Proceedings of the 20th Conference on Computer Science and Intelligence Systems (FedCSIS) pp. 451–460
DOI: 10.15439/2025F4862
ISSN 2300-5963 ACSIS, Vol. 43

AraXLM: Evaluating Arabic Diacritization Tools for Cross-Language Plagiarism Detection

Mona Alshehri
0000-0002-4193-6230
Department of Computer Science, King Abdulaziz University, Jeddah, Saudi Arabia
Department of Informatics, University of Sussex, Brighton, United Kingdom
Email: ma2250@sussex.ac.uk

Natalia Beloff
0000-0002-8872-7786
Department of Informatics, University of Sussex, Brighton, United Kingdom
Email: n.beloff@sussex.ac.uk

Martin White
0000-0001-8686-2274
Department of Informatics, University of Sussex, Brighton, United Kingdom
Email: m.white@sussex.ac.uk

***Abstract*—In recent years, plagiarism detection systems have evolved from basic lexical matching and n-gram overlap methods to Deep Learning (DL) models capable of capturing semantic relationships between texts. While these DL-based approaches have achieved notable success across various languages, their effectiveness in Arabic remains limited due to inherent linguistic ambiguities, particularly the omission of diacritical marks. This absence hinders accurate semantic interpretation and limits the ability of models to detect paraphrased or semantically obfuscated content in Arabic texts. This paper presents an evaluation of Arabic Text Diacritization (ATD) tools as the initial phase of a plagiarism detection framework designed for Arabic–English cross-lingual model text analysis (AraXLM). It describes the first stage of the framework, which focuses on assessing the performance of state-of-the-art ATD tools. An empirical analysis was conducted on six ATD models using Word Error Rate (WER), Diacritic Error Rate (DER), both with and without case endings (CE), and Bilingual Evaluation Understudy (BLEU) metrics. The results show that tools such as Shakkelha produced lower DER and high BLEU values, indicating high accuracy in diacritic restoration, while Fine-Tashkeel demonstrates the lowest WER and highest BLEU, reflecting best word-level performance. In contrast, CAMeL Tools and Mishkal display comparatively higher error rates across both metrics. These findings suggest that incorporating accurate diacritization models into Arabic NLP tasks, such as Machine Translation (MT) and Plagiarism Detection (PD), improves text normalisation and the quality of semantic embeddings. Thus, the AraXLM framework, supported by effective diacritization pre-processing, enhances linguistically aware detection of plagiarism involving Arabic text, where precise semantic alignment between languages is essential.**

***Index Terms*—Deep Learning (DL), Arabic Text Diacritization (ATD), Word Error Rate (WER), Diacritic Error Rate (DER), Case Ending (CE), Machine Translation (MT), Plagiarism Detection (PD), Bilingual Evaluation Understudy (BLEU).**

I. Introduction

DIACRITIC marks are a fundamental characteristic of the Arabic language. These symbols, placed above or below letters, indicate pronunciation and disambiguate meaning [1], [2]. Each mark has a specific position, Unicode representation, and functional role, influenced by syntactic and phonetic context. Diacritics are essential for accurate word interpretation, enhancing reading fluency, improving speech recognition systems, and facilitating natural language understanding. However, most Arabic texts omit these marks, which introduces semantic ambiguity and creates homographic conflicts [3].

A. Problem Statement

Recently, NLP techniques have started to include diacritical marks in their approaches to enhance language processing performance [4]. Most techniques used in plagiarism detection for the Arabic language preprocess the source document by removing the diacritic marks [5],[6]. The performance of these tests has still not improved due to pre-processing the document and not using semantic-based algorithms. The absence of diacritical marks in Arabic text introduces ambiguity that negatively impacts NLP tasks such as machine translation and plagiarism detection. Despite advances in deep learning and NLP, many systems either ignore diacritics or preprocess them away, limiting semantic accuracy and cross-lingual performance [7], [8]. Therefore, linguistic knowledge, such as vowel marks in Arabic language, needs to be addressed and evaluated in NLP tasks for enhancing semantic alignment across languages.

B. Research Questions

In order to address the issues that have been identified of applying and evaluating the ATD models, we will commence with these research sub-questions:

1) what is the most effective ATD tool for integrating into our proposed framework (AraXLM).
2) What issues are identified when conducting our experimental study, particularly for Arabic language?

The main research question is: given 2 sentences in different language, S1(English) and S2(Arabic), is S2 plagiarized from S1?

In order to answer this broad question, we defined the sub-questions that help to provide a complete answers to the overall research question. Our previous proposed framework was designed and presented for cross-language Plagiarism Detection (PD), including ATD as a linguistic feature [9]. The aim of our research contributes to detect the similarity between Arabic and English sentence pairs, including the diacritization process for Arabic sentences (see Figure 1). Diacritization is the process of adding diacritical marks (such as fatḥah, ḍammah, kasrah, shaddah, tanwīn, and sukūn) to unvowelled Arabic letters in order to indicate correct pronunciation and clarify meaning. This process is essential for various Natural Language Processing (NLP) tasks, such as MT, as it reduces the ambiguity caused by the absence of these marks. This paper presents the results of evaluating several tools concerning the accuracy of ATD.

In summary, our experimental design evaluates six Arabic diacritization tools using publicly available corpora. This study aims to identify the most effective Arabic ATD models to integrate into our proposed framework [9]. The evaluation employs three standard metrics: Word Error Rate (WER), Diacritic Error Rate (DER), and Bilingual Evaluation Understudy (BLEU). This multi-metric analysis provides a comprehensive comparison and deeper insight into the types of errors produced by each tool, with enhancing semantic accuracy in ML-based models. The results indicate that ML-based approaches achieve lower DER and higher BLEU scores, reflecting improved diacritic restoration and closer alignment with reference texts. However, these models produced higher WER, highlighting challenges associated with word-level tokenisation in Arabic, despite their character-level and semantic accuracy strengths.

II. Background

Arabic diacritization is a task in Arabic natural language processing (ANLP). These diacritics ensure correct pronunciation, disambiguation, and improved MT. However, the complex morphology of Arabic and the high level of ambiguity in unvowelled text make diacritization challenging task [10]. Over the years, researchers have developed various approaches to address this task, which can be categorized into three main types: rule-based, machine learning-based, and hybrid-based methods. Each approach employs different methodologies to overcome the challenges and develop accuracy in this field.

A. Rule-Based Diacritization Tools

Rule-based approaches apply diacritic marks based on predefined linguistic features (e.g., morphology, syntax, lexicon) [11]. In these tools, sequences of Arabic letters are mapped to specific diacritic categories through encoded rules derived from morphological analysers, syntactic parsers, and lexicon lookup tables [12]. However, pure rule based systems can fail with out of vocabulary (OOV) words and ambiguous contexts where multiple diacritic patterns may be valid. Moreover, validating the hundreds of rules and large lexicons required for comprehensive coverage can be time-consuming, error prone, and challenging to scale as the language evolves [12]. In contrast, rule-based diacritization tools achieve good result on words or structures explicitly defined within their linguistic frameworks, by applying deterministic rules derived from formal Arabic grammar.

