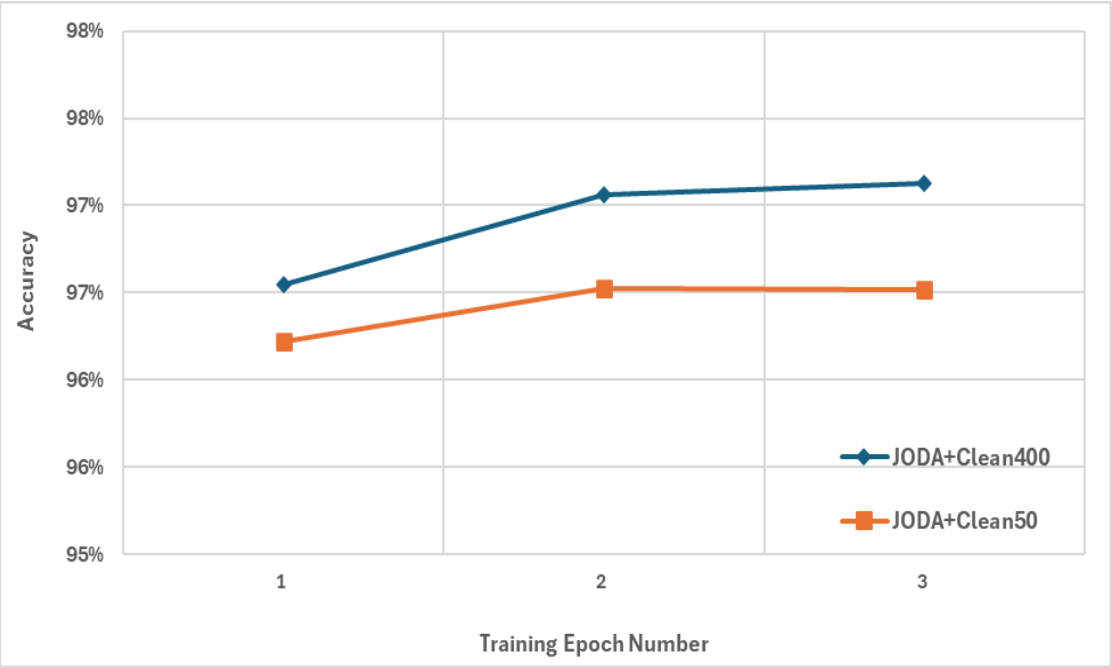

Figure 22. Validation Accuracy Curves for JODA+Clean-50 and JODA+Clean-400 Datasets over Three Epochs.

5.3.4 Investigating REALMS Dataset Integration

ByT5 model that was fine-tuned using JODA+Clean-400 dataset achieved best performance on both JODA and JODA+Clean-400 test sets. However, results on the JODA+Clean-400 test sets were significantly higher than that of JODA only test set. The reason behind that is the imbalance between Jordania dialectal texts and other MSA and CA texts. Also, 98.85% of the Tashkeela dataset samples, and indeeded Clean-400, are formed in CA. We tried to reduce this gap by doubling the JODA dataset; by using each sequence twice, once for each task. Also, we added REALMS dataset (Hasan and Abandah, 2023), which consists of 4123 sequences that contain real spelling mistakes collected from different social media platforms. The REALMS dataset was diacritized using same approach that was adopted in diacritizing JODA dataset. Using close parameter settings, compared to the model fine-tuned using JODA+Clean-400, ByT5-

Base model was fine-tuned using JODA+REALMS+Clean-400 dataset. The results are presented in Figure 23.

