Jordanian Journal of Computers and Information Technology (JJCIT), Vol. 11, No. 03, September 2025.

JORDANIAN ARABIC TO MODERN STANDARD ARABIC TRANSLATION USING A LARGE MODEL TUNED ON A PURPOSE-BUILT DATASET AND SYNTHETIC ERROR INJECTION

Gheith A. Abandah[1,2], Moath R. Khaleel[1], Iyad F. Jafar[1], Mohammad R. Abdel-Majeed[1], Yousef H. Hamdan[3], Ashraf E. Suyyagh[1], Asma A. Abdel-Karim[1] and Shorouq M. AlAwawdeh[1]

(Received: 2-Mar.-2025, Revised: 7-Jun.-2025, Accepted: 8-Jun.-2025)

ABSTRACT

This paper addresses the challenge of accurately translating Jordanian Arabic into Modern Standard Arabic (MSA) and correcting common linguistic errors. Although MSA is the formal standard for Arabic communication, the widespread use of local dialects in social media and everyday interactions often results in texts laden with spelling and grammatical issues. To overcome these challenges, we present an end-to-end system based on a newly constructed Jordanian Arabic dataset (JODA) comprising 59,135 sentences, as well as the Tashkeela dataset perturbed through synthetic error injection. We employ ByT5, a large pre-trained language model that processes text at the byte level, making it resilient to spelling variations and morphological complexities common in Arabic dialects. Our experimental results show that fine-tuning ByT5 on JODA and a 10% error-injected Tashkeela subset notably improves both BLEU score and character error rate (CER). Combining JODA with the synthetically modified Tashkeela data reduces the CER to 4.64% on the Test-200 test set and 1.65% on the TSMTS test set. Moreover, manual inspections reveal that the model produces correct or near-correct translations in most cases. Finally, we developed a custom smartphone keyboard and a web portal to demonstrate how the system can be made easily accessible to interested users, offering a practical solution for millions of Arabic speakers seeking to produce accurate, diacritized MSA text. This solution is currently limited to the Jordanian dialect; future work will focus on developing similar datasets and solutions for other Arabic dialects.

KEYWORDS

Jordanian Arabic, Modern Standard Arabic, Dialectal translation, Large language models, Synthetic error injection, Natural-language processing, ByT5.

1. INTRODUCTION

Arabic, as the official language of over 20 countries, exhibits a rich linguistic diversity shaped by various regional dialects [1]. In Jordan, everyday communication relies heavily on an informal local dialect distinct from *Modern Standard Arabic* (MSA). While MSA remains the formal standard for written communication in official contexts, many Jordanians encounter difficulties expressing themselves accurately, often producing texts riddled with lexical, morphological, grammatical, syntactic and spelling errors. The proliferation of social media has further amplified this issue, as informal dialects and spelling inconsistencies dominate many online platforms [2].

To address these challenges, modern natural-language processing (NLP) techniques offer promising solutions by leveraging powerful pre-trained large language models. These models have demonstrated remarkable success in understanding and generating text across different languages, including Arabic, when sufficiently trained on diverse and high-quality examples [3]. However, collecting large-scale datasets that reflect the intricacies of informal dialects and embedding them in a unified framework for effective NLP applications pose significant hurdles. Despite recent advancements, current solutions for

1. G. Abandah (Corresponding Author), M. Khaleel, I. Jafar, M. Abdel-Majeed, A. Suyyagh, A. Abdel-Karim and S. AlAwawdeh are with Computer Engineering Department, The University of Jordan, Amman, Jordan. Emails: abandah@ju.edu.jo, moathkhaleel@outlook.com, iyad.jafar@ju.edu.jo, M.Abdel-Majeed@ju.edu.jo, A.Suyyagh@ju.edu.jo, a.abdelkarim@ju.edu.jo and sh.alawawdeh@gmail.com
2. G. Abandah is currently on a sabbatical leave with Department of Computer Science and Engineering, American University of Sharjah, Sharjah, UAE. Email: gabandah@aus.edu
3. Y. Hamdan is with Department of Arabic Language and Literature, The University of Jordan, Amman, Jordan. Email: yousefhamdan4@yahoo.com

translating dialectal Arabic to MSA remain unsatisfactory in terms of accuracy and robustness. This is largely due to the low-resource nature of the problem, as most dialects lack extensive parallel corpora. The development of additional high-quality, dialect-specific resources is therefore essential to improve translation performance and to enable fine-tuning of large models for this challenging task.

In this work, we present an end-to-end system designed to translate Jordanian Arabic into MSA, correct common linguistic mistakes and provide optional diacritization (automatic restoration of missing short vowel marks). Our project involved collecting 59,135 Jordanian Arabic sentences, spanning various dialectal usages and error types, then pairing them with carefully proofread MSA renditions. This dataset was augmented with additional resources to address the scarcity of real-world error examples. By fine-tuning pre-trained large language models on these combined resources, we have created a robust system capable of significantly improving the quality of Jordanian Arabic texts. While this solution is currently limited to the Jordanian dialect, it can be extended to other Arabic dialects as similar datasets become available.

The key contributions of our research can be summarized as follows. First, we provide a new, purpose-built Jordanian Arabic dataset that captures authentic usage and errors, serving as a valuable resource for future NLP research in Arabic. Second, we introduce synthetic spelling errors into a well-known diacritized dataset, enabling the model to learn extensive error patterns beyond the scope of the Jordanian dialect alone. Third, we fine-tune and evaluate a large language model for the translation task, demonstrating its effectiveness in handling informal dialect and spelling issues. Finally, we make the resulting models available through user-friendly web and smartphone applications, allowing Jordanians to produce clear and accurate MSA texts.

After this introduction, Section 2 reviews some related previous work. The approach is outlined in Section 3, followed by the datasets in Section 4, which includes the Jordanian dialect dataset, the Tashkeela datasets with synthetic error injection and the test sets. Section 5 focuses on the models and experiments, describing the model tuning, optimization of synthetic error injection and training using the developed datasets. The results and discussion are presented in Section 6, encompassing a manual inspection of model predictions and a detailed analysis of the results. Finally, the paper concludes with insights, implications and future work in Section 7.

2. LITERATURE REVIEW

This review traces the evolution of machine translation, from rule-based methods to neural architectures, focusing on large language models (*e.g.*, GPT, BERT, T5 and ByT5) and highlighting their key features. Finally, it examines recent approaches for translating Arabic dialects into MSA.

2.1 Evolution of Machine-translation Approaches

Traditional language-translation methods, such as *rule-based machine translation* (RBMT), rely on comprehensive morphological, semantic and syntactic rules for both the source and target languages, requiring extensive expert input [4]. In contrast, *example-based machine translation* (EBMT) maps sentence examples from one language to another without requiring any handcrafted linguistic rules. However, its performance is heavily influenced by the quality of the example database [5]. *Statistical machine translation* (SMT), which was once dominant, integrates phrase, syntax and hierarchical models, but its complexity necessitates combining translation, language and sentence-reordering models [6]-[9]. Hybrid approaches that combine RBMT and SMT have also been explored [10].

Recently, *neural machine translation* (NMT) has become the standard, with widespread adoption by companies, like Google and Microsoft [6], [11]-[13]. NMT employs advanced models, like recurrent neural networks (RNNs), convolutional neural networks (CNNs), encoder-decoder stacks and transformers. With sufficient training data, these models can learn complex linguistic relationships and capture context and semantics from parallel data [12]-[13]. Popular NMT variants, such as bidirectional encoder representations from transformers (BERT) [14]-[15] and text-to-text transfer transformer (T5) [16], are widely used for natural-language processing. NMT systems, initially focused on language pairs, are now capable of translating across 200+ languages [17].