B. Machine Learning Based Diacritization Tools

ML-based approaches for Arabic diacritization leverage neural networks (NNs) to learn and predict diacritic marks from text patterns. These approaches require datasets for training the model on diacritizing text pattern. ML algorithms can be categories to different types of Neural Networks (NNs): Artificial Neural Networks (ANNs), Convolution Neural Network (CNNs), Recurrent Neural Network (RNNs), and Transformer Neural Network (TNNs). Each type is used to process specific data and extract features that are relevant to the task.

FNNs are the simplest form of neural networks, consisting of input, interconnected hidden layers, and output layers. They process input data in a forward direction without cycles or loops [13]. FNNs have been used as baseline models for Arabic diacritization for capturing local patterns without requiring morphological pre-processing [14]. In ANNs types, back-propagation concept adjusts each weight in the layers. This step is done to eliminate the error (Loss) between the predicted output and the true target [15]. Thus, FNN is a subclass of ANN because the data flows on forward propagation only. However, there is no feedback loop in each node as outputs to inputs during inference. There-

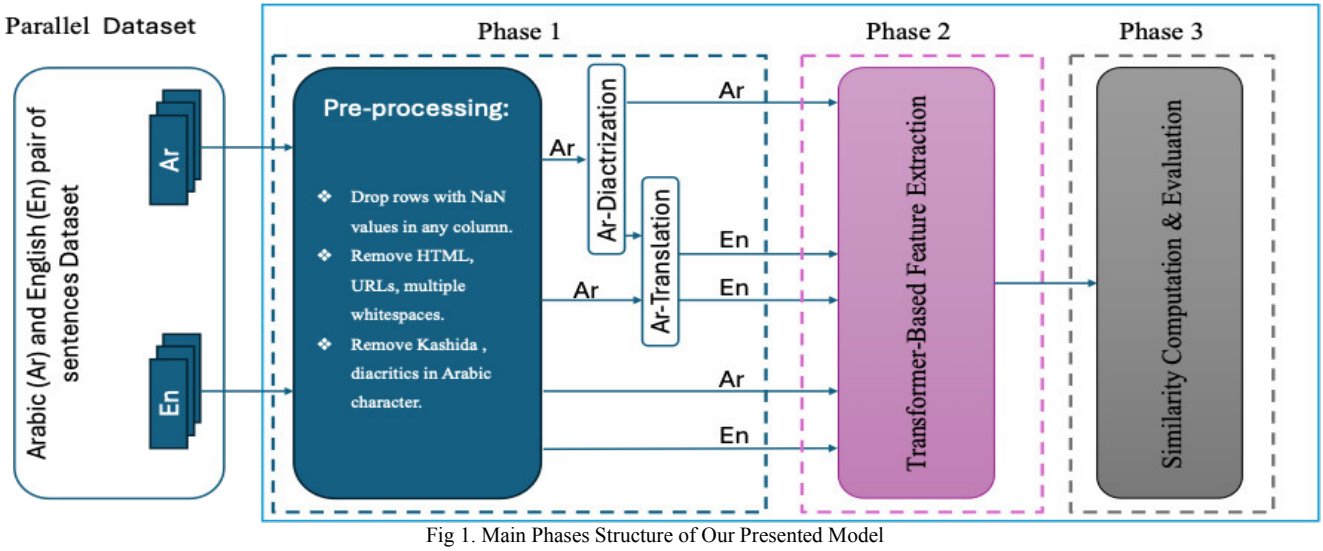


Fig 1. Main Phases Structure of Our Presented Model

fore, their ability to capture syntactic diacritization, such as case endings, is limited.

CNNs apply convolutional filters over embedding vectors to detect local patterns and have been used in image and text processing tasks [16]. In Arabic diacritization, CNNs have been employed to recognize diacritic marks in images of Arabic text, such as in Optical Character Recognition (OCR) systems, rather than for direct sequence labelling of plain text.

RNNs enable each hidden unit to receive inputs from both the current and previous time steps, thereby allowing the network to capture temporal dependencies within sequential data [17]. By combining the current input with the previous hidden state, the network updates the current hidden state, which is then used to generate the output. This sequential modelling allows RNNs to predict diacritics for each character in context [18]. However, RNNs encounter difficulties when processing long sequences due to the gradient problem. The gradient represents the scale of change used to update the network during training. Over extended sequences, gradients can either vanish (become very small) or explode (grow very large), depending on the activation functions employed [19]. This occurrence causes the network to forget earlier information in the sequence, thereby limiting its ability to learn long-range dependencies.

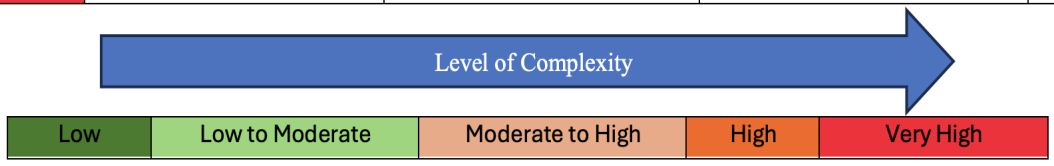
The Transformer architecture was introduced to process data in parallel by replacing recurrence and convolution with positional encoding and self-attention mechanisms [20]. In this architecture, the encoder generates contextualized representations of the input sequence, which are used by attention mechanisms in the decoder to focus on relevant parts of the input during output generation. The transformer architecture can consist of encoder and/or decoder layers. Recently, TNNs have been employed in ATD models to capture both local and global contextual information in parallel [21]. As a result, TNNs models have transformed NLP by introducing attention mechanisms that address key limitations of earlier approaches [22]. TABLE I shows the main differences between neural network approaches types. For example, TNN leverage attention mechanism for tasks requiring long sequence dependency, such as NLP task. The emergence of NN types with ATD have driven the advancement of structure models.