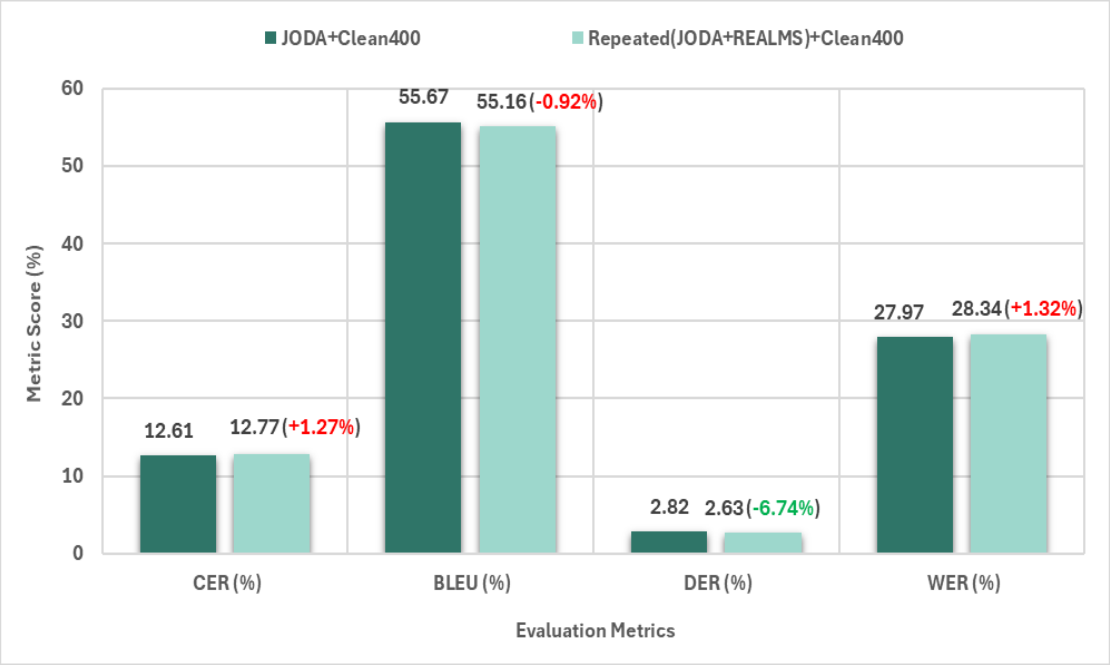

Figure 23. Evaluation Metrics Comparison Between Repeated (JODA+REALMS)+Clean-400 and JODA+Clean-400 Datasets (Evaluated using JODA Test Set), Showing Percentage Enhancements/Degradations in CER, BLEU, DER, and WER.

The performance evaluation, conducted using the JODA test set, revealed a marginal advantage for the JODA+Clean-400 dataset. While the inclusion of Repeated (JODA+REALMS) resulted in a slightly improved DER, there was a corresponding degradation in other key metrics, including CER, BLEU, and WER. Consequently, the model fine-tuned on the Repeated (JODA+REALMS)+Clean-400 combination achieved a lower overall score of 77.85 compared to the 78.06 obtained by the JODA+Clean-400 model. Furthermore, this configuration required significantly more computational resources. Given the negligible performance gains and the increased computational cost, we opted to retain the JODA+Clean-400 model as our optimal choice.

5.3.5 Final Dataset Selection and Results

Figure 24 presents a comparative overview of the total scores achieved by our model across various dataset configurations. As observed, training solely on the JODA dataset resulted in the lowest performance, highlighting the necessity of incorporating additional, high-quality data. The addition of the Clean-50 dataset significantly improved the model's performance, demonstrating the positive impact of enriching the training data with accurate diacritization and contextual information.

However, the JODA+Clean-400 configuration yielded the highest total score, confirming the benefits of utilizing a larger, high-quality dataset. This result underscores the importance of dataset size and quality in enhancing model performance.

The incorporation of the Repeated (JODA+REALMS) dataset alongside Clean-400, while intended to mitigate potential CA bias and balance the training data, did not result in an overall improvement. Instead, it led to a slight decrease in performance and increased computational costs.

Based on these findings, we conclude that the JODA+Clean-400 dataset provides the optimal balance between performance and computational efficiency for our model. The larger size and high quality of the Clean-400 dataset significantly contributes to improved diacritization accuracy and overall performance. Therefore, we recommend the JODA+Clean-400 configuration for further model development and deployment.

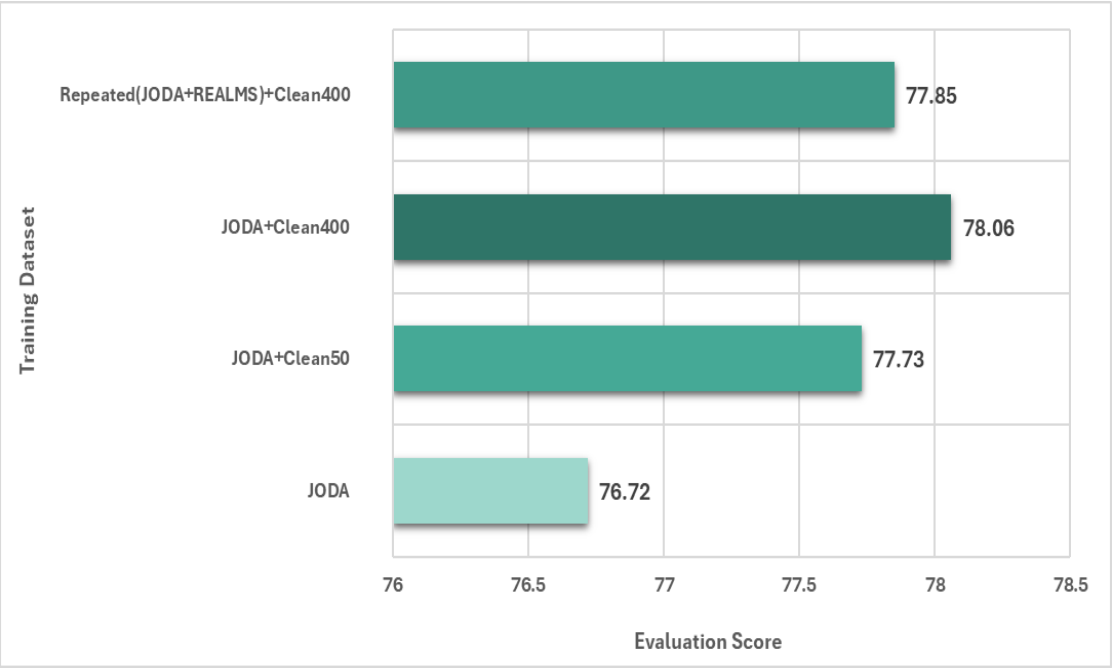

Figure 24. Total Score Comparison Across Different Dataset Configurations.