2.2 Large Language Models

Large language models (LLMs), such as generative pre-trained transformer (GPT) [18], BERT [19] and T5 [16] have significantly advanced NLP. These models are based on the *transformer* architecture, which uses self-attention mechanisms to process text in parallel rather than sequentially [20]. This parallel processing enables LLMs to better handle long-range dependencies and complex linguistic structures. Trained on large datasets, LLMs can perform various tasks, like text generation, translation, summarization and error correction, making them versatile tools for language applications. However, models like GPT and T5, which rely on token-based representations, may struggle with out-of-vocabulary words or small typographical errors.

ByT5 is a *token-free* variant of the T5 model that operates directly on byte-level inputs rather than relying on tokenized text [21]. It retains T5's core architecture, consisting of a heavy encoder and a lighter decoder, both built with multi-head self-attention mechanisms and feed-forward neural networks. The encoder converts raw byte sequences into continuous representations, effectively capturing semantic meaning even in the presence of spelling errors or non-standard formatting, while the decoder generates coherent output sequences from these representations. This byte-level processing eliminates the limitations of traditional tokenization, enabling ByT5 to handle diverse languages and character sets more flexibly.

We adopt ByT5 in our solution due to its demonstrated robustness against spelling variations, misspellings and unconventional text formats—characteristics that are prevalent in dialectal and informal Arabic. These strengths make it particularly well-suited for tasks, such as error correction, normalization and diacritization. ByT5 has also proven effective in Arabic NLP applications, including automatic text diacritization [22].

2.3 Recent Approaches to Translating Arabic Dialects

This sub-section reviews recent efforts in Arabic-dialect translation, arranged from broader to more closely related work.

Some studies have focused on translating Arabic dialects to or from English. Alzamzami and Saddik [23] proposed a transformer-based model for translating English tweets into four Arabic dialects. Nagoudi *et al*. [24] developed AraT5, a transformer model pre-trained on large-scale data and fine-tuned on several tasks, including Arabic dialect-to-English translation. AraT5 outperformed the more general multilingual mT5 model in these tasks.

Several other studies have targeted the translation of *multidialectal Arabic content* into MSA. Slim and Melouah [25] addressed the translation of three Maghrebi dialects into MSA using an incremental fine-tuning strategy on a transformer model to address the low-resource nature of dialectal Arabic. Baniata *et al*. [26] proposed integrating RNN-based part-of-speech tagging to enhance translation from Levantine and Maghrebi dialects into MSA, achieving a BLEU score of 43 for Levantine dialects. Alimi *et al*. [27] fine-tuned a variant of AraT5 for translating Levantine and Maghrebi dialects into MSA, reporting high BLEU scores of 43.38 and 64.99, respectively. Notably, both works on Levantine dialects include coverage of the Jordanian dialect.

There is also a line of research focusing on the translation of a *single dialect*, which aligns more closely with our work. Kchaou *et al*. [28], [29] applied data-augmentation techniques to Tunisian-dialect translation and demonstrated that a transformer model outperformed CNN and RNN baselines, achieving a BLEU score of 60. Faheem *et al*. [30] focused on translating the Egyptian dialect into MSA. Their model, trained on 40,000 supervised parallel sentences and supplemented with 35 million monolingual sentences in an unsupervised manner, achieved a BLEU score of 29.5.

Our approach aligns with Refs. [29]-[30] in targeting the translation of a single Arabic dialect into MSA and with Refs. [30], [27], [25], [24] in fine-tuning transformer-based models. However, we distinguish our work by adopting a pretrained, token-free transformer (ByT5), which we fine-tune using a parallel Jordanian-MSA dataset and stochastic error injection. To the best of our knowledge, this is the first work to fine-tune a transformer model specifically for translating not only Jordanian dialect, but also error-prone MSA text—including linguistic and spelling errors—into proper MSA.

3. APPROACH

Our research implements a comprehensive approach, shown in Figure 1, to accurately translate Jordanian Arabic into MSA, correct spelling mistakes and add diacritics. We began by collecting a dataset of 59,135 Jordanian Arabic sentences, encompassing a broad spectrum of language mistakes and dialectal variations. Working with Arabic-language specialists, we corrected mistakes, translated colloquial forms into MSA and thoroughly proofread all samples.

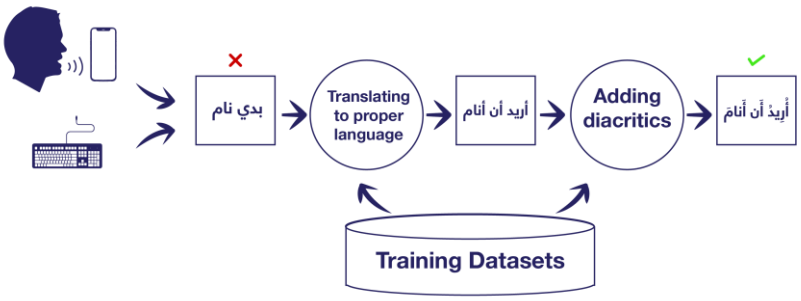

Figure 1. End-to-end approach for Jordanian Arabic to diacritized MSA conversion.

Building on this dataset, we further expanded it using the diacritized Tashkeela Classical Arabic dataset [31]. Synthetic spelling errors were introduced into Tashkeela via random error injection, enhancing the model's capacity to handle real-world misspellings.

We fine-tuned pre-trained ByT5 models, leveraging their broad language understanding developed through training on large datasets. *Pre-trained* models like ByT5 are neural networks designed to learn general language representations, enabling them to understand and generate text effectively. *Fine-tuning* involves adapting these models to specific tasks by training them further on smaller, task-specific datasets. In our case, one model was fine-tuned to translate Jordanian Arabic into proper MSA, specializing in this linguistic transformation. Additionally, we explored another model inspired by Al-Rfooh *et al*. [22] to optionally add diacritics, though this lies outside the scope of this paper [32], [33].

Upon completion of training, the models exhibited strong performance in error correction, translation and diacritization. Finally, we integrated these trained models into both internet-based and smartphone applications, exploring open access for Jordanian users seeking reliable and accurate linguistic support. Despite its effectiveness, the approach faces limitations including the cost of developing high-quality parallel datasets and the computational intensity of training and deploying large models like ByT5, which constrains scalability and performance on resource-limited devices.

4. DATASETS

This section describes the datasets used for training and evaluating our approach. Sub-section 4.1 details the newly developed Jordanian dialect dataset [34], Sub-section 4.2 introduces the Tashkeela-based datasets alongside synthetic error injection and Sub-section 4.3 outlines the test sets employed to measure model performance.

4.1 Jordanian Dialect Dataset

One key contribution of this research is the development of the *Jordanian dialect dataset* (JODA). This parallel dataset was constructed by collecting Arabic sentences that contain various linguistic mistakes or are in the informal Jordanian dialect. Each collected sentence was then paired with its corresponding correct MSA equivalent. The dataset draws from three primary sources to ensure diversity and authenticity (Figure 2). Approximately 72% of JODA comes from social-media platforms: YouTube, Facebook, Instagram and Twitter (X), covering various topics like economics, society and politics. Additionally, 22% are sentences selected from publicly available Arabic-dialect datasets: Dialectal Arabic tweets (DART) dataset [35] and the Shami dialect corpus (SDC) [36]. The remaining 6.6% of the dataset consists of transcriptions of eight short Jordanian movies capturing cultural and linguistic diversity.

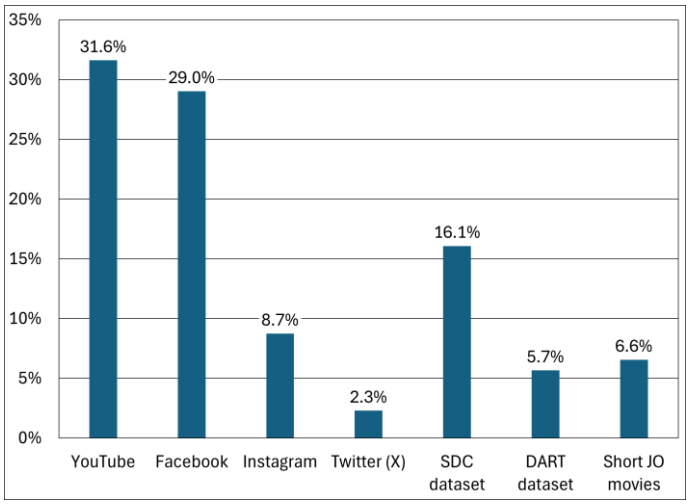

Figure 2. Composition of JODA dataset by sample source.