C. *Hybrid Based Diacritization Tools*

Hybrid-based diacritization approaches combine rule-based and machine learning (ML) methods to leverage both linguistic knowledge and contextual information in text. These tools apply ML components to handle lexical disambiguation and rule-based frameworks for syntactic case endings, capitalizing on the strengths of both paradigms. [23]. Recent research has adopted hybrid approaches for Arabic diacritization, combining rule-based linguistic knowledge with ML techniques to enhance accuracy and address the

TABLE I.
OVERVIEW OF NEURAL NETWORKS AND THEIR FEATURES

NN Type	Complexity	Data Type	Task	Feature	Authors
FNN		Structured, tabular data	Traditional ML (Classification, Regression)	One Directional (forward) Connected Layer	[13], [14]
ANN		Structured, tabular data	Traditional ML (Classification, Regression)	Fully Connected Layer	[15]
CNN		Spatial Data (Image, Videos)	Object Detection, Image recognition, Image Processing Computer Vision	Convolution Layer, Pool Layer, Flattening Layer and Fully Connected Layer.	[16]
RNN		Sequential Data (Text, Time Series)	Weather (time series) Forecasting, Speech Recognition	Recurrent Layers with hidden states that maintain a memory of previous inputs	[17], [18]
TNN		Parallelized Data (Text, image, Audio)	NLP, Text Summarization, Question Answering, Translation, PD	Attention Mechanism	[19], [20]



complexities inherent in the language [24], [25]. However, these systems, incorporating ML components, remain under development for requirement of large and good quality annotated corpora for effective training [26], [27]. Therefore, integrating rule-based and ML models require alignment and coordination to ensure consistency and resolve conflicts between these components when deployed together in hybrid tools.

III. Experimental Setup

In ATD, several studies have established systematic benchmarks frameworks, based on manually curated and expertly validated datasets, as the gold-standard for evaluating Arabic diacritization tools [28], [29]. To determine the most effective ATD tools for Phase 1 of the proposed framework [9], we conducted experiment on six diacritization tools. This experiment focused on evaluating their ability to restore diacritic marks on Arabic sentences for subsequent processing. We employed a repeated measures (within-subjects) design [30], in which each sentence (experimental unit) was processed by every diacritization tool (treatment) in the same order. This setup was designed to test the hypothesis that diacritization enhances the semantic clarity of Arabic text and aligns it more closely with its English counterparts for the next Phase.

A. Experiment Design

The dataset used in this experiment was designed by Alasmary in 2024 and consists of 742 sentences collected from internet sources covering various topics such as science, art, sport, and culture [28]. The author introduced this benchmark, known as the Character-based Arabic Tashkeel Transformer (CATT) dataset, to evaluate different ATD tools. All sentences were manually diacritized by native Arabic speakers and subsequently validated by experts to establish a gold-standard reference.

Preparation of the Arabic dataset for assessment involved several steps to ensure data quality. For example, text normalization was performed to eliminate noise caused by foreign symbols (non-Arabic characters). Second, numerical digits were converted from their digital (numeric) form into textual form, thereby ensuring the dataset is fully normalized and ready for use without additional pre-processing. This conversion is used during preparation, because Arabic numbers are represented in different numeral systems depending on regional preferences and historical influences [31]. TABLE II shows three Arabic numeral systems may appear in Arabic texts. These pre-processing steps ensure the dataset is standardised and ready for direct use without extra modification.

TABLE II.
NUMERL SYSTEMS IN ARABIC LANGUAGE [30]

System	Digit
1-Arabic numerals-European	0 1 2 3 4 5 6 7 8 9
2-Arabic-Indic	٩ ٨ ٧ ٦ ٥ ٤ ٣ ٢ ١ ٠
3-Eastern Arabic-Indic	۰ ۱ ۲ ۳ ۴ ۵ ۶ ۷ ۸ ۹

B. Evaluation Metrics

We conducted our experiment using six Arabic diacritization tools and evaluated their performance by calculating the WER and the DER metrics. However, Arabic grammar requires different evaluation rules for diacritics on the final letter of words [32] as follow:

- Full Diacritization, including case endings (CE).
- Core-Word Diacritization, excluding CE.

Therefore, WER and DER can be reported with and without case endings. Each rule is used to provide distinct insights into performance measurements. For example, error rates with CE measures overall performance, including syntactic case endings that influence sentence meaning. Whereas error rates without CE measures the model's ability to restore word-internal morphology.

The `diacritization_evaluation`[1] Python package is used to calculate DER and WER metrics, relying on a predefined set of Arabic characters stored in its configuration files. The DER is computed by the file `der.py` as the percentage of mismatched diacritics-those that differ from the reference gold-standard text file over the total number of diacritic comparisons (matches and mismatches), rounded to two decimal places (see Equation 1). Similarly, the WER is calculated by the file `wer.py` as the percentage of mismatched words-those that differ from the reference gold-standard text file over the total number of word comparisons (matches and mismatches) (see Equation 2).

$$DER = \frac{Number\ of\ mismatched\ diacritics}{matched + mismatched} \quad (1)$$

$$WER = \frac{Number\ of\ mismatched\ words}{matched + mismatched} \quad (2)$$

Moreover, the BLEU (Bilingual Evaluation Understudy) metric is used to evaluate the quality of ATD models alongside WER and DER, providing a comprehensive assessment of diacritization performance. BLUE metric developed for machine translation (MT) evaluation, but it can be applied to measure the output of Arabic diacritization tools [33]. There are two types of BLEU metrics: word-level and character-level. In word-level BLEU, n-gram overlaps (default n=4) between the gold-standard dataset and the output file from the ATD models are computed, with mismatched diacritic marks counted as errors. Character-level BLEU, on the other hand, treats each Unicode code-point as a separate token for comparison.

C. Tools Performance

The following diacritization tools were compared in the experiment: CAMel Tools [34], Farasa [27], Mishkal [35], i2Text[2] online tool, Fine-Tashkeel [29] and Shakkelha [36]. These tools and their associated Python packages were implemented using the NVIDIA A100 GPU runtime on Google

[1] https://pypi.org/project/diacritization-evaluation/#files
[2] https://www.i2text.com/diacritize-arabic-text

Colab Pro. Since the experimental setup relies on multiple packages and modules written in Python, the scripts and files used in our tests were modified to suit our needs. The modifications are detailed as follows:

- Farasa Tool: The `diacritization_evaluation` package DER and WER using a predefined set of Arabic characters. However, the output generated by Farasa included a combined diacritic not originally present in the package's character file. To ensure accurate error rate calculation, we added this new diacritic mark to the set. Notably, this mark results from the combination of Shaddah and Sukun into a single diacritic, which is not standard in Arabic but appeared during the diacritization process by the Farasa tool.
- Mishkal tool, The output text file from the Mishkal tool represents the Tanween vowel as a combination double vowel (e.g., `Fatha+Fatha`). However, Tanween and Fatha have different Unicode representations when processed as diacritic marks. To handle this, we updated the diacritization_evaluation pack-age to recognize `Fatha + Fatha` as Tanween. During testing, Mishkal produced other compinged marks, such as `Fatha + Kasra`, `Kasra + Kasra`, and `Sukun + Sukun`, which we similarly added to the package's set of recognized diacritic characters.
- Fine-Tashkeel, We tokenised the input file and set the parameter `max_new_tokens=512` when invoking `model.generate` to process texts longer than the model's default context window of 128 tokens. This tool took long time (e.g., 5 hours) to diacritize the text file.