5.4 Final Model and Evaluation

This section consolidates the findings of our experimental work, focusing on the selection and evaluation of the final model. We detail the optimal configuration, including architecture, hyperparameters, and dataset, that yielded the best performance in our study. We then present a comprehensive analysis of the model's performance. Recognizing the inherent limitations of automated evaluation in capturing the nuances of dialectal Arabic translation, we concluded with a discussion of a manual evaluation conducted on a random subset of the JODA test set, providing a more human-centered assessment of the model's capabilities.

5.4.1 Model Configuration and Dataset

This subsection presents the configuration of the final model selected for our task. The model is based on the ByT5-Base architecture, fine-tuned using the JODA+Clean-

400 dataset. This dataset combination was chosen due to its demonstrated superiority in the model's performance, as established in Section 5.3.

The model was fine-tuned with the hyperparameters and architectural parameters in Table 14. The ratio of 75% trainable parameters, 0.001 learning rate, and 128 batch size yielded in best performance based on our experimental work detailed in Section 5.1 And 5.2. Also, the number of maximum epochs was chosen based on our mentioned findings and it was well founded as the model started to have signs of overfitting after the third epoch. The precision bf16-mixed was used as it offers very close accuracy to that presented by fp32-true with much less computational burden. For the maximum input and output sequences lengths, and as we concluded, the longer the sequence is, the more information the model can capture, but on the expense of computational burden. However, it was necessary to compromise between the computational burden and performance. Consequently, to manage computational resources and training time, the dataset's maximum sequence lengths were reduced from 1024 bytes to the 95th percentile of the lengths distribution, resulting in input and output limits of 634 and 966 bytes, respectively.

Table 14. Hyperparameters and Architectural Parameters Adopted in our Final Model.

Parameter	Value
Trainable Parameters Ratio	75%
Max Input Sequence Length (bytes)	634
Max Output Sequence Length (bytes)	966
Max Number of Epochs	5
Drop out Ratio	0.3
Optimizer	AdamW
Learning Rate	0.001
Precision	bf16-mixed
Batch Size	128

5.4.2 Model Performance and Evaluation

This subsection presents a comprehensive evaluation of the model's performance, encompassing both training dynamics and test set results. To assess the model's learning progress, we examined the training and validation loss and accuracy curves across epochs. Figure 25 illustrates the training and validation loss, while Figure 26 depicts the corresponding accuracy trends.

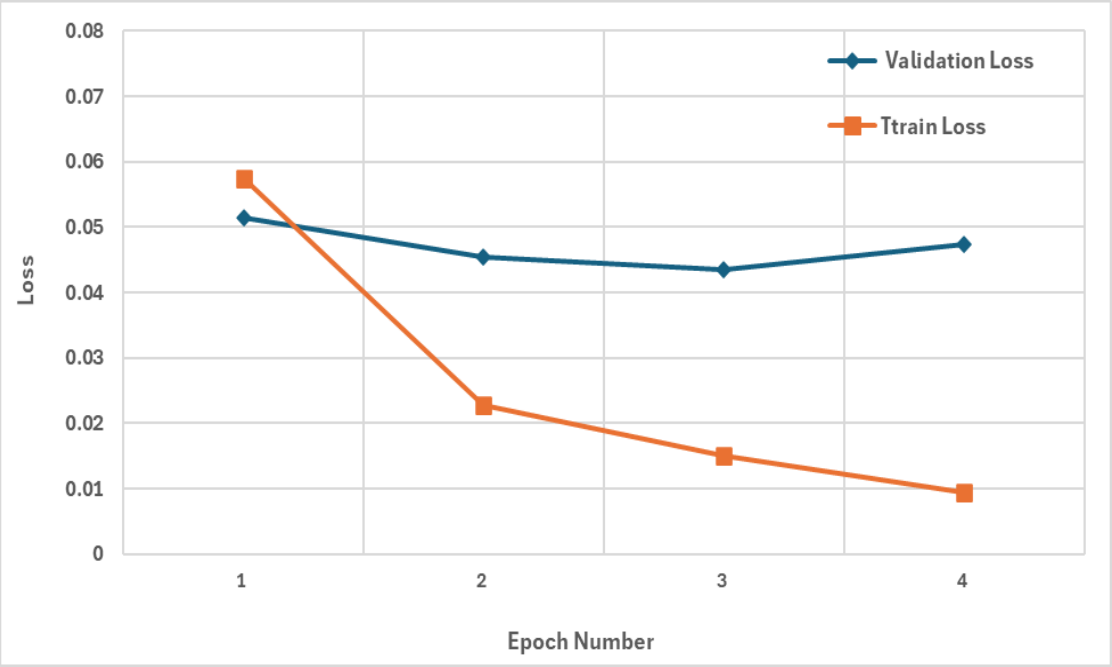

Figure 25. Training and Validation Loss Curves of the Final Model Across the Training Epochs.