The collected samples from the various sources underwent extensive preprocessing, which included removing irrelevant elements, duplicates, emojis and unnecessary characters, as well as segmenting the text into meaningful sentences. Each sentence was manually reviewed to ensure proper segmentation and meaningful content, retaining only those in MSA containing mistakes or in the Jordanian dialect. The sentences range from 2 to 277 characters, reflecting natural-language usage. Arabic linguistic experts contributed to the development of this parallel dataset by providing either corrections for MSA sentences containing mistakes or translating Jordanian dialect sentences into MSA. For a broader linguistic perspective on this translation from Jordanian Arabic into MSA, interested readers are referred to [37].

While JODA was designed to be as representative as possible, some bias may exist. The Jordanian dialect varies by region, but most data likely reflect the central region, where most of the population resides. Northern and southern dialects may be underrepresented. Additionally, the heavy reliance on social-media content may skew the language toward younger, urban speakers. We also used curated datasets and film transcripts, which may not fully capture spontaneous speech. Despite these limitations, we made deliberate efforts to ensure diversity in topics, sources and linguistic styles across the dataset.

To expedite dataset corrections, we developed a custom PyQt-based GUI specifically tailored for Arabic-text processing. The tool is employed by both experts and auditors, who can selectively load dataset files, navigate individual sequences, classify entries and either provide or validate corrections. This interface was designed to accommodate right-to-left scripts and fully support Arabic display and parsing, ensuring minimal friction during annotation and review. Additionally, it offers streamlined functionality for saving changes, flagging problematic entries and maintaining detailed logs of edits. Figure 3 illustrates the tool's layout and features, highlighting its user-friendly design.

Figure 3. Correction and auditing tool.

The final version of the JODA dataset comprises 59,135 sentences, with 62.4% in the Jordanian dialect and 37.6% in MSA containing mistakes (Table 1). This version was randomly split into three sub-sets; 91.5% of the sentences were included in the training sub-set, while the remaining sentences were evenly divided between the validation and test sub-sets (2,500 sentences each).

Table 1. Distribution of the JODA dataset by sentence type and data split. The "Total" row and column show the number of sentences and their percentages relative to the entire dataset.

Sentence type	**Training subset**	**Validation subset**	**Test subset**	**Total**
Jordanian dialect	33,767	1,560	1,559	36,886 (62.4%)
MSA containing mistakes	20,368	940	941	22,249 (37.6%)
Total	54,135 (91.5%)	2,500 (4.2%)	2,500 (4.2%)	59,135 (100%)

During this split, stratification was applied to ensure representative sampling of the various sentence sources and types across the three sub-sets. Figure 4 shows the number of sentences in the three dataset sub-sets, categorized by sentence source and sentence type.

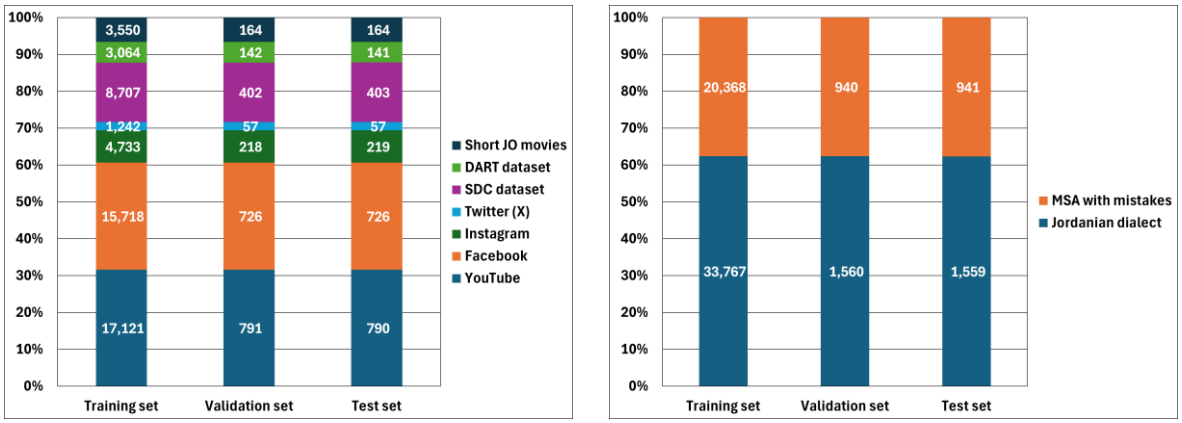

Figure 4. Stratified split of the JODA dataset by sentence source (left) and sentence type (right).

4.2 Tashkeela Datasets and Synthetic Error Injection

In addition to JODA, the proposed model was also trained using the Tashkeela Clean-50 and Clean-400 datasets, which primarily contain diacritized Classical Arabic text. The *Tashkeela Clean-50* dataset, developed by Fadel *et al*. [38], comprises 50,000 training sequences extracted from the original Tashkeela dataset [31]. These sequences were filtered to ensure a diacritic-to-character ratio of at least 80% and were processed using heuristics, such as diacritic correction, removal of English letters and isolation of numbers from words. Abdel-Karim and Abandah [39] expanded this dataset, creating the *Tashkeela Clean-400* dataset with 400,000 training sequences. Both datasets include, in addition to their respective training sets, the same validation sub-set of 2,500 sequences and the same test sub-set of 2,500 sequences. These datasets were truncated to a maximum sequence length of 512 bytes to maintain consistency with the JODA dataset.

These datasets were further processed into input-target pairs by introducing synthetic stochastic spelling errors [40]. Two methods were employed for error injection: directed error injection and general error injection. *Directed error injection* focuses on "soft spelling mistakes," which are common among Arabic speakers and learners due to the complexity of Arabic orthography. Following the approach of Abandah *et al*. [41], this method specifically targets frequent mistakes involving words with different forms of *hamza* (ء، أ، إ، آ، ؤ، ئ) and words ending with similarly pronounced letters (ه، ة، ت) and (و، وا). Errors were introduced based on their position within words, using three injection rates (2.5%, 10% and 40%) to evaluate their impact on model training. This method ensures that artificial errors closely resemble common real-world mistake patterns.

General error injection extends directed error injection by incorporating a broader range of spelling error patterns, including letter deletion, insertion, swapping and replacement. This approach introduces stochastic errors of selected probability, also evaluated at three injection rates. These errors simulate

various mistake patterns found in Arabic text, allowing the model to learn corrections for a variety of mistake types. By combining directed and general error injection methods, the dataset is designed to improve the model's ability to correct both specific and general spelling mistakes. Table 2 provides the statistics for JODA, Tashkeela Clean-50 and Tashkeela Clean-400.

Table 2. Statistics of the datasets used.

Metric	**JODA**	**Tashkeela Clean-50**	**Tashkeela Clean-400**
Size (MB)	10.3	12.80	102.50
Number of sequences	59,135 pairs	50,000	400,000
Word count	1.14×10^6	1.62×10^6	12.95×10^6
Character count	5.92×10^6	7.34×10^6	58.67×10^6
Average number of words per sequence	9.6	32.40	32.36

4.3 Test Sets

To thoroughly evaluate the developed model's performance, we use three test sets. The first is the *JODA test subset* described above, which is critical for assessing performance and selecting optimal configurations. The second, *Test-200* [41], contains 200 sentences with "soft" spelling mistakes, averaging 6.5 mistakes per sentence and a 5%-character mistake rate. This set is particularly useful for fine-tuning the model when training on data with directed error injection.