IV. Results of The Experiment

The results obtained from applying the selected ATD models using CATT benchmark are presented below:

A. WER and DER Evaluation

TABLE III and Figure 2 presents the evaluation results using WER and DER, both with and without case endings (CE). In the context of Arabic diacritization, lower WER and DER values indicate fewer errors and therefore better tool performance.

TABLE III.
WER AND DER IN % OF EVALUATED ATD TOOLS

Diacritization Tool	Case Ending %		No Case Ending %	
	WER	DER	WER	DER
CAMeL Tools	75.85	71.71	56.84	69.86
Farasa	65.79	18.04	59.99	18.47
Mishkal	60.92	74.59	39.33	73.16
Fine-Tashkeel	35.70	67.73	30.24	67.28
i2text	61.04	21.14	39.30	17.39
Shakkelha	47.45	13.22	37.76	12.0

The results show that Shakkelha and i2text yield the lowest DER values, particularly when case endings are excluded, reflecting their effectiveness in diacritic restoration at the character level. In contrast, the generation of merged and non-standard diacritic combinations during the diacritization process affected the performance and the output quality of tools such as Mishkal and Farasa. These irregular patterns introduced inconsistencies in the evaluation and interpretation of the diacritized output, which is reflected in their elevated DER scores despite achieving moderate WER. This indicates that while token-level segmentation can be acceptable, the character-level precision required for reliable diacritic restoration remains insufficient in these tools.

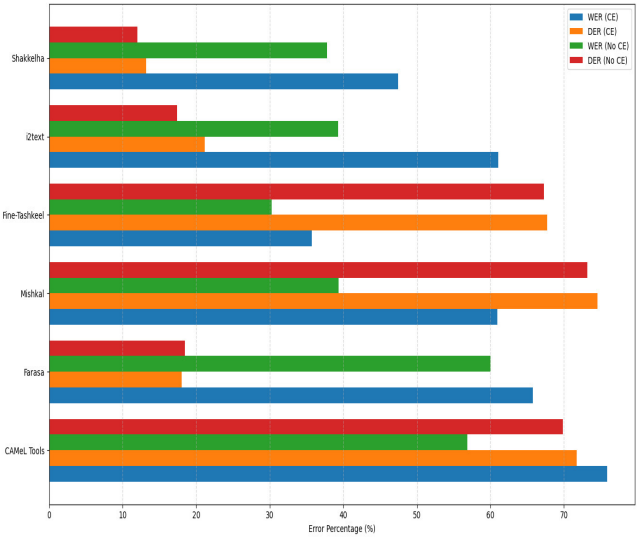


Fig 2. WER and DER by Arabic Diacritization Tool (with/without Case Endings)

Fine-Tashkeel, which leverages a fine-tuned multilingual transformer model (ByT5) with integrated tokenisation, generates fully diacritized text by assigning a diacritic to every character and applying multiple diacritic marks to some characters. This approach results in a lower WER, as words are correctly segmented. However, it leads to a higher DER, since the model introduces unnecessary or additional diacritics, increasing character-level errors. During the rendering of diacritic marks, several issues were introduced (see TABLE IV), including incorrect diacritic substitutions (e.g., fatḥah replaced with kasrah), and the addition of unintended punctuation such as parentheses or colons. These errors highlight the need for enhanced morphological awareness and post-processing techniques to improve the accuracy of diacritic restoration and ensure better alignment with gold-standard references.

TABLE IV.
IDENTIFIED ISSUES IN FINE-TASHKEEL' ATD TOOL

Gold-Standard Reference	Fine-Tashkeel Prediction
اِنْتَقَلَتْ إِلَى رَحْمَةِ اللهِ تَعَالَى	انْتَقَلَتْ إِلَى رَحْمَةِ اللَّهِ تَعَالَى
فَعَالِيَّتَها فِي تَقْلِيلِ نِسَبِ الإِصابَةِ وَلَيْسَ مَنْعُها بِالْكامِلِ	فِعَالِيَّتَهَا فِي تَقْلِيلِ نَسَبِ الْإِصَابَةِ وَلَيْسَ مَنْعُهَا بِالْكَامِلِ(
التَّجْرِبَةِ مَرَّةً أُخْرَى	التَّجْرِبَةِ مَرَّةً أُخْرَى:

On the other hand, CAMeL Tools tokenises each word independently and applies diacritics using the `mle.disambiguate` function. However, it introduces additional diacritical marks, such as maddah and hamza, that were not present in the original undiacritized input. TABLE V compares the output of CAMeL Tools with the gold-standard diacritized reference. Each row shows a separate sentence from the dataset, highlighting the performance of the tool in restoring Arabic diacritics. It demonstrates inconsistent performance when handling morphological structures and complex verb forms. The model shows orthographic overcorrections, including the unnecessary insertion of the dagger alif, duplication of shaddah, and frequent mismatches in grammatical case endings. As a result, both its WER and DER are high due to the inclusion of these extraneous diacritics, which negatively impact alignment with the reference text.

TABLE V.
IDENTIFIED ISSUES IN CAMEL TOOLS' ATD

Gold-Standard Reference	CAMeLTools Prediction
نَجَحَ مُسْتَشْفَى المَلِكِ سَلْمانَ عُضْوٌ تَجَمُّع الرِّياضِ الصِحِّيّ الأَوَّلِ فِي تَثْبِيتِ كَسْرِ مُضاعَفٍ وَمُهَشَّمٍ لِعَضُدٍ مُصابَةٍ إِثْرَ تَعَرُّضِها لِلسُّقُوطِ	نَجَحَ مُسْتَشْفَى المَلِكُ سَلْمان عُضْوٌ تَجْمَع الرياضِ الصِحِّيّ الأَوَّلِ فِي تَثْبِيتُ كَسْرِ مُضاعَف وَمُهَشَّم لِعَضُد مُصابَةٌ أَثْرَ تَعَرُّضِها لِلسُّقُوطِ
وَاخْتُتِمَ المِرانُ بِمُناوَرَةٍ عَلَى جُزْءٍ مِنَ المَلْعَبِ	وَاخْتَتَمَ المَرْآن بِمُناوَرَة عَلَى جُزْء مِن المَلْعَبِ
إِذا طَلَعَتِ الجَوْزاءُ فَامْلَأِ الحَوزاءَ	إِذا طَلَعَتْ الجَوْزاء فَأَمْلا الحوزاء
وَأَحالَتْهُمْ إِلَى النِيابَةِ العامَّةِ	وَأَحالَتْهُم إِلَى النِيابَةُ العامَّة
غَفَرَ اللهُ لَها وَرَحِمَها	غَفَرَ اللهُ لَها وَرَحِمَها

B. BLEU-Based Evaluation

To provide a more comprehensive evaluation of diacritization quality, beside WER and DER, this study incorporates the Bilingual Evaluation Understudy (BLEU) metric. BLEU offers a corresponding perspective by measuring the degree of n-gram overlap between the predicted and reference outputs.