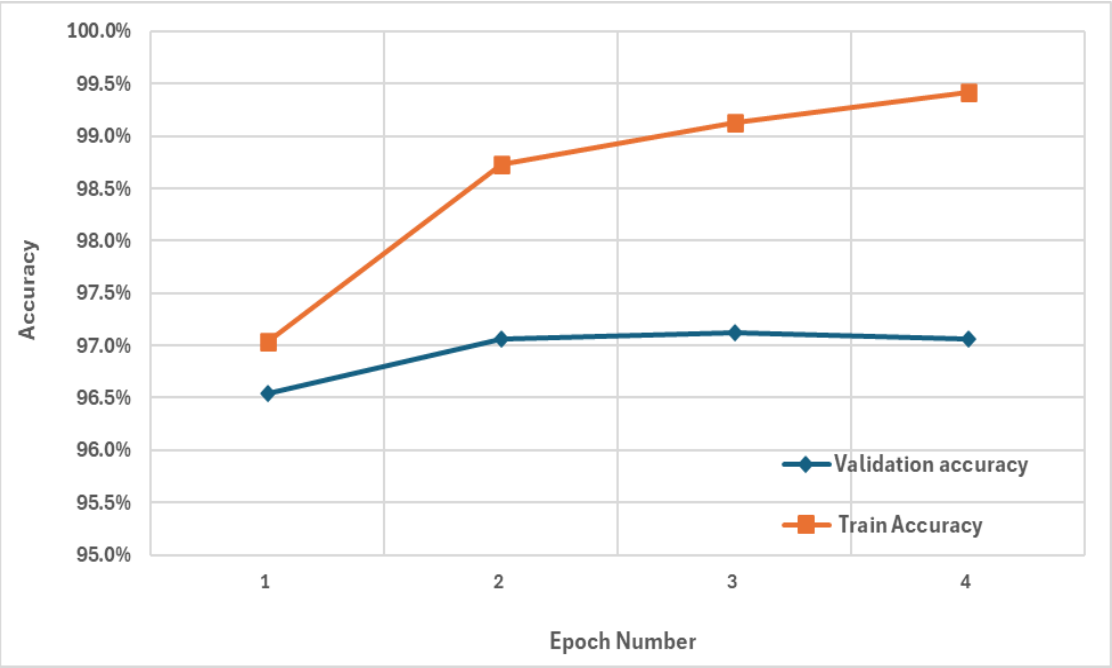

Figure 26. Training and Validation Accuracy Curves of the Final Model Across the Training Epochs.

The training loss exhibited a consistent downward trend throughout the training process, indicating effective learning. However, the validation loss, while initially decreasing, reached a minimum of 0.0434 and subsequently started to increase after the third epoch. This divergence between training and validation loss suggests the onset of overfitting, where the model begins to memorize the training data rather than generalize to unseen data.

The model required 4 epochs (14,155 steps) to complete training, triggered by early stopping. Notably, the best performance, as indicated by the minimum validation loss and maximum validation accuracy, was achieved during the third epoch (10,616 steps). This observation reinforces the notion that the model's generalization capabilities peaked at the third epoch, after which it started to have overfitting.

The model's performance on the test sets was evaluated using CER, BLEU, DER, and WER. The results are presented in Table 15.

Table 15. Final Model: Performance Metrics on JODA and JODA+Clean-400 Test Sets.

Metric	JODA+Clean-400 Test Set	JODA Test Set
CER (%)	3.99	12.61
BLEU (%)	84.79	55.67
DER (%)	1.26	2.82
WER (%)	9.72	27.97
Total Score (%)	92.455	78.0675

As observed, the model exhibited significantly better performance on the JODA+Clean-400 test set compared to the JODA test set. This disparity can be attributed to the model's extensive training on formal Arabic, which constitutes a substantial portion of the JODA+Clean-400 dataset. The dataset primarily comprises CA texts with a smaller portion of MSA. Given the close similarity between MSA and CA, albeit with MSA's simpler expressions, this composition contributes to the model's strong performance in handling formal Arabic structures.

Conversely, the Jordanian dialect, with its inherent variability across regions and its relatively smaller representation in the training data, poses a more challenging task for the model. The dialect's diversity and complexity contribute to the observed differences in performance between the two test sets.

5.4.3 Manual Linguistic Evaluation

While automated metrics like CER, BLEU, DER, and WER provide quantitative assessments of model performance, they often fail to capture the nuances of language, particularly in the context of dialectal Arabic translation. Notably, metrics such as DER and WER are highly sensitive to variations in the target sequence, including synonymous expressions, additional or missing punctuation marks, and minor lexical differences, even when the model's prediction accurately conveys the intended meaning. These automated

evaluations do not account for such linguistic variations, potentially underestimating the model's true performance.

