*Article*

# Employing a Multilingual Transformer Model for Segmenting Unpunctuated Arabic Text

**Abdullah Alshanqiti [1,†], Sami Albouq [2,†], Ahmad Alkhodre [3,†], Abdallah Namoun [4,†], and Emad Nabil [5,†]**

† Faculty of Computer and Information Systems, Islamic University of Madinah, Madinah 42351, Saudi Arabia
* Corresponding author: Abdullah Alshanqiti a.m.alshanqiti@gmail.com; amma@iu.edu.sa.

**Abstract:** Long unpunctuated texts containing complex linguistic sentences are a stumbling block to processing any low-resource languages. Thus, approaches that attempt to segment lengthy texts with no proper punctuation into simple candidate sentences are a vitally important preprocessing task in many hard-to-solve NLP applications. In this paper, we propose (PDTS) a punctuation detection approach for segmenting Arabic text, built on top of a multilingual BERT-based model and some generic linguistic rules. Furthermore, we showcase how PDTS can be effectively employed as a text tokenizer for unpunctuated documents (i.e., mimicking the transcribed audio-to-text documents). Experimental findings across two evaluation protocols (involving an ablation study and a human-based judgment) demonstrate that PDTS is practically effective in both performance quality and computational cost.

**Keywords:** text splitting; text tokenization; transfer learning; mask-fill prediction; NLP linguistic rules; missing punctuations; cross-lingual BERT model; Masked Language Modeling

## 1. Introduction

Segmenting unpunctuated textual documents into simple candidate sentences is a vitally important preprocessing task for the success of many NLP applications. However, this ambiguous segmentation task is intuitively challenging when attempting to tokenize long text that consists of compound sentences but with no proper punctuation (e.g., no obvious clue how to apply sentence-based tokenization when full-stop punctuation is missing). With hard-to-solve NLP applications that require text understanding (such as automatic summarization [1], simplification [2–4], and question-answering [5]), text segmentation is considered a key preprocessing step for achieving a high-precision performance. Take the example of summarizing a large transcribed audio-to-text document, targeting individuals with low-literacy skills [6] (e.g., non-native readers or children). Here, such an audio document requires complicated or probably manual preprocessing to perform the needed tokenization precisely.

Lengthy unpunctuated texts reveal to be a stumbling block to processing any low-resource languages. The Arabic language is no exception [7] due to its high linguistic complexity and shortage in having needed parsing and tokenizing resources, including complete lexicons. More in detail, some of the linguistic hurdles related to text splitting have fundamentally resulted from the ambiguity in identifying (1) word-stemming/lemmatization (as the Arabic language has a significantly rich derivational morphology) and (2) POS-tagging precisely (i.e., in the case the diacritics are missing, which is usually the case) [1].

Approaches to tackling text segmentation problems (i.e., under the low-resource setting) can be distinctly classified into either traditional or transfer-based paradigms. In this context, traditional paradigm would include approaches that focus on, e.g., constructing hand-crafted rules for text segmentation or extensively designing/training a specific monolingual model. Transfer-based approaches, however, strive to take advantage of languages with rich resources (such as English) in two ways: transferring resources (a.k.a cross-lingual transfer learning) or adopting a pre-trained multilingual model for transferring the needed NLP capabilities [8]. Concerning the Arabic language, different text segmentation methods have been put forward, focusing only on the traditional techniques [1,9–15], and to our knowledge, no transfer-based method has been proposed yet.

In this paper, we shed motivating insights into the capabilities of adopting a multilingual model for the Arabic text segmentation. We consider multilingual BERT [16,17], a language understanding model trained on a Masked-Language Modeling (MLM) objective, which can be fine-tuned for the *Mask-fill* task (i.e., allows predicting masked words according to the context of a given input text without reconstructing the entire text). While few works have attempted to train an Arabic-specific BERT model extensively (such as AraBERT [18], CAMeLBERT [19]), we hypothesize and reveal that fine-tuning a state-of-the-art pre-trained multilingual model for the needed task could give highly competitive results, compared with the monolingual Arabic models, in a more straightforward manner. In a nutshell, the major contributions presented in this paper are stated as follows:

- We propose a punctuation detection approach for splitting a complex Arabic text into a set of simple candidate sentences; each sentence is potentially representing an independent clause. This approach begins with employing an MLM method to detect the missing punctuations in a long-given text and then validates them using straightforward linguistic rules. The originality lies in employing a transfer-based paradigm, for the first time, to address the Arabic text segmentation problem.
- Since our approach can generate a large set of potentially missed punctuations that are prone to prediction errors, we tackle this by proposing a greedy strategy for selecting the best subset punctuations (i.e., subject to unknown size of the optimal subset).
- We showcase the efficient performance and practicality of our proposal through several experimental evaluations conducted on two public Arabic corpora. The evaluation protocols include standard quantitative metrics with an ablation study and a qualitative human-based judgment. Further, we report on an implementation and make the code publicly available for replicating our experiments[1].