As represented in TABLE VI and Figure 3, BLEU scores were calculated at both the word level and character level to capture accuracy across different linguistic features. In addition, we employed `camel_tools.tokenizers` to reflect token-based alignment accuracy in Arabic text. This BLEU metrics evaluation enables a more understanding of how effectively each tool restores diacritics in ways that are compatible with downstream token-based NLP tasks. The word-level BLEU scores remain low across all tools, reveal-ing penalties for tokenisation mismatches and sensitivity to n-gram overlap. In contrast, character-level BLEU scores are higher, as this metric treats each Unicode character as a token, making it more suitable for assessing diacritic placement precision at the subword level. Shakkelha performs well in character-level and tokeniser-based BLEU, suggesting good subword precision and compatibility with token-based Arabic NLP frameworks. In contrast, CAMeL Tools shows the lowest scores across all BLEU metrics, consistent with its poor performance in WER/DER due to added and misaligned diacritics within sentence structure.

TABLE VI.
BLEU IN % OF EVALUATION ATD TOOLS

Tool	Diacritized %		Diacritized with camel_tools.tokenizers
	Word-level BLEU	Char-level BLEU	
CAMeL Tools	2.97	66.95	2.94
Farasa	7.94	71.55	7.87
Mishkal	13.01	78.44	13.27
Fine-Tashkeel	39.36	86.80	38.74
i2text	12.98	78.32	13.24
Shakkelha	25.07	81.72	24.69

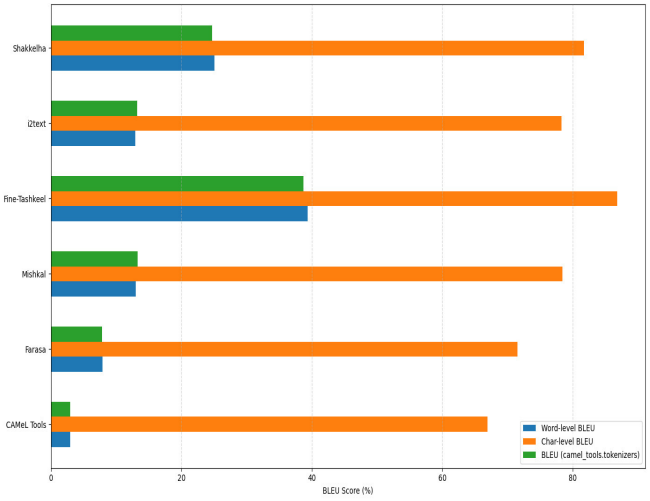


Fig 3. Comparison of BLEU Scores for Arabic Diacritization Tools

We conducted a Unicode-level analysis comparing the predicted outputs against the gold-standard references. As illustrated in TABLE VII. For instance, this is one diacritized pair:

Original no-Diacritic Marks:
"وأكدت الحوسني في إفادتها أن اللقاحات المضادة لفيروس كورونا أثبتت فعاليتها في تقليل نسب الإصابة وليس منعها بالكامل"

Gold-Standard Reference:
"وَأَكَّدَت الحُوسَني في إفادَتها أَنَّ اللُّقاحات المُضادَّةَ لِفَيْرُوس كورونا أَثْبَتَت فَعَالِيَّتَها في تَقْليلِ نِسَبِ الإِصابَةِ وَلَيْسَ مَنْعُها بِالكامِلِ"

Predicted by Fine-Tashkeel:
"وَأَكَّدَت الحَوْسَنِيُّ في إِفَادَتِهَا أَنَّ اللُّقَاحَاتِ المُضَادَّةَ لِفَيْرُوس كورُونَا أَثْبَتَت فَعَالِيَّتَهَا في تَقْلِيلِ نَسَبِ الإِصَابَةِ وَلَيْسَ مَنْعُهَا بِالكَامِلِ"

The results showed several differences were identified at the character encoding level that impact the accuracy and evaluability of the tool's output. These irrelevant characters are introduced during token reconstruction and text formatting, and they do not reflect linguistic features. Their presence negatively affects character-level and word-level metrics. Furthermore, the predicted output included duplicated diacritics (e.g., double fathah, parenthesis, dot) and altered character spacing that increased the total Unicode code point count, which was not present in the original input or reference.

TABLE VII.
EXAMPLE OF UNICODE CHARACTER PROPRIETIES

Gold-Standard Reference	Fine-Tashkeel Prediction
وَلَيْسَ مَنْعُها بِالْكامِلِ	وَلَيْسَ مَنْعُهَا بِالْكَامِلِ)
.....	.....
و → U+0648 → ARABIC LETTER WAW	و → U+0648 → ARABIC LETTER WAW
◌َ → U+064E → ARABIC FATHA	◌َ → U+064E → ARABIC FATHA
ل → U+0644 → ARABIC LETTER LAM	ل → U+0644 → ARABIC LETTER LAM
◌َ → U+064E → ARABIC FATHA	◌َ → U+064E → ARABIC FATHA
ي → U+064A → ARABIC LETTER YEH	ي → U+064A → ARABIC LETTER YEH
◌ْ → U+0652 → ARABIC SUKUN	◌ْ → U+0652 → ARABIC SUKUN
س → U+0633 → ARABIC LETTER SEEN	س → U+0633 → ARABIC LETTER SEEN
◌َ → U+064E → ARABIC FATHA	◌َ → U+064E → ARABIC FATHA
→ U+0020 → SPACE	→ U+0020 → SPACE
م → U+0645 → ARABIC LETTER MEEM	م → U+0645 → ARABIC LETTER MEEM
◌َ → U+064E → ARABIC FATHA	◌َ → U+064E → ARABIC FATHA
ن → U+0646 → ARABIC LETTER NOON	ن → U+0646 → ARABIC LETTER NOON
◌ْ → U+0652 → ARABIC SUKUN	◌ْ → U+0652 → ARABIC SUKUN
ع → U+0639 → ARABIC LETTER AIN	ع → U+0639 → ARABIC LETTER AIN
◌ُ → U+064F → ARABIC DAMMA	◌ُ → U+064F → ARABIC DAMMA
ه → U+0647 → ARABIC LETTER HEH	ه → U+0647 → ARABIC LETTER HEH
ا → U+0627 → ARABIC LETTER ALEF	◌َ → U+064E → ARABIC FATHA
→ U+0020 → SPACE	ا → U+0627 → ARABIC LETTER ALEF
ب → U+0628 → ARABIC LETTER BEH	→ U+0020 → SPACE
◌ِ → U+0650 → ARABIC KASRA	ب → U+0628 → ARABIC LETTER BEH
ا → U+0627 → ARABIC LETTER ALEF	◌ِ → U+0650 → ARABIC KASRA
ل → U+0644 → ARABIC LETTER LAM	ا → U+0627 → ARABIC LETTER ALEF
◌ْ → U+0652 → ARABIC SUKUN	ل → U+0644 → ARABIC LETTER LAM
ك → U+0643 → ARABIC LETTER KAF	◌ْ → U+0652 → ARABIC SUKUN
ا → U+0627 → ARABIC LETTER ALEF	ك → U+0643 → ARABIC LETTER KAF
م → U+0645 → ARABIC LETTER MEEM	◌َ → U+064E → ARABIC FATHA
◌ِ → U+0650 → ARABIC KASRA	ا → U+0627 → ARABIC LETTER ALEF
ل → U+0644 → ARABIC LETTER LAM	م → U+0645 → ARABIC LETTER MEEM
◌ِ → U+0650 → ARABIC KASRA	◌ِ → U+0650 → ARABIC KASRA
	ل → U+0644 → ARABIC LETTER LAM
	◌ِ → U+0650 → ARABIC KASRA
	(→ U+0028 → LEFT PARENTHESIS
Number of characters: 177	**Number of characters: 192**