To address these limitations and provide a more human-centered evaluation, we conducted a manual assessment of 200 randomly selected samples from the test set. Due to time constraints and the complexity of manual evaluation, we focused on calculating CER and DER. These metrics, while not exhaustive, provide a representative measure of the model's performance in terms of character-level and diacritization accuracy. The 200 samples were allocated as follows: 100 for the task of converting to MSA only (the input text starts by the prefix "correct: "), and 100 for the task of conversion including diacritization (the input text starts by the prefix "diacritize: "). CER was evaluated using the 200 samples, while DER was evaluated using samples for the task conversion to MSA including diacritization.

The manual evaluation process involved two primary steps. First, we calculated CER to assess the accuracy of the model's character-level predictions. Subsequently, we calculated DER, specifically focusing on the accuracy of diacritization. To avoid double-penalization, we only punished the model for incorrect diacritics, considering any character-level errors as already accounted for in the CER calculation. This approach ensured that the final total score, which combines CER and DER, accurately reflects the model's performance in both character prediction and diacritization. For instance, if a word or character was incorrectly predicted, we treated it as correct for the DER evaluation, focusing solely on the accuracy of the diacritics applied to that incorrect word or character.

Crucially, it is important to note that the JODA test set, as detailed in Chapter 4, was generated using a machine diacritization process. Consequently, it contains a certain percentage of inherent diacritization errors. To comprehensively evaluate our model's

performance, we conducted the manual DER evaluation in two distinct ways: including and excluding these machine-generated errors. When including the errors from the machine generated diacritics, the DER score was higher than when the errors were excluded. This discrepancy highlights the impact of the test set's inherent noise on the evaluation results.

The results of the manual evaluation, including CER, DER with machine-generated errors, DER without machine-generated errors, and the final total score (calculated both with and without these errors), are presented in Table 16. The DER is further calculated with and without the diacritic of the last letter of each word, which is the hardest to predict since it is highly dependent on contextual and grammatical factors that vary significantly across sentences. The total score, which combines CER and DER, was calculated using Equation (5.1).

$$Evaluation\ Score = \frac{(100 - CER_{Score}) + (100 - DER_{Score})}{2} \quad (5.1)$$

Table 16. Final Model: Manual Evaluation Results, Including and Excluding Machine-Generated Errors.

Metric (%)	Value (Including Machine-Generated Errors)	Value (Excluding Machine-Generated Errors)
CER	4.64	4.64
DER (Including Last Letter's Diacritic)	1.93	1.32
DER (Excluding Last Letter's Diacritic)	1.03	0.73
Total Score (CER + DER - Including Last)	96.71	97.02
Total Score (CER + DER - Excluding Last)	97.16	97.31

It is important to note that the DER and total score calculated including machine-generated errors and including the last letter's diacritic represent the primary evaluation

metrics for our model's performance. The DER and total scores calculated excluding these factors are presented solely for demonstration purposes, to illustrate the impact of these variables on the evaluation results.

Figure 27 illustrates the differences in CER and DER between manual and automatic evaluation methods using the 200 random samples. The chart highlights that the evaluation methodology significantly impacts the perceived model performance. The manual evaluation, which accounts for linguistic nuances and contextual understanding, resulted in a higher total score of 96.715 compared to the automatic evaluation's 92.26, for the 200 random samples and using CER and DER metrics. This underscores the importance of manual evaluation in providing a more accurate assessment of the model's capabilities in handling the complexities of dialectal Arabic. It is important to acknowledge that the manual linguistic evaluation was a complex undertaking requiring specialized knowledge in Arabic language. To our knowledge, we avoided mistakes, but the possibility of undetected errors in our evaluation remains.