The rest of this paper unfolds as follows. First, section 2 strives to examine gaps in the literature regarding Arabic text segmentation approaches, focusing on their main drawbacks. Next, section 3 introduces our proposed punctuation detection method for text splitting, and section 4 presents the subsequent analysis and results. Lastly, section 5 concludes the paper and mentions our intention to extend this work in the future.

## 2. Related Literature

Text segmentation is rooted as a preprocessing phase in many needed solutions that focuses on philosophical understandings of natural languages. As a consequence, one can review various text segmentation approaches with different objectives within NLP-literary circles (see the pre-2018 approaches surveyed in [25] and, more recently, in [26]). To elaborate more, existing approaches endeavor to split a given text into subtexts while tackling different levels of language complexities. These levels range from identifying morphological and lexical cohesion boundaries within word-to-small text segments [1,4,13] (e.g., words, phrases, and simple discourse[2] sentences) to higher-level syntactic and thematic structures [22,24,27] (such as paragraphs, passages, chapters, or textual documents containing many chapters). In terms of the underlying technique used, existing text segmentation approaches also vary according to their chronological development, which can be classified into three genres:

- the early discourse segmentation approaches (some approaches are dated back to the previous decade), which often depend on extracting linguistic annotation features, including treebank structure and POS tags [4,9,10,22];
- the statistical and machine learning methods, which often manipulate text similarity metrics and distances; and recently [12–14,24];

[1] https://github.com/AMahfodh/ArSummarizer
[2] a.k.a Elementary Discourse Units (EDUs)

**Table 1.** A brief summary of text segmentation studies carried out for Arabic and English languages.

| | | | Method | | | | Dialect- | Language | Transfer |
|---|---|---|---|---|---|---|---|---|---|
| Language | Reference | Segmentation level | A | B | C | D | specific | Understanding | Learning |
| Arabic | Rhetorical SVM classifier [9] | Phrase to a simple sentence | | | ✓ | | | ✗ | |
| | Linear-chain Markov model [10] | Word | | ✓ | ✓ | | ✓ | ✗ | |
| | Chunking procedure [11] | Random between word to small text segments | ✓ | | | | | ✗ | |
| | USeg (utterances) [12] | Phrase to a simple sentence | | ✓ | ✓ | | ✓ | ✗ | |
| | Farasa - linear SVM model [13] | Word | | ✓ | ✓ | | | ✗ | |
| | SVM and Bi-directional LSTM [14] | Word | | | ✓ | ✓ | | ✗ | |
| | DJAZI (Chunking procedure) [15] | Random between word to small text segments | ✓ | | | | | ✗ | |
| | Regular rule-based expressions [1] | Simple sentence (independent clause) | ✓ | ✓ | | | | ✗ | |
| | Our PDTS | Simple sentence (in)dependent clause | | ✓ | | ✓ | | Multilingual | ✓ |
| English | SegBot [20] and [21] | Sentence and document | | | | ✓ | | ✗ | |
| | DisSim [22] | Discourse sentence | | ✓ | | | | ✗ | |
| | Three BERT-style models [23] | Discourse sentence and document | | | | ✓ | | Monolingual | |
| | Context-preserving approach [4] | Simple sentence | | ✓ | | | | ✗ | |
| | TopicDiff-LDA *Latent Dirichlet Allocation* [24] | Document | | | ✓ | | | ✗ | |

A: Chunking procedure based on pre-defined split-delimiters.
B: Document/discourse segmentation approaches depending on extracting linguistic annotation features.
C: Statistical / classical machine learning methods
D: Deep learning / transformer-based models

- the natural language understanding (NLU) approaches depending on artificial neural networks, particularly those adopted transformer-based models (e.g., [23]).

In this paper, we follow the third genre for generating subtexts that potentially represent independent clauses by surrounding them with proper punctuation predicted.

To date, tremendous NLP work has made a productive contribution to only a few Indo-European languages (mainly, the English language, which is an extremely high-resource). In contrast, work on low-resource languages such as Arabic is still rudimentary. Unfortunately, a handful of related text segmentation approaches are suggested for Arabic [1,9–15]. Therefore, in the remaining of this section, we limit the discussion by reviewing these related approaches only.