In summary, the Pearson correlation coefficient was used to calculate the correlation for understanding the linear relationship between different error rates across Arabic diacritization tools (see Figure 4). The results indicated that WER and DER measure different aspects of model performance and do not strongly correlate with each other. WER is relevant to downstream tasks such as MT and text-to-speech, where word-level accuracy is critical, while DER provides assessment of diacritization quality, making it suitable for evaluating ATD models.These findings highlight the importance of evaluating both WER and DER separately, and demonstrate how correlation analysis can guide the selection or improvement of diacritization models.

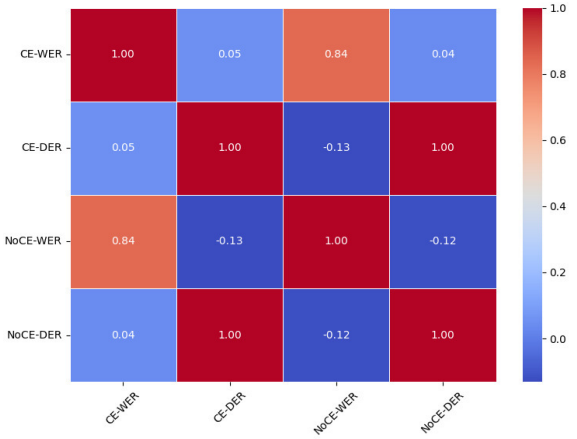


Fig 4. Pearson Correlation Matrix between WER and DER of ATD

V. EMPIRICAL EVALUATION OF FINE-TUNED TRANSFORMER MODELS FOR DIACRITIZED ARABIC-ENGLISH STS

In our proposed framework, a transformer-based approach is employed to effectively address the challenges of cross-lingual STS, with a particular focus on Arabic-English sentence pairs. The Arabic input is diacritized by Shakkelha prior to training, enabling the model to better disambiguate meanings and improve alignment with corresponding English texts. By incorporating fine-tuning into our approach, the transformer encoder is adapted to the specific semantic and linguistic characteristics of the dataset. In order to select the appropriate XLM-R variant for our model, we conducted a comparative evaluation of three XLMR-based transformer encoders by empirical experimentation. The chosen models were tested on an Arabic-English STS dataset, STS-Benchmark has 250 pairs and introduced in 2017, with Arabic sentences diacritized before and after training the models. This STS benchmark contains a variety of semantic similarity levels showing moderate similarity label balance. Then, the score in this dataset was normalized from [0 – 5] to [0 – 1] for fine-tuning. Normalizing STS scores to the [0 – 1] range is standard preparation in sentence embedding models with STS task and is fundamental for correct training and

evaluation. The dataset was randomly divided using the train_test_split function from scikit-learn, with 80% of the sentence pairs used for training and the remaining 20% reserved for development (dev) validation. This split was applied to the pre-processed and normalized STS-labelled CSV data to ensure a clear separation between training and evaluation samples. The results are summarised in TABLE VIII, which presents the Pearson correlation coefficients computed on the development (dev) set before and after fine-tuning. It shows the effect of fine-tuning five different multilingual sentence encoders on a SemEval Semantic Textual Similarity task using Arabic text that has diacritization by ATD tool. The metric used to evaluate the effectiveness of fine-tuning is the Pearson correlation coefficient (r) between the predicted similarity scores and the gold standard labels, along with p-values to test statistical significance. Pearson correlation measures the degree of linear association between the similarity scores produced by the model and the human-annotated gold-standard similarity scores. Evaluation on the dev set, rather than the training set, ensures an unbiased estimation of the model's generalization performance on unseen data and avoid the risk of overfitting or performance inflation due to exposure to training data. In addition to the correlation coefficients, corresponding p-values are reported to assess the statistical significance of the observed correlations. A low p-value (significant if $p < 0.05$) indicates that the correlation is unlikely to have arisen by random chance under the null hypothesis of no association, thereby supporting the reliability and validity of the model's semantic alignment with human judgments. Based on these results, model 1 & 2 are statistically significant because the null hypothesis (Fine-tuning the encoder on diacritized Arabic text does not affect performance on the STS benchmark) is rejected. Therefore, the paraphrase-xlm-r-multilingual-v1 model that achieved the highest Pearson correlation scores both before and after fine-tuning was selected for use in AraXLM. Fine-tuning this encoder model on diacritized Arabic text enhances semantic similarity detection, particularly when applied to paraphrase pretrained multilingual encoder variant.

TABLE VIII.
EXPERIMENTAL RESULTS OF PEARSON CORRELATION PRE- AND POST-FINE-TUNING ON DEV ATD SET

#	Model	Pearson Correlation (Before)	P-Value	Pearson Correlation (After)	P-Value
1	sentence-transformers/paraphrase-xlm-r-multilingual-v1	32.32%	0.0220	37.36%	0.0070
2	sentence-transformers/xlm-r-distilroberta-base-paraphrase-v1	32.32%	0.0220	37.03%	0.0081
3	sentence-transformers/stsb-xlm-r-multilingual	14.66%	0.310	18.96%	0.187
4	xlm-roberta-base	-1.71%	0.90	9.16%	0.53
5	xlm-roberta-large	-24.32%	0.088	5.94%	0.68

VI. COMPARISON WITH OTHER APPROACHES IN STS TRACK2

The SemEval-2017 STS Cross-lingual Arabic-English task provided benchmark reference systems against which new approaches can be evaluated [37]. The ECNU [38] and BIT [39] systems reported Pearson correlation scores of 0.74 and 0.69 , respectively, on the official test set[3]. In comparison, the fine-tuned variant incorporating diacritized Arabic text demonstrated competitive performance. The evaluated textual inputs included:

- The original English sentence from STS (En)
- The original Arabic sentence from STS (Ar)
- The Arabic-to-English translation of Ar (Ar-TrE)
- The diacritized Arabic sentence (ArATD)
- The translation of ArATD into English (ArATD-TrE)

ArATD_TrE_CosineSimilarity and FAISS IP (Inner Product) achieved the highest Pearson correlation scores of 0.78 and 0.77, respectively, outperformed both ECNU and BIT baselines (see TABLE IX). Additionally, Ar_TrE_CosSim exhibited strong performance (Spearman: 0.80, Pearson: 0.77), closely aligning with the top-performing metrics. These results confirm that leveraging translated and diacritized Arabic variants significantly enhances cross-lingual semantic similarity detection.