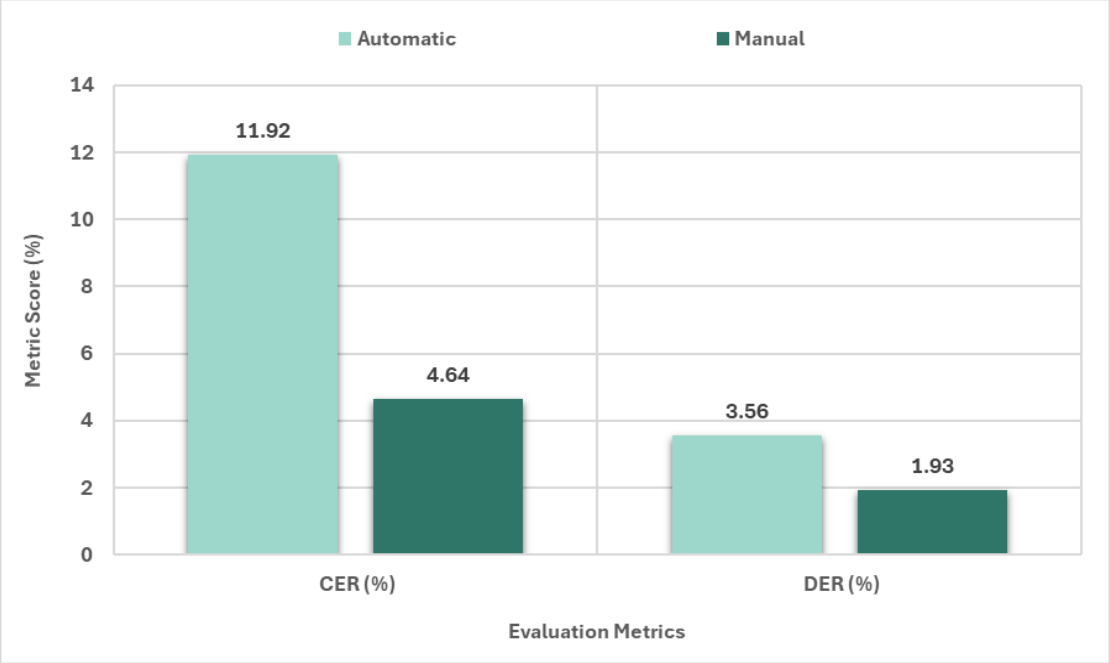

Figure 27. Impact of Evaluation Methodology: CER and DER Results from Automatic and Manual Evaluation.

Table 17 presents four illustrative examples from the JODA test set, showcasing the model's performance in various scenarios. As observed, the model demonstrates the ability to generate translations that are sometimes better than the target (first example), perfectly accurate (second example), and semantically correct using synonyms (third example). However, the model also produces incorrect translations in certain cases (forth example), highlighting the challenges associated with dialectal Arabic translation. These examples provide qualitative insights into the model's strengths and weaknesses, complementing the quantitative evaluation presented earlier.

Table 17. Illustrative Examples from JODA Test Set with Prediction Notes.

Input	Target	Predicted Text	Prediction Outcome
diacritize: سألو الفاشل شو سبب فشلك؟ حكالهم ما كان معي فلوس	سَأَلُوا الْفَاشِلَ مَا سَبَبُ فَشَلِكَ؟ قَالَ لَهُمْ مَا كَانَ مَعِي فُلُوسٌ	سَأَلُوا الْفَاشِلَ مَا سَبَبُ فَشَلِكَ؟ قَالَ لَهُمْ مَا كَانَ مَعِي نُقُودٌ	Better prediction
correct: الناس لحالها رح تيجي	الناس وحدها ستأتي	الناس وحدها ستأتي	Perfect prediction
diacritize: لحد الحين مافهمت شو المطلوب مني	لِلْآنَ لَمْ أَفْهَمْ مَا الْمَطْلُوبُ مِنِّي	حَتَّى الْآنَ لَمْ أَفْهَمْ مَا الْمَطْلُوبُ مِنِّي	Synonymous prediction
correct: من متى ادبي يحفظو الرياضيات	منذ متى طلاب الأدبي يحفظون الرياضيات؟	منذ متى أدبي يحفظ الرياضيات؟	Wrong prediction

Chapter Six

Conclusion and Future Work

The Arabic language, with its intricate grammar and the added complexity of diverse dialects, presents unique challenges for NLP. In this chapter, we summarize our findings, highlight our key contributions, discuss the limitations encountered, and propose potential avenues for future work in this domain.

6.1 Conclusion

This research contributed to address MTL setup for Arabic language, within a single model framework, incorporated two primary tasks: The first task is orthographic correction, which involved transforming text from Jordanian dialect or MSA with errors into corrected MSA text, and the second one is correction and diacritization, which expanded on the first task by generating corrected MSA text with diacritics.

The significance of this approach is underscored by the prevalence of spelling errors in Arabic text and the crucial role of diacritics in conveying intended meaning, thereby aiming to enhance readability and comprehension across text-based applications for both formal and colloquial Arabic, particularly Jordanian dialect.

Through a series of experiments exploring various architectural and hyperparameter configurations for fine-tuning the ByT5 model, we observed that incorporating high-quality, well-diacritized datasets, particularly Clean-50 and Clean-400, alongside the JODA dataset, which contains sentence in both Jordanian dialect and MSA with errors, led to notable performance improvements. Specifically, combining JODA with Clean-400 yielded the highest overall score of 78.06% (considering CER, BLEU, DER, and WER), outperforming the use of JODA alone (76.72%).

To gain deeper insights beyond automatic evaluation, we conducted a manual assessment on a subset of 200 randomly selected samples from the test set, focusing on CER and DER metrics. This manual evaluation, which inherently accounts for linguistic nuances and contextual understanding, revealed CER of 4.64% and DER of 1.93%, resulting in a higher total score of 96.715% compared to the automatic evaluation's 92.260% for the same 200 samples, highlighting the limitations of purely automatic metrics in capturing the complexities of dialectal Arabic.