Word-level segmentation solutions are presented wider in the literature (e.g., [10,13,14]) than the other higher and linguistically more complex levels (such as phrases/sentences and (in)dependent clauses, e.g., [1,9,11,12,15]). In the former segmentation, [10] suggested an improved clitic segmentation model (at a character-based level) targeting formal and informal dialectal Egyptian text. In particular, the authors extend the labeled features of an existing linear-chain Markov model and then retrain it on some annotated Egyptian corpus. Nonetheless, the proposed extension does not seem to generalize well to other Arabic dialects. In a similar line with [10], [13] proposed (Farasa, which means *insight* in Arabic) a more robust character-level segmentation model (implemented using a linear SVM classifier) for anticipating segmentations of a word. However, it depends heavily on extracted lexicons and morphological features. As yet, Farasa is one of the state of the art word-level segmentation method, which outperforms other well-known similar approaches, including Madamira [28] and Stanford-Arabic[3]. A follow-up attempt to improve existing word-level segmentation approaches, depending on an SVM and Bi-directional LSTM machine learning models, is suggested by [14]. Their models are trained on authors-created corpora containing naturally occurring text from social media. While [14] reports an improved performance compared to [13], [13] seems much lighter and more straightforward to adopt.

Concerning sentence-level segmentation solutions that we consider in this paper, we come across quite a few works: [1,9,11,12,15]. Here, [9] attempted to segment texts into phrases/simple sentences based on just one specific linking word (i.e., wāw, /w/, *AND* in English), which is a coordinating conjunction that has six rhetorical rules associated with it. The authors trained and tested a simple SVM classifier on a small-scale dataset for predicting the rhetorical type that allows text splitting. Nonetheless, many other conjunction types are not covered, which critically raises its shortcomings and hinders its usability. Another attempt based on an SVM classifier is suggested by [12]. The authors, here, proposed (USeg) for segmenting turn (a complete transcribe speech-to-text unit) into utterances (a segment of a turn), trained on authors-defined Egyptian Arabic corpus. This USeg relies on extracting morphological features extracted from their annotated data. However, the considered morphological features (e.g., n-gram contextual word, conjunction POS, and previously predicted tags) could be expensive to implement and may not scale well with a medium-to-large textual input.

Furthermore, [15], proposed (DJAZI) a text segmentation tool for Arabic, which attempts to split texts into contextual sentences as well as morphological words. The authors implemented a simple brute-force procedure for identifying split boundaries based on punctuation marks (assumed to be available for sentence splitting) and a lexical knowledge database for word matching and isolating. Regardless, their considered problem seems far from being addressed by the suggested DJAZI, and more importantly, their procedure is not capable of handling any unpunctuated texts that we can do in our PDTS. In addition, [11] and [1] proposed a handy text chunking process based on a pre-defined list of conjunctions as text split delimiters for randomly splitting texts into word to small text segments. Nonetheless, their error rate for determining segment boundaries is expected to be high as no precise linguistic rules are considered in their approaches.

In Table 1, we briefly outline all the post-2011 text segmentation studies we encountered, carried out for Arabic and English. Based on the literature inspected, the apparent limitations/drawbacks of the mentioned approaches, particularly for Arabic, are:

[3] Stanford-Arabic NLP https://nlp.stanford.edu/projects/arabic.shtml

- The high computational effort consumed in generating linguistic annotation features;
- The uncertainty in implementing large discourse segmentation, especially with unpunctuated texts; and
- The inability to effectively handle multiple textual dialects that often raises the problem of having variable size output vocabulary. This inability mainly results from training specific models on a limited available annotated corpus.

Irrespective of these drawbacks, our approach is ahead of all works (i.e., summarized in Table 1) in handling large unpunctuated text effectively without text-annotating nor going through ambiguous linguistic analysis (e.g., extracting TreeBank structures). Moreover, to the best of our knowledge, this paper introduces a novel work concerning the use of the MLM method depending on a leading state-of-the-art multilingual model (mBERT) for clause-based text segmentation.

## 3. PDTS: Punctuation Detector for Text Segmentation

Encouraged by the MLM method that particularly enables learning complex linguistic patterns of a natural language in a self-supervised objective, we base our proposed punctuation detector for text splitting (PDTS) on top of a robust pre-trained MLM. Given a very large complex text $X$ (typically a document composed of sentences but without full-stop punctuation) that consists of a sequence of words $X = \{x_1, x_2, \cdots, x_{|n|}\}$, PDTS attempts to segment $X$ into a set of simple sentences $Y \leftarrow PDTS(X)$. Each simple sentence $Y_i$ represents an independent-clause segmentation without rephrasing. Intuitively, the problem that needs to tackle here lies in detecting the missing punctuations (mainly, full-stop, comma, semicolon, colon, exclamation/question marks).