We also developed a third test set, the *Tashkeela spelling mistakes test set* (TSMTS), derived from the 2,500 sequences of the Tashkeela test set. Each sequence in the Tashkeela test set serves as a target, paired with an input sequence generated by applying the general error injection described above to the original sequence. A character error rate of 5% was used to ensure that TSMTS mirrors the Test-200 set. This test set provides a benchmark for evaluating general error injection.

5. MODELS AND EXPERIMENTS

We selected ByT5 for its robustness in handling multilingual text and noisy inputs, operating at the byte level without tokenization. This language-agnostic approach ensures high flexibility across diverse languages and scripts [21]. ByT5's strengths include resilience to misspellings and compatibility with low-resource languages. For our experiments, we utilized the Small and Base model sizes due to their lower computational requirements. We did not use larger models, as the significantly higher computational cost was not justified by the relatively small performance gains reported in prior work [21]-[22]. Table 3 summarizes the architectures of both models.

Table 3. Architectures of the two ByT5 models explored.

Criterion	**Small**	**Base**
Number of parameters	300M	582M
Encoder/decoder layers	12 / 4	18 / 6
Feed forward dimension (d_{ff})	3,584	3,968
Model dimension (d_{model})	1,472	1,536

For evaluation, we used the BLEU and CER metrics. BLEU (*bilingual evaluation understudy*) measures the similarity between the model's output and reference translations by comparing overlapping n-grams, providing a score for translation quality. CER (*character error rate*) calculates the percentage of character-level errors, such as substitutions, insertions and deletions, in the model's output compared to the reference, offering insight into fine-grained accuracy.

The experiments were conducted on Google's Colab Pro Plus platform, utilizing TPU v2 units to accelerate the training process. The programming language used was Python 3.7.13, with TensorFlow 2.12.0 as the primary library.

The following sub-sections detail the experiments and results for tuning the ByT5 model, refining the error injection approach used in preparing the Tashkeela datasets and training the optimized model on a combined dataset of JODA and Tashkeela.

5.1 Tuning the ByT5 Model

The ByT5 model comes in multiple sizes and offers numerous hyperparameters that can be adjusted to improve performance, depending on the target task. In this work, we began by establishing a baseline model and then explored various hyperparameter configurations to arrive at a final tuned model. Table 4 summarizes the explored hyperparameter options and lists the values used in both the baseline and the tuned models. The following paragraphs describe the tuning experimental procedure and summarize the results.

Table 4. Explored ByT5 hyperparameters, evaluated options and the corresponding values for both the baseline and the tuned models.

Hyperparameter	Options	Baseline model	Tuned model
Model size	Small, Base	Small	Base
Batch size	128, 256, 512	256	128
Learning rate	0.0001, 0.003, 0.01	0.003	0.003
Optimizer	AdaFactor, Adam Weight Decay	AdaFactor	Adam Weight Decay

We fine-tuned the model using the JODA dataset, which includes the two implicit tasks: translating Jordanian Arabic into MSA and correcting linguistic mistakes. Our initial experiment assessed the baseline model's performance. Figure 5 shows the BLEU scores for both the training and validation sub-sets over successive training steps, where each step corresponds to a batch of a specified size (256 for the baseline model). During this experiment and others, we observed that the model exhibits overfitting, with the BLEU score on the training sub-set approaching 100 while the validation score plateaus at a lower level. To mitigate overfitting, we halted training when the validation score ceased to improve and adopted the model weights from the training step with the highest validation score. The baseline model achieves its highest BLEU score of 57.49 at the 3,000th training step, with a corresponding BLEU score of 56.07 on the JODA test sub-set.

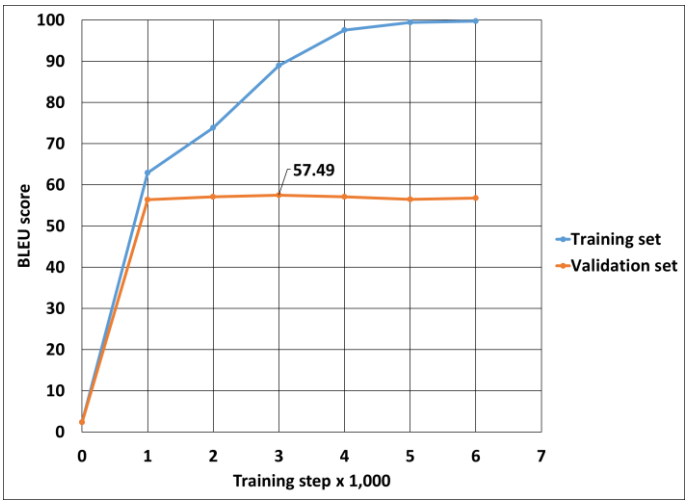

Figure 5. Training curves of the baseline model trained on JODA dataset.

In our fine-tuning experiments, we followed the methodology described in [42], which involves adjusting one hyperparameter at a time and comparing the resulting performance to the baseline. Although this "coordinate ascent" approach may overlook higher-order interactions between parameters (for instance, a different learning rate might produce better results with a larger model size), a full factorial design would be expensive, as it would require 2×3×3×2 = 36 experiments. Once the best individual hyperparameters were identified, we used those values to train the final model.

Table 5 provides the outcomes of the seven fine-tuning experiments involving the four hyperparameters. Each row presents the examined hyperparameter option, the training step where the validation score peaked and the corresponding BLEU scores for both the validation and test sub-sets. Based on these results, the optimal hyperparameters for the tuned model are those shown in Table 4.

Table 5. Results of fine-tuning the hyperparameters of the ByT5 model.

Hyperparameter	Option	Best training step	BLEU score (validation)	BLEU score (test)
Model size	Small (baseline)	3,000	57.49	56.07
	Base	7,000	59.01	**57.08**
Batch size	**128**	4,000	57.54	**56.43**
	256 (baseline)	3,000	57.49	56.07
	512	1,000	58.03	56.32
Learning rate	0.0001	20,000	56.53	55.38
	0.003 (baseline)	3,000	57.49	**56.07**
	0.01	9,000	55.13	53.90
Optimizer	AdaFactor (baseline)	3,000	57.49	56.07
	Adam Weight Decay	2,000	57.60	**56.29**

When trained on JODA, the tuned model achieves its highest BLEU score of 59.07 at the 3,000th training step on the validation sub-set, yielding a BLEU score of 57.77 on the test sub-set, which represents a 3% improvement over the baseline model.

5.2 Tuning Error Injection

The performance of a model trained with synthetic error injection is influenced by the chosen injection rate in [41]. This sub-section describes the experiments conducted to determine optimal rates and summarizes the results. In these experiments, we trained the tuned model on the Tashkeela datasets and evaluated it on the Test-200 or TSMTS test sub-sets. As in previous experiments, we stopped training once the validation score ceased to improve and adopted the model weights from the training step that produced the highest validation score for final evaluation.

5.2.1 Directed Error Injection

We explored three rates for directed error injection: 2.5%, 10% and 40%. In each experiment, the model was trained on a Tashkeela dataset with the specified rate of directed error injection, then evaluated on Test-200. We selected Test-200, because it contains common real-life spelling mistakes, like those introduced by the directed method.

Table 6 shows the results obtained using the Clean-50 dataset, where a 10% injection rate yielded the lowest CER on Test-200 (1.37%). Note that the CER on the validation sub-set increases with higher error rate in this sub-set. The table also reports results for training on the larger Clean-400 dataset at the same 10% rate, which further reduced the CER on Test-200 to 1.23%. This improvement demonstrates that a larger dataset provides the model with more examples of spelling variations, enhancing its ability to correct errors.

Table 6. Results of tuning directed error injection.