6.2 Future Work

This research opens avenues for further exploration of unified MTL in Arabic and emphasizes the need for more sophisticated evaluation techniques, while also providing a valuable diacritized version of the MSA component of the JODA dataset. However, a more precisely diacritized version of the JODA dataset would be invaluable for further enhancing model performance. Also, extending the JODA dataset by introducing a new sentence or apply data augmentation and converting it to from a single-reference parallel corpus to multi-reference parallel corpus along with exploring more transfer learning techniques, like the incremental transfer learning introduced by Slim and Melouah (2024), would greatly provide the model with more nuances of the Jordanian dialect and ultimately enhance the model's performance. Furthermore, developing a more accurate automatic evaluation approach that better aligns with human linguistic judgment remains a crucial area for future work, potentially involving techniques that incorporate contextual understanding and linguistic nuances, which reduce the gap between the automatic and manual evaluation and may ideally dispense with the labor-intensive manual evaluation.

References

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

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

Abandah, G., Khaleel, M., Jafar, I., Abdel-Majeed, M., Hamdan, Y., Suyyagh, A., Abdel-Karim, A., AlAwawdeh, S. (2025, Jan). "Accurate translation of Jordanian Arabic to Modern Standard Arabic using a pre-trained model fine-tuned on a purpose-built Jordanian dataset and synthetic error injection," Submitted for publications.

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.

Abbad, H., & Xiong, S. (2020, April). Multi-components system for automatic Arabic diacritization. **In European conference on information retrieval** (pp. 341-355). Cham: Springer International Publishing.

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

Al Alili, S., & Hassan, W. (2017). Attitudes of Arabic and non-Arabic speaking parents toward the importance of learning Arabic in the United States. **Journal of the National Council of Less Commonly Taught Languages**, 21(1), 1-36.

Al-Ali, M. N., & Arafa, H. I. M. (2010). An experimental sociolinguistic study of language variation in Jordanian Arabic. **The Buckingham Journal of Language and Linguistics**, 3, 220-243.

Al-Ibrahim, R., & Duwairi, R. M. (2020, April). Neural machine translation from Jordanian Dialect to modern standard Arabic. **In 2020 11th International Conference on Information and Communication Systems (ICICS)** (pp. 173-178). IEEE.

Alimi, T., Boujebane, R., Derouich, W., & Belguith, L. (2024). Fine-Tuned transformers for translating Multi-Dialect texts to modern standard Arabic. **In World Academy of Science, Engineering and Technology International Journal of Cognitive and Language Sciences.** https://publications.waset.org/10013906.pdf

Al-Jarf, R. (2019). Effect of Social Media on Arabic Language Attrition. Online Submission.

Almajdoubah, A. N., Abandah, G. A., & Suvvagh, A. E. (2021, November). Investigating Recurrent Neural Networks for Diacritizing Arabic Text and Correcting Soft Spelling

Mistakes**.** **In** ***2021 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT)*** (pp. 266-271). IEEE.

Alnefaie, R., & Azmi, A. M. (2017). Automatic minimal diacritization of Arabic texts. **Procedia Computer Science**, 117, 169-174

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

Al-Qaraghuli, M., & Jaafar, O. A. (2024). Arabic soft spelling correction with T5. **Jordanian Journal of Computers and Information Technology**, 10(1).

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.

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.

Baniata, L. H., Park, S., & Park, S. B. (2018). A neural machine translation model for Arabic dialects that utilises multitask learning (MTL). **Computational intelligence and neuroscience,** 2018(1), 7534712.

Bouhdima, M.E. On Arabic Language Maintenance Among Arabs Living in Western Countries: A Review of Literature. Preprints 2022, 2022030226. https://doi.org/10.20944/preprints202203.0226.v1.

Boumaraf, A., Bekal, S., & Macoir, J. (2022). The Orthographic Ambiguity of the Arabic Graphic System: Evidence from a Case of Central Agraphia Affecting the Two Routes of Spelling. **Behavioural neurology**, 8078607.

Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., ... & Stoyanov, V. (2019). Unsupervised cross-lingual representation learning at scale. arXiv preprint arXiv:1911.02116.

Darwish, K., Mubarak, H., & Abdelali, A. (2017, April). Arabic diacritization: Stats, rules, and hacks. **In** ***Proceedings of the third Arabic natural language processing workshop*** (pp. 9-17).

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019, June). Bert: Pre-training of deep bidirectional transformers for language understanding. **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. 4171-4186).

Fadel, A., Tuffaha, I., & Al-Ayyoub, M. (2019, May). Arabic text diacritization using deep neural networks. **In 2019 2nd international conference on computer applications & information security (ICCAIS)** (pp. 1-7). IEEE.

Fadel, A., Saleh, M., Salama, R., & Abulnaja, O. (2024). MTL-AraBERT: an enhanced multi-task learning model for Arabic aspect-based sentiment analysis. **Computers**, 13(4), 98.

Farhan, W., Talafha, B., Abuammar, A., Jaikat, R., Al-Ayyoub, M., Tarakji, A. B., & Toma, A. (2020). Unsupervised dialectal neural machine translation. **Information Processing & Management**, 57(3), 102181.