TABLE IX.
COMPARATIVE EVALUATION BASED ON STS DATASET

Metric	Spearman Correlation	Pearson Correlation
ECNU	NA	0.74
BIT	NA	0.69
IP_Score_FAISS	0.81	0.77
ArabicSentence_CosSim	0.75	0.72
Ar_TrE_CosSim	0.8	0.77
ArATD_CosSim	0.35	0.33
ArATD_TrE_CosSim	0.81	0.78

VII. DISCUSSION OF THE RESULTS AND CONCLUSION

This study aimed to evaluate the performance of six ATD tools for Arabic using established metrics, WER, DER, and BLEU. In response to research sub-question 1: What is the most effective ATD tool for integration into our proposed framework (AraXLM)?

Our experiments demonstrated that Shakkelha, a deep learning-based tool designed for Modern Standard Arabic, outperformed other models across key evaluation metrics. It achieved high character-level BLEU scores and low DER when case endings were excluded, making it the most effective candidate for integration into the AraXLM framework.

In response to research sub-question 2: What issues are identified when conducting our experimental study, particularly for Arabic?

Several challenges were noted, including:

- A significant increase in WER (up to 16 percentage points) when syntactic case endings (CE) were restored.

[3] https://docs.google.com/spreadsheets/d/1a5ZNg5IqKnBLaNKHMyVvmAn2_xc79FqwbXLL5z4ABAk/edit?gid=0#gid=0

- Difficulties in tokenising diacritized Arabic, due to non-standard outputs (e.g., merged diacritics like shaddah with sukun, or irregular representations of tanwīn).
- Variability in how tools apply full diacritization, sometimes leading to redundant or incorrect diacritic marks.

These issues highlight both linguistic and technical limitations in current ATD models when applied to Arabic morphology and syntax.

The results highlight that fully diacritizing every character is not always good and may introduce noise or errors, thereby skewing evaluation metrics and negatively impacting downstream NLP applications. The merging of diacritics (e.g., shaddah with sukun) and inconsistent representations (e.g., tanwīn as repeated fatha) complicate tokenization and evaluation.

These findings emphasize the requirement of standardised evaluation methodologies and tokenization practices that are sensitive to Arabic's unique linguistic structure. The implementation of robust ATD models is vital not only for accurate diacritization but also as a foundational layer in advanced NLP tasks such as semantic search and plagiarism detection across languages. The study also supports the integration of linguistic knowledge with deep learning models, showing the way for adaptable and accurate transformer-based systems in low-resource languages, such as Arabic.

Despite the insights gained from the results, it was limited by the lack of gold-standard diacritized datasets and the narrow range of tools, many of which were trained on specific text domains. Additionally, integrating these tools into Python required adjustments due to the complexity of Arabic script and morphology.

In conclusion, this study highlights the importance of accurate ATD for improving ANLP,and supports the integration of Shakkelha as the most effective tool within the proposed AraXLM framework. The results demonstrate its consistent performance across key evaluation metrics, while exposing limitations in current tokenisation and evaluation methods. These findings underscore the need for linguistically informed, standardised pre and post-processing approaches tailored to the structural complexity of the Arabic language.

In future work, the next phase investigate how diacritization affects cross-lingual semantic similarity in AraXLM. We plan to integrate ATD outputs within our proposed AraXLM framework to enhance performance in cross-lingual semantic similarity tasks with FAISS vector-based. This integration aims to improve accuracy, contextual adaptability, and the robustness of transfer learning systems in Arabic-English applications.

REFERENCES

[1] M. Elyaakoubi and A. Lazrek, "Justify just or just justify," *Journal of Electronic Publishing,* vol. 13, no. 1, 2010. doi: 10.3998/3336451.0013.105.

[2] R. Rjeily, Cultural Connectives: Bridging the Latin and Arabic Alphabets, vol. 1. Brooklyn, NY: Mark Batty Publisher, 2021.

[3] M. Hssini and A. Lazrek, "Design of Arabic Diacritical Marks," *International Journal of Computer Science*, vol. 8, no. 3, May 2011.

[4] M. Maamouri, A. Bies, and S. Kulick, 'Diacritization: A Challenge to Arabic Treebank Annotation and Parsing', *the International Conference on the Challenge of Arabic for NLP/MT* , pp. 35-47, 2006.

[5] S. Alzahrani, "Arabic plagiarism detection using word correlation in N-Grams with K-overlapping approach," Taif, 2015. https://ceur-ws.org/Vol-1587/T5-2.pdf

[6] E. M. B. Nagoudi et al., "2L-APD: A two-level plagiarism detection system for Arabic documents," *Cybernetics and Information Technologies,* vol. 18, no. 1, pp. 124–138, 2018. doi: 10.2478/cait-2018-0011.

[7] B. Akanksha et al., "A survey on plagiarism detection," *International Journal of Computer Applications*, vol. 10, no. 8, pp. 2359–2365, 2017. http://www.ripublication.com

[8] M. F. Akan et al., "An analysis of Arabic-English translation: Problems and prospects," *Advances in Language and Literary Studies,* vol. 10, no. 1, p. 58, Feb. 2019. doi: 10.7575/aiac.alls.v.10n.1p.58.

[9] M. Alshehri, N. Beloff, and M. White, "AraXLM: New XLM-RoBERTa based method for plagiarism detection in Arabic text," *in Intelligent Computing*, K. Arai, Ed., Cham: Springer, 2024, pp. 81–96. doi: 10.1007/978-3-031-62277-9_6

[10] M. M. Elmallah et al., "Arabic diacritization using morphologically informed character-level model," *in Proc. LREC-COLING* 2024, Torino, Italy: ELRA and ICCL, May 2024, pp. 1446–1454. https://aclanthology.org/2024.lrec-main.128/

[11] K. Shaalan and Khaled, "Rule-based approach in Arabic natural language processing," *International Journal on Information and Communication Technologies*, vol. 3, p. 11, May 2010.