Dataset	Error injection rate	Best training step	CER (validation sub-set)	CER (Test-200)
Clean-50	2.5%	8,000	0.03%	2.26%
	10%	8,000	0.07%	**1.37%**
	40%	8,000	0.14%	1.53%
Clean-400	**10%**	13,000	0.04%	**1.23%**

5.2.2 General Error Injection

For general error injection, we similarly evaluated three rates: 2.5%, 10% and 40%. In each experiment, the model was trained on a Tashkeela dataset with the chosen rate of general error injection and tested on TSMTS. TSMTS was selected, because it contains synthetic spelling errors comparable to those produced by the general error injection method.

As shown in Table 7, using the Clean-50 dataset with a 10% injection rate resulted in the lowest CER on TSMTS (1.77%). When the model was trained on the larger Clean-400 dataset at the same 10% rate, the CER dropped further to 1.28%, indicating that a bigger training sub-set helps the model better generalize to diverse error patterns.

Table 7. Results of tuning general error injection.

Dataset	Error injection rate	Best training step	CER (validation sub-set)	CER (TSMTS)
Clean-50	2.5%	14,000	0.80%	2.09%
	10%	10,000	2.99%	**1.77%**
	40%	12,000	16.69%	2.78%
Clean-400	**10%**	14,000	2.20%	**1.28%**

Overall, these experiments confirm that a 10% error injection rate is most effective for both directed and general error injection methods. Furthermore, training on a larger dataset (Clean-400) yields better results, highlighting the importance of data size in improving the model's ability to correct spelling errors.

5.3 Training Using JODA and Tashkeela Datasets

To further improve the model's performance on both translating Jordanian Arabic into MSA and correcting linguistic mistakes, we explored training on a combined dataset. Specifically, we combined JODA with the 10% directed error-injected Clean-50 dataset and the 10% general error-injected Clean-50 dataset. As usual, this combined dataset was partitioned into training, validation and test sub-sets by merging the corresponding sub-sets from the three individual datasets.

Figure 6 illustrates the training curves for the tuned model on this combined dataset. The BLEU score for the training sub-set continued to improve with more training steps, whereas the validation score increased more slowly. Training was halted at Step 15,000 due to the slowing improvement on the validation sub-set and the widening gap between the training and validation scores. At this step, the validation BLEU score reached 87.57, which is considerably higher than the BLEU score of 59.07 achieved by training solely on the JODA dataset. This apparent discrepancy arises, because the validation sub-set in the single-dataset experiment contains only JODA sentences, which tend to be more challenging than the mixed-validation sub-set here. Indeed, when evaluated on the JODA test sub-set, this model achieves a BLEU score of only 57.39.

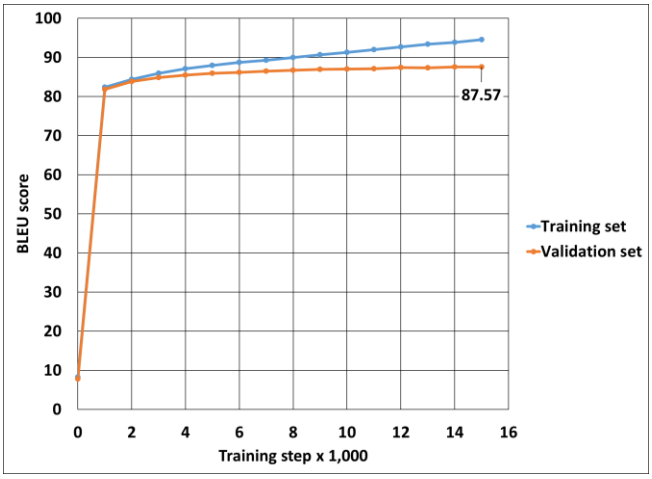

Figure 6. Training curves of the tuned model trained on JODA and Clean-50 datasets.

To assess whether the model could generalize beyond JODA, we compared the CER on the Test-200 and TSMTS test sub-sets between (1) the model trained on JODA only and (2) the model trained on the combined dataset. As shown in Figure 7, the combined-dataset model generalizes more effectively: the CER on Test-200 improves from 6.37% to 4.64% and on TSMTS from 11.95% to 1.65%. This result demonstrates the model's enhanced ability to correct common and general spelling mistakes.

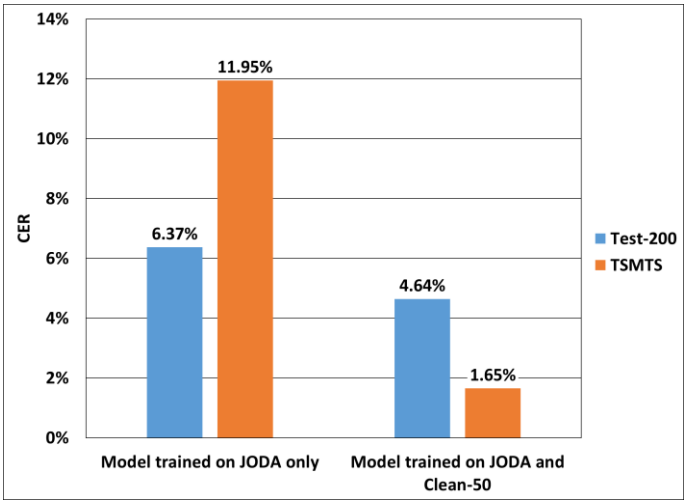

Figure 7. CER on two test sub-sets for the tuned model trained with two dataset configurations.

We also examined the model's performance when trained on a combined dataset consisting of JODA and the larger Tashkeela Clean-400 dataset. In this case, the model yielded a lower BLEU score of 53.24 on the JODA test sub-set, likely due to an imbalance between Jordanian dialect and MSA content in the larger dataset. Consequently, we adopted the model trained on the combined JODA and Clean-50 datasets.

6. Results and Discussion

Table 8 compares the three main models trained under different conditions to evaluate their performance in translating Jordanian Arabic into MSA and correcting linguistic mistakes. The baseline model, trained only on JODA, reaches a high BLEU 56.07, because many JODA references differ from the inputs by only minor spelling errors; n-gram overlap is therefore already near-saturated. However, CER exposes those spelling mistakes: the baseline scores 6.58% on Test-200 and 12.41% on TSMTS. Hyper-parameter tuning (still on JODA) nudges BLEU to 57.77 and trims CER to 6.37% and 11.95%. Adding the Clean-50 corpus introduces many perfectly spelled targets and forces the model to generalize beyond JODA. BLEU on the JODA test sub-set dips slightly to 57.39, but CER falls sharply to 4.64% on Test-200 and 1.65% on TSMTS. Thus, while BLEU shows only marginal gains, the steep CER reduction demonstrates that the final model corrects errors more aggressively and transfers this ability to unseen text, striking a practical balance between fluency (BLEU) and accuracy (CER).

Table 8. Comparison of the three main experiments on three test sub-sets.

Model	**Training time in hours**	**BLEU score (JODA test set)**	**CER (Test-200)**	**CER (TSMTS)**
Baseline model (trained on JODA)	1.5	56.07	6.58%	12.41%
Tuned model (trained on JODA)	2.1	57.77	6.37%	11.95%
Tuned model (trained on JODA + Clean-50)	10.3	57.39	4.64%	1.65%

Although large language models deliver impressive results, they often come with substantial computational costs. Table 8 lists the training times for the three models, showing that the tuned model employing the base ByT5 requires longer training than the baseline model, which uses the smaller ByT5 variant. Moreover, the final model trained on the combined larger dataset increases training time to around five times that of the tuned JODA-only model. In the prediction mode, the trained model translates a single Jordanian dialect sentence into MSA in approximately 1.5 seconds.