Fashwan, A., & Alansary, S. (2017, April). SHAKKIL: an automatic diacritization system for modern standard Arabic texts. **In Proceedings of the Third Arabic Natural Language Processing Workshop** (pp. 84-93).

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

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

Ja'afreh, H., & Al-Saudi, J. (2024). Language Variation, Regional Identities, and Social Perceptions of Karak and Amman Dialects in Jordan: A Comparative Study. **International Journal of Linguistics, Literature & Translation**, 7(9).

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).

Khaleel, M. and Abandah, G. (2025). “Efficient Stochastic Error Injection for Optimizing Large Language Models in Arabic Spelling Correction,” The 3rd International Conference on new Trends in Computing Sciences (ICTCS’25).

Kudo, T., & Richardson, J. (2018). Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. arXiv preprint arXiv:1808.06226.

Luong, M. T., Pham, H., & Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. arXiv preprint arXiv:1508.04025.

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

Papineni, K., Roukos, S., Ward, T., & Zhu, W. J. (2002, July). Bleu: a method for automatic evaluation of machine translation. **In Proceedings of the 40th annual meeting of the Association for Computational Linguistics** (pp. 311-318).

Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-training.

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.

Schmidt, R. M. (2019). Recurrent neural networks (rnns): A gentle introduction and overview. *arXiv preprint arXiv:1912.05911*.

Skiredj, A., & Berrada, I. (2024). Arabic text diacritization in the age of transfer learning: Token classification is all you need. arXiv preprint arXiv:2401.04848.

Slim, A., & Melouah, A. (2024). Low resource arabic dialects transformer neural machine translation improvement through incremental transfer of shared linguistic features. **Arabian Journal for Science and Engineering**, 49(9), 12393-12409.

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., 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.

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.

Zaidan, O. F., & Callison-Burch, C. (2014). Arabic dialect identification. **Computational Linguistics**, 40(1), 171-202.

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

نموذج لغة كبير ثنائي الوظيفة لتصحيح الأخطاء الإملائية وتشكيل النصوص العربية: تجسير اللهجة الأردنية للعربية الفصحى

إعداد

ربيع امين عتوم

المشرف

الأستاذ الدكتور غيث علي عبندة

المشرف المشارك

الدكتور محمد رجب عبدالمجيد

ملخص

تمثل اللغة العربية، بتركيبها النحوي الدقيق وتنوّع لهجاتها، تحديات خاصة في مجال معالجة اللغة الطبيعية، لا سيما عندما تتداخل اللغة العربية الفصحى الحديثة مع اللهجات المحلية مثل اللهجة الأردنية. الطرق التقليدية المعتمدة على القواعد أو الطرق الإحصائية غالبًا ما تواجه صعوبة في معالجة هذه النصوص بدقة.

يقترح هذا البحث نموذجًا موحدًا يؤدي مهمتين رئيسيتين: تحويل النصوص من اللهجة الأردنية أو من العربية الفصحى التي تحتوي على أخطاء إلى نصوص عربية فصحى سليمة، وإضافة التشكيل (الحركات) إلى النص المصحح. ويوفر النموذج مرونةً في أداء هذه المهام، سواء كلًا على حدة أو بالتزامن، مما يُحسّن بشكل كبير من قابلية القراءة.

قيّمنا فعالية نموذج (ByT5)، الذي لا يعتمد على التجزئة، عن طريق ضبطه بشكل أساسي على مجموعة بيانات (JODA)، مدعومة بمجموعات فرعية من بيانات (Tashkeela) تحديداً، (Clean-50) و (Clean-400). كما استكشفنا بشكل منهجي تأثيرات المعاملات الفائقة والمعاملات التركيبية للنموذج لتحسين الأداء المتزامن للمهام، مستخدمين مفهوم البادئات الخاصة بكل مهمة،

والمستوحاة من نموذج (T5). تم تقييم النموذج باستخدام مقاييس معيارية شملت معدل خطأ التشكيل (DER)، معدل الخطأ على مستوى الأحرف (CER)، ومقياس BLEU، ومعدل الخطأ على مستوى الكلمات (WER).

أظهرت نتائج التقييم الآلي أداءً مميزًا، حيث حقق أفضل نموذج نسبة (78.06%) على مجموعة اختبار (JODA)، و (92.45%) على مجموعة اختبار شملت كلًا من (JODA) و (Clean-400). ومع إدراك محدودية المقاييس الآلية في رصد التفاصيل اللغوية الدقيقة، أجرينا تقييمًا يدويًا شمل 200 عينة عشوائية، وأظهر هذا التقييم نسبة نجاح بلغت (96.71%)، مما يؤكد كفاءة النموذج العالية. يبرز هذا البحث إمكانات نموذج (ByT5) ، ويمهد الطريق لتطوير أساليب معالجة لغة طبيعية أكثر تقدمًا وقدرة على التعامل مع تعقيدات النصوص العربية الفصحى واللهجات المحلية.