[12] A. Chennoufi and A. Mazroui, "Morphological, syntactic and diacritics rules for automatic diacritization of Arabic sentences," *Journal of King Saud University* - Computer and Information Sciences, vol. 29, no. 2, pp. 156–163, 2017. doi: 10.1016/j.jksuci.2016.06.004.

[13] M. Mézard and J.-P. Nadal, "Learning in feedforward layered networks: the tiling algorithm," *Journal of Physics A*, vol. 22, pp. 2191–2203, 1989. https://api.semanticscholar.org/CorpusID:44826720

[14] W. Almanaseer et al., "A deep belief network classification approach for automatic diacritization of Arabic text," *Applied Sciences,* vol. 11, no. 11, 2021. doi: 10.3390/app11115228.

[15] N. Srivastava et al., "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, Jan. 2014.

[16] Y. Alalawi et al., "A CNN-based Arabic diacritic symbol recognition system using domain adaptation," i*n Proc. 8th Int. Conf. Sustainable Information Engineering and Technology (SIET*), New York, USA: ACM, 2023, pp. 23–32. doi: 10.1145/3626641.3627212.

[17] H. Hewamalage et al., "Recurrent neural networks for time series forecasting: Current status and future directions," *International Journal of Forecasting*, vol. 37, no. 1, pp. 388–427, 2021. doi: 10.1016/j.ijforecast.2020.06.008.

[18] Y. Belinkov and J. Glass, "Arabic diacritization with recurrent neural networks," in Proc. *EMNLP 2015, Lisbon, Portugal: ACL*, Sep. 2015, pp. 2281–2285. doi: 10.18653/v1/D15-1274.

[19] A. Vaswani et al., "Attention is all you need," *in Proc. NeurIPS* 2017, 2017. http://arxiv.org/abs/1706.03762

[20] A. Assad et al., "Transformer-based automatic Arabic text diacritization," *Sustainable Engineering and Innovation*, vol. 6, no. 2, pp. 285–296, Nov. 2024. doi: 10.37868/sei.v6i2.id305.

[21] Y. Bengio et al., "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, Mar. 1994. doi: 10.1109/72.279181.

[22] A. Gillioz, J. Casas, E. Mugellini and O. Abou Khaled, "Overview of the Transformer-based Models for NLP Tasks," Proceedings of the Federated Conference on Computer Science and Information Systems, vol. 21, pp. 179–183, 2020, doi: 10.15439/2020F20.

[23] R. Al-Sabri and J. Gao, "LAMAD: A linguistic attentional model for Arabic text diacritization," *in Findings of the Association for Computational Linguistics: EMNLP 202*1, Punta Cana, Dominican Republic: ACL, Nov. 2021, pp. 3757–3764. doi: 10.18653/v1/2021.findings-emnlp.317.

[24] M. Al-Badrashiny et al., "A layered language model based hybrid approach to automatic full diacritization of Arabic," *in Proc. 3rd Arabic*

NLP Workshop, Valencia, Spain: ACL, Apr. 2017, pp. 177–184. doi: 10.18653/v1/W17-1321.

[25] H. Alaqel and K. El Hindi, "Improving diacritical Arabic speech recognition: Transformer-based models with transfer learning and hybrid data augmentation," *Information, vol.* 16, no. 3, 2025. doi: 10.3390/info16030161.

[26] O. Obeid et al., "CAMeL Tools: An open source Python toolkit for Arabic NLP," 2020. http://qatsdemo.cloudapp.net/farasa/

[27] A. Abdelali et al., "Farasa: A fast and furious segmenter for Arabic," *in Proc. NAACL Demonstrations*, San Diego, CA: ACL, Jun. 2016, pp. 11–16. doi: 10.18653/v1/N16-3003.

[28] F. Alasmary et al., "CATT: Character-based Arabic Tashkeel Transformer," *arXiv preprint*, vol. abs/2407.03236, 2024. https://api.semanticscholar.org/CorpusID:270924323

[29] B. Al-Rfooh et al., "Fine-Tashkeel: Fine-tuning byte-level models for accurate Arabic text diacritization," *in Proc. IEEE JEEIT 2023*, pp. 199–204, 2023. https://api.semanticscholar.org/CorpusID:257767345

[30] B. M. King, "Analysis of variance," *in International Encyclopedia of Education,* 3rd ed., Jan. 2009, pp. 32–36. doi: 10.1016/B978-0-08-044894-7.01306-3.

[31] A. Lazrek, "Arabic mathematical notation," *National Institute of Standards and Technology,* USA, 2006. https://www.w3.org/TR/2006/NOTE-arabic-math-20060131/

[32] K. Darwish et al., "Arabic diacritization: Stats, rules, and hacks," *in Proc. 3rd Arabic NLP Workshop,* Valencia, Spain: ACL, Apr. 2017, pp. 9–17. doi: 10.18653/v1/W17-1302.

[33] A. Fadel et al., "Neural Arabic text diacritization: State of the art results and a novel approach for Arabic NLP downstream tasks," *ACM Transactions on Asian and Low-Resource Language Information Processing,* vol. 21, no. 1, Jan. 2022. doi: 10.1145/3470849.

[34] O. Obeid et al., "CAMeL Tools: An open source Python toolkit for Arabic NLP," *in Proc. LREC 2020, Marseille, France: ELRA,* May 2020, pp. 7022–7032. https://aclanthology.org/2020.lrec-1.868/

[35] T. Zerrouki, "Towards an open platform for Arabic language processing," 2020. doi: 10.13140/RG.2.2.29882.82881.

[36] A. Fadel et al., "Neural Arabic text diacritization: State of the art results and a novel approach for machine translation," *in Proc. 6th Workshop on Asian Translation,* Hong Kong, China: ACL, Nov. 2019, pp. 215–225. doi: 10.18653/v1/D19-5229.

[37] D. Cer, M. Diab, E. Agirre, I. Lopez-Gazpio, and L. Specia, "SemEval-2017 Task 1: Semantic Textual Similarity – Multilingual and Cross-lingual Focused Evaluation," in Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017), Vancouver, Canada, Aug. 2017, pp. 1–14.

[38] Y. Tian, Y. Song, H. Xia, Y. Li, and Q. Zhang, "ECNU at SemEval-2017 Task 1: Leverage Kernel-Based Traditional NLP Features and Distributed Word Representations for Semantic Textual Similarity Estimation," in Proc. 11th Int. Workshop on Semantic Evaluation (SemEval-2017), Vancouver, Canada, Aug. 2017, pp. 125–131. https://aclanthology.org/S17-2015

[39] H. Wu, H. Huang, P. Jian, Y. Guo, and C. Su, "BIT at SemEval-2017 Task 1: Using Semantic Information Space to Evaluate Semantic Textual Similarity," in *Proc. 11th Int. Workshop on Semantic Evaluation (SemEval-2017)*, Vancouver, Canada, Aug. 2017, pp. 77–84. https://aclanthology.org/S17-2007