6.1 Comparison with Previous Work

Table 9 presents a comparative overview of recent efforts in translating Arabic dialects into MSA, highlighting the methods, datasets and BLEU scores reported for different dialects. Compared to previous studies, our work utilizes JODA—the largest Arabic mono-dialect dataset focused on Jordanian

Arabic—and achieves the highest BLEU score reported for Levantine dialects, demonstrating the effectiveness of our fine-tuned ByT5 model with stochastic error injection.

Table 9. Comparison with previous work in translating Arabic dialects into MSA.

Work	**Method**	**Dataset/Size**	**BLEU score**
Baniata *et al*. [26]	RNN with POS tagging	Multidialectal / 36K	43 for Levantine dialect
Alimi *et al*. [27]	Fine-tuning pretrained AraT5 model	Multidialectal / 69K	43.38 for Levantine dialect
Kchaou *et al*. [29]	Transformer with data augmentation	Tunisian dialect / 36K	60
Faheem *et al*. [30]	Pretraining followed by fine-tuning a transformer	Egyptian dialect / 40K	29.5
This work	Fine-tuning ByT5 and stochastic error injection	JODA / **59K**	**57.77**

6.2 Manual Inspection of the Model Predictions

Throughout this research, we primarily relied on BLEU and CER scores to evaluate translation and correction quality. While these metrics are generally appropriate, they can also underestimate the model's performance—especially given Arabic's linguistic richness, where a single sentence can be correctly translated into multiple valid forms. Consequently, the model may receive a penalty if it produces a legitimate translation that does not exactly match the target sequence.

Our main objective is to develop a model capable of translating Jordanian Arabic into MSA, rather than replicating a specific reference translation. To gain deeper insight into the model's real-world performance, we manually reviewed 100 randomly selected predictions from the JODA test sub-set, comparing them with the expert-provided target sentences. Table 10 provides sample inputs, target sentences and model outputs, categorized into three classes: (1) correct translations with contextual variations, (2) translations exhibiting minor differences and (3) translations with more substantial discrepancies, often reflecting synonym usage.

Table 10. Overview of 100 manually audited predictions.

	Prediction classification	**Count**	**Example input sentence**	**Target sentence**	**Predicted sentence**
1	Correct predictions	51	بامكاني انزل لمستواك و احكي بالطريقة الهمجية الي بتحكي فيها ،	بإمكاني أن أنزل إلى مستواك وأتكلم بالطريقة الهمجية التي تتكلمين بها	بإمكاني أن أنزل إلى مستواك وأتكلم بالطريقة الهمجية التي تتكلم بها
2	Small differences	40	المذيعه مش معاه ابدا ما خلتله مجال يحكي أسلوبها مزعج	المذيعة ليست معه أبدا، لم تترك له مجالا ليتكلم، أسلوبها مزعج	المذيعة ليست معه أبدا، لم تترك له مجالا ليتكلم بأسلوبها مزعج
3	Large differences	9	يا بنت الحلال مهو طول عمره هون شو عملك يعني	يا بنت الحلال، هو طوال عمره هنا، ماذا فعل لك إذا؟	يا بنت الحلال فهو طوال عمره هنا ماذا عمل لك إذا؟

Notably, the manual review revealed that the model's true performance exceeds the CER evaluation of 12.39%. When correct translations are not counted as errors, the CER declines to 5.56%. Although these findings are already encouraging, they further confirm that the model's practical performance is stronger than what traditional metrics alone may indicate.

Despite strong overall results, the final model shows two noteworthy limitations. First, ambiguous short sentences, where number or gender is underspecified, can lead to incorrect disambiguation in MSA. Second, because training data was restricted to Jordanian Arabic and MSA, the system struggles with inputs that mix codes (*e.g.* bilingual Arabic and English sentences). Addressing these issues will require additional annotated data and explicit modeling of ambiguity.

6.3 Evaluation on Additional Jordanian and Other Levantine-dialect Sentences

We conclude that our final model effectively translates Jordanian Arabic into MSA and corrects linguistic errors.

Table 11 provides examples of input sentences that were not part of the JODA dataset, alongside the model's corresponding predictions and evaluations. In most instances, the model produces high-quality translations, occasionally exhibiting only minor errors.

Table 11. Sample input sentences, model predictions and evaluations.

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

To probe generalizability beyond Jordanian Arabic, we manually assembled fifteen unseen sentences, five each in Palestinian, Syrian and Lebanese dialects and translated them with the final model. The model successfully rendered all sentences into grammatical MSA, confirming that its byte-level representations capture many shared Levantine structures. Accuracy, however, was lower than for Jordanian input: output fluency occasionally suffered from dialect-specific lexemes and translations of Lebanese examples that contained French loanwords (*e.g. ascenseur*, *parfum*). These observations suggest that while the system generalizes reasonably within the Levantine group, expanded training data would be needed for consistently high performance across all regional variants.

6.4 Accessing the Model *via* Smartphones and Web Portal

To provide the model's Jordanian Arabic-to-MSA translation and Arabic error correction capabilities to end users, we developed a custom keyboard and a web-based portal. The model is hosted on a server and communicates with both the keyboard and web interface using the Flask framework. When users enter text and request a correction, the front end sends this text to the Flask API, which processes it through the trained model and returns the corrected output in real time. This setup ensures a responsive, lightweight user experience by offloading complex processing tasks to the server.

The custom keyboard, called *AI Board*, was developed using the open-source OpenBoard project [43] for Android and the KeyboardKit 7.9.8 package [44] for iOS. As shown in Figure 8, it features a dedicated "صحح" (Correct) button that translates or corrects any text entered *via* the Arabic keyboard or microphone, seamlessly converting Jordanian dialect into MSA.

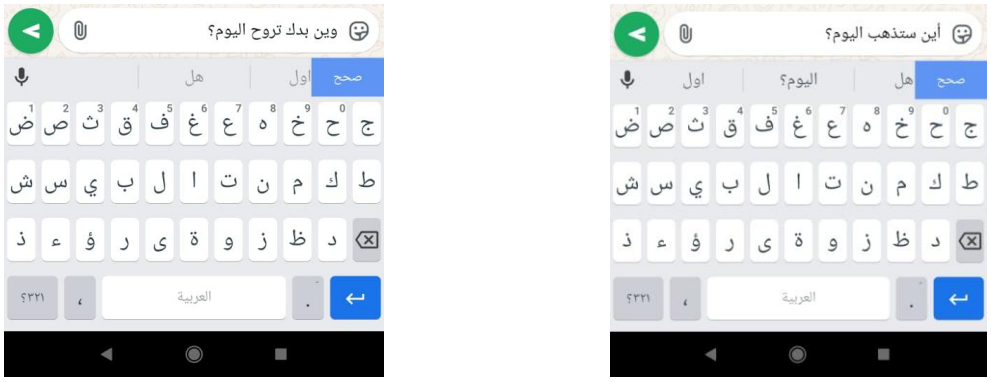

Figure 8. The AI Board translating a Jordanian dialect sentence (left) into MSA (right).

We also built a web-based portal named *Loghati* (Arabic for "my language") to offer open access to this solution. In addition to the translation feature shown in Figure 9, the portal provides references for Arabic grammar and spelling rules. It supports keyboard and microphone input, allows copying of translated text and is built using HTML, CSS, JavaScript, Bootstrap and React.js.

Figure 9. *Loghati* interface translating a sentence entered from the Jordanian dialect into MSA.

7. CONCLUSIONS

In this work, we presented an end-to-end system for translating Jordanian Arabic into MSA, correcting common linguistic errors and optionally adding diacritics. We began by collecting a large dataset of Jordanian dialect sentences (JODA), comprising diverse dialectal usages and error types. Each entry was carefully curated by Arabic-language experts, ensuring accurate MSA equivalents. To further enhance performance, we incorporated additional resources from Tashkeela, introducing synthetic spelling errors to increase the model's exposure to spelling mistake patterns and ability to correct Arabic text.

Our experiments employed the ByT5 architecture—well-suited for Arabic dialect processing due to its byte-level input handling—to achieve robust translation and correction. Through systematic fine-tuning of hyperparameters, we identified a tuned combination that improved BLEU scores on the JODA test subset by 3% over a baseline system. Furthermore, integrating the error-injected Tashkeela dataset enhanced the model's generalization, as evidenced by significant improvements in CER across various benchmark test sub-sets.

Beyond quantitative metrics, manual reviews revealed that the model's output often matched or closely approximated expert translations, underscoring its practical effectiveness. Finally, we made the resulting models accessible via a custom keyboard and a web portal, thus offering user-friendly solutions that expand the reach and impact of this research. These solutions will first be introduced in pilot scenarios to collect user feedback, enabling further refinement before a wider public launch.

Our approach, trained on JODA, the largest mono-dialect corpus, achieves the highest reported BLEU for Levantine dialects, outperforming prior Arabic-dialect-to-MSA systems. Nevertheless, it can mishandle number/gender ambiguities and code-mixed Arabic-English inputs, pointing to the need for richer data and explicit ambiguity modeling. Tests on other Levantine samples show reasonable cross-dialect transfer, but reduced accuracy with dialect-specific or French-derived terms, underscoring the need for further adaptation to other Levantine varieties.

One avenue for future research is to explore larger, more advanced ByT5 or similar transformer-based models. Increasing model parameters could enhance their capacity to capture a broader range of linguistic nuances, especially when trained on significantly expanded datasets.

While large models often produce superior results, they may be too resource-intensive for deployment on mobile devices with limited computational capabilities. A natural extension is to investigate smaller, more efficient architectures, employing techniques, like model distillation or quantization, to reduce size and inference time. This would facilitate on-device processing, ensuring offline usability and faster, more personalized performance.

Currently, we rely on two separate models whenever the corrected text also needs diacritization. However, modern-language models are powerful enough to handle multiple tasks within a single

architecture—one task for correction only and another task for both correction and diacritization. This approach eliminates the need to chain two distinct models, which will reduce latency. Future work could integrate the developed translation capabilities into Arabic chatbots [45] to enable them to automatically understand and translate user inputs from dialectal Arabic into MSA, thereby enhancing their generality and linguistic accuracy.

ACKNOWLEDGEMENTS

The authors would like to thank the Jordanian Scientific Research and Innovation Support Fund for their support of this work. We are deeply grateful to data scientists Boshra Sadder, Raghda Hasan and Shorouq AlAwawdeh for their meticulous efforts in collecting and processing the JODA dataset. Additionally, we sincerely thank Arabic-language experts Dalal Alzuheiri, Hussein Adwan, Sondos Marian and Islam Eid for providing the translations of the collected Jordanian text into MSA. We also extend our thanks to application developers Abrar Samara, Eman AbuKhater and Maysa Al-Hwayan for their valuable contributions.

This work has been carried out during sabbatical leave granted to Gheith Abandah from the University of Jordan during the academic year 2024/2025.

REFERENCES

[1] G. Khan, M. P. Streck and J. C. Watson, The Semitic Languages: An International Handbook, vol. 36, Walter de Gruyter, 2011.

[2] G. Abandah, M. Khedher, W. Anati, A. Zghoul, S. Ababneh and M. S. Hattab, "The Arabic Language Status in the Jordanian Social Networking and Mobile Phone Communications," Proc. of the 7th Int'l Conf. on Information Technology (ICIT 2015), pp. 449–456, 2015.

[3] B. Min et al., "Recent Advances in Natural Language Processing via Large Pre-trained Language Models: A Survey," ACM Computing Surveys, vol. 56, no., pp. 1–40, 2023.

[4] S. Castilho et al., "Is Neural Machine Translation the New State of the Art?" The Prague Bulletin of Mathematical Linguistics, vol.108, no. 1, pp. 109–120, 2017.

[5] M. S. H. Ameur, F. Meziane and A. Guessoum, "Arabic Machine Translation: A Survey of the Latest Trends and Challenges," Computer Science Review, vol. 38, DOI: 10.1016/j.cosrev.2020.100305, 2020.

[6] H. Wang et al., "Progress in Machine Translation," Engineering, vol. 18, pp. 143–153, 2021.

[7] I. Rivera-Trigueros, "Machine Translation Systems and Quality Assessment: A Systematic Review," Language Resources and Evaluation, vol. 56, no. 2, pp. 593–619, 2022.

[8] G. Doğru, "Statistical Machine Translation Customization between Turkish and 11 Languages," TransLogos Translation Studies J., vol. 3, no. 1, pp. 98–121, 2020.

[9] I. T. Khemakhem, S. Jamoussi and A. B. Hamadou, "Improving English-Arabic Statistical Machine Translation with Morpho-syntactic and Semantic Word Class," Int'l J. of Intelligent Systems Technologies and Applications, vol. 19, no. 2, 172, 2020.

[10] C. España-Bonet and M. R. Costa-Jussà, "Hybrid Machine Translation Overview," Part of the Book Series: Theory and Applications of Natural Language Processing (NLP), pp. 1–24, Springer, 2016.

[11] J. Zakraoui, M. Saleh, S. Al-Maadeed and J. M. AlJa'am, "Evaluation of Arabic to English Machine Translation Systems," Proc. of the 2020 11th IEEE Int'l Conf. on Information and Communication Systems (ICICS), pp. 185–190, Irbid, Jordan, 2020.

[12] R. Dabre, C. Chu and A. Kunchukuttan, "A Survey of Multilingual Neural Machine Translation," arXiv, [Online], Available: https://arxiv.org/abs/1905.05395v1, 2019.

[13] S. C. Siu, "Revolutionizing Translation with AI: Unravelling Neural Machine Translation and Generative Pre-trained Large Language Models," SSRN Electr. J., DOI: 10.2139/ssrn.4499768, 2023.

[14] J. Guo, Z. Zhang, L. Xu, B. Chen and E. Chen, "Adaptive Adapters: An Efficient Way to Incorporate BERT into Neural Machine Translation," IEEE/ACM Trans. on Audio Speech and Language Processing, vol. 29, pp. 1740–1751, 2021.

[15] X. Wu, Y. Xia, J. Zhu, L. Wu, S. Xie and T. Qin, "A Study of BERT for Context-aware Neural Machine Translation," Machine Learning, vol. 111, no. 3, pp. 917–935, 2022.

[16] C. Raffel et al., "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer," J. of Machine Learning Research, vol. 21, no. 140, pp. 1–67, 2020.

[17] M. R. Costa-Jussà et al., "Scaling Neural Machine Translation to 200 Languages," Nature, vol. 630, no. 8018, pp. 841–846, 2024.

[18] T. B. Brown et al., "Language Models are Few-shot Learners," Advances in Neural Information Processing Systems, vol. 33, pp. 1877–1901, 2020.

[19] J. Devlin, M. W. Chang, K. Lee and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," Proc. of the 2019 Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 4171–4186, 2019.

[20] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser and I. Polosukhin, "Attention Is All You Need," Adv. Neural Inf. Process. Syst., vol. 30, pp. 5998–6008, 2017.

[21] L. Xue et al., "ByT5: Towards a Token-free Future with Pre-trained Byte-to-Byte Models," Trans. of the Association for Computational Linguistics, vol. 10, pp. 291–306, 2022.

[22] B. Al-Rfooh, G. Abandah and R. Al-Rfou, "Fine-Tashkeel: Fine-tuning Byte-level Models for Accurate Arabic Text Diacritization," Proc. of the 2023 IEEE Jordan Int'l Joint Conf. on Electrical Engineering and Information Technology (JEEIT), pp. 199–204, 2023.

[23] F. Alzamzami and A. E. Saddik, "OSN-MDAD: Machine Translation Dataset for Arabic Multi-dialectal Conversations on Online Social Media," arXiv: 2309.12137, 2023.

[24] E. Nagoudi, A. Elmadany and M. Abdul-Mageed, "AraT5: Text-to-Text Transformers for Arabic Language Generation," Proc. of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 628-647, 2022.

[25] A. Slim and A. Melouah, "Low Resource Arabic Dialects Transformer Neural Machine Translation Improvement through Incremental Transfer of Shared Linguistic Features," Arabian J. for Science and Engineering, vol. 49, no. 9, pp. 12393–12409, 2024.

[26] L. H. Baniata, S. Park and S. B. Park, "A Multitask-based Neural Machine Translation Model with Part-of-Speech Tags Integration for Arabic Dialects," Applied Sciences, vol. 8, no. 12, pp. 2502, 2018.

[27] T. Alimi, R. Boujebane, W. Derouich and L. Belguith, "Fine-Tuned Transformers for Translating Multi-dialect Texts to Modern Standard Arabic," Int'l J. of Cognitive and Language Sciences, vol. 18, no. 11, pp. 679-684, 2024.

[28] S. Kchaou, R. Boujelbane and L. Hadrich-Belguith, "Parallel Resources for Tunisian Arabic Dialect Translation," Proc. of the 5th Arabic Natural Language Processing Workshop, pp. 200–206, Barcelona, Spain, 2020.

[29] S. Kchaou, R. Boujelbane and L. Hadrich-Belguith, "Hybrid Pipeline for Building Arabic Tunisian Dialect-Standard Arabic Neural Machine Translation Model from Scratch," ACM Trans. on Asian and Low-Resource Language Information Processing, vol. 22, no. 3, pp. 1–21, 2022.

[30] M. A. Faheem, K. T. Wassif, H. Bayomi and S. M. Abdou, "Improving Neural Machine Translation for Low-resource Languages through Non-parallel Corpora: A Case Study of Egyptian Dialect to Modern Standard Arabic Translation," Scientific Reports, vol. 14, no. 1, 2024.

[31] T. Zerrouki and A. Balla, "Tashkeela: Novel Corpus of Arabic Vocalized Texts, Data for Auto Diacritization Systems," Data Brief, vol. 11, pp. 147–151, 2017.

[32] G. Abandah and A. Abdel-Karim, "Accurate and Fast Recurrent Neural Network Solution for the Automatic Diacritization of Arabic Text," Jordanian Journal of Computers and Information Technology (JJCIT), vol. 6, no. 2, pp. 103–121, 2020.

[33] R. Younis and G. Abandah, "Automatic Diacritization of Arabic Text and Poetry Using Pretrained Byte-to-Byte Language Models and Multiphase Training," Proc. of the 2025 1st Int'l Conf. on Computational Intelligence Approaches and Applications (ICCIAA), pp. 1-6, Amman, Jordan, 2025.

[34] G. Abandah, "Jordanian Dialect Dataset," GitHub, [online], Available: https://github.com/Gheith-Abandah/JODA, 2025.

[35] I. Alsarsou, E. Mohamed, R. Suwaileh and T. Elsayed, "DART: A Large Dataset of Dialectal Arabic Tweets," Proc. of the 11th Int'l Conf. on Language Resources and Evaluation (LREC-2018), Miyazaki, Japan, 2018.

[36] C. Qwaider, M. Saad, S. Chatzikyriakidis and S. Dobnik, "Shami: A Corpus of Levantine Arabic Dialects," Proc. of the 11th Int'l Conf. on Language Resources and Evaluation (LREC-2018), Miyazaki, Japan, 2018.

[37] D. Alzuheiri, H. Adwan, G. Abandah and Y. Hamdan, "Converting Colloquial to Classical Arabic as within the Project Developing Applications to Correct Jordanian Spoken Arabic to Proper Language Using Machine Learning Techniques," Int'l J. on Islamic Applications in Computer Science and Technology (IJASAT), vol. 12, no. 4, 2024.

[38] A. Fadel, O. Oueslati, H. Gahbiche and R. Shafik, "Neural Arabic Text Diacritization: State of the Art Results and a Novel Approach for Machine Translation," Proc. of the 6th Workshop on Asian Translation, pp. 215-225, Hong Kong, China, 2019.

[39] A. Abdel Karim and G. Abandah, "On the Training of Deep Neural Networks for Automatic Arabic-text Diacritization," Int'l J. of Advanced Computer Science and Applications, vol. 12, no. 8, 2021.

[40] M. Khaleel and G. Abandah, "Efficient Stochastic Error Injection for Optimizing Large Language Models in Arabic Spelling Correction," Proc. of the 2025 IEEE Int'l Conf. on New Trends in Computing Sciences (ICTCS), pp. 505-510, Amman, Jordan, 2025.

[41] G. Abandah, A. Suyyagh and M. Z. Khedher, "Correcting Arabic Soft Spelling Mistakes Using BiLSTM-based Machine Learning," Int'l J. of Advanced Computer Science and Applications, vol. 13, no. 5, 2022.
[42] P. Phakmongkol and P. Vateekul, "Enhance Text-to-Text Transfer Transformer with Generated Questions for Thai Question Answering," Applied Sciences, vol. 11, no. 21, 2021.
[43] OpenBoard Team, "OpenBoard," GitHub, [Online], Available: https://github.com/openboard-team/openboard, 2022
[44] KeyboardKit, "KeyboardKit," [Online], Available: https://keyboardkit.com/, 2023.
[45] B. Sadder, R. Sadder, G. Abandah and I. Jafar, "Multi-domain Machine Learning Approach of Named Entity Recognition for Arabic Booking Chatbot Engines Using Pre-Trained Bidirectional Transformers," Jordanian Journal of Computers and Information Technology (JJCIT), vol. 10, no. 1, pp. 1–16, 2024.

ملخص البحث:

تُعالج هذه الورقة التّحدّي المتمثل في إنتاج ترجمةٍ دقيقةٍ من العربية الأردنية إلى العربية الفصحى الحديثة (MSA) مع تصويب الأخطاء اللّغوية الشّائعة. وعلى الرّغم من أنّ العربية الفصحى الحديثة هي الصّيغة الرّسمية للتّواصل باللّغة العربية، فإنّ الانتشار الواسع للّهجات المحلّية في وسائل التّواصل الاجتماعي إلى جانب التّفاعلات اليومية نجم عنها انتشار نُصوص تُعاني من أخطاء في التّهجئة وعيوبٍ قواعدية.

وللتغلب على هذه التّحديات، نقدّم نظاماً مبتكراً مبنياً على مجموعة بياناتٍ بالعربية الأردنية (JODA) تتألف من (59135) جُملة، إلى جانب مجموعة البيانات (تشكيلة) المعدّلة من خلال حقن الأخطاء الصناعية. ونقوم بتوظيف نموذجً لغويٍ ضخم مُدرَّبٍ مُسبقاً يقبل نُصوص على مستوى البايْت تجعله قابلاً للتّكيُّف للتّبايُنات المتعلّقة بالتّهجئة في اللّهجات المحلّية المختلفة والتّعقيدات التّركيبية المختلفة في اللهجات العربية. ويظهر نظامنا من خلال التّجارب العملية تحسين معدّل الخطأ في النّصّ عبر تقليل ذلك المعدّل بفعل معايرة النّظام على مجموعات البيانات المذكورة، كما اتّضح أنّ النّظام المقترح يُنتج ترجماتٍ دقيقةً في معظم الحالات. وتبين أيضاً أنّ النّظام مَدار البحث يُشكّل حلّاً ناجعاً لملايين المتحدّثين بالعربية السّاعين للحصول على نصوصٍ دقيقة بالعربية الفصحى الحديثة.

والنّظام المقترح في وضعه الرّاهن مُقتصر على ترجمة النّصوص من العربية الأردنية، وتتّجه مساعي البحث المستقبلية إلى توسيع استخدامه ليغطّي التّرجمة من لهجاتٍ عربية أخرى.