Considering Figure 1, PDTS begins with detecting the missing punctuations using mBERT (described in A, B, and C), then it applies some linguistic rules for validating them (described in D and E). In the following sections, we detail the underlying steps of our PDTS (expressed outrightly in algorithm 1 and 2) and use Figure 1 for illustration.

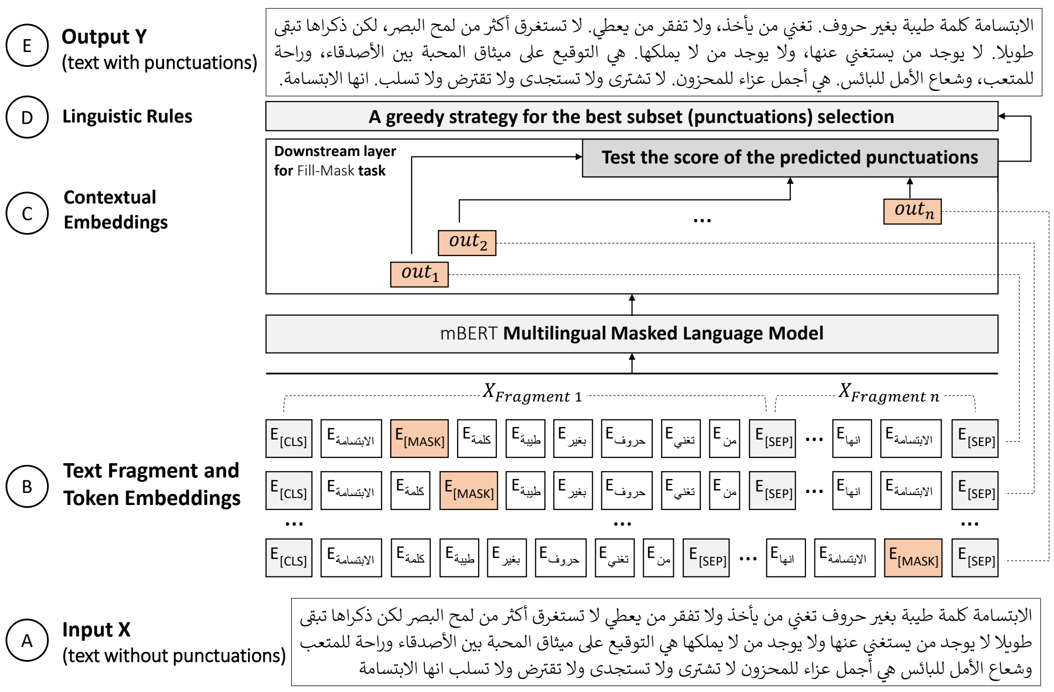
### *3.1. mBERT: Multilingual Masked Language Model*

Multilingual BERT (mBERT[4]) is an extended language understanding model from the original monolingual BERT [16] (Bidirectional Encoder Representations from Transformers [29]) with a focus on improving different cross-lingual NLP tasks. mBERT is pre-trained on a tremendous amount of Wikipedia text from the top 104 languages using the MLM method. Its stacked Transformer architecture consists of 12-layer, 768-hidden, and 12-heads, which gives 110M parameters in total. While mBERT allows sharing vocabulary and weights parameters across these languages, the training samples are designed based on the monolingual method, and no cross-lingual objectives are considered. However, mBERT generalizes well from the monolingual training data and surprisingly gives state-of-the-arts results on cross-lingual benchmarks [17].

Instead of extensively training a large language understanding model on a specific non-English language (such as ArBERT [30], AraBERT [18], CAMeLBERT [19] for the Arabic language), a more straightforward alternative is to fine-tune a comprehensive pre-trained multilingual model for the needed task that could give highly competitive results. We discuss the performance of some of these models (cf. section 4), but for designing our approach, we follow the latter (i.e., the base of our approach is mBERT, which is fine-tuned for the *Mask-fill* task) as it gives better performance for understanding the Arabic sentences in general.

The input and output representations of mBERT (like any other pre-defined encoder-based models) are grounded to token embeddings at different layers (i.e., word token embedding, segment embedding, transformer positional embedding, and contextual token embedding for the output). We consider all these embeddings in our approach except segment embedding (i.e., recognized by the specific BERT token $[SEP]$) as our input text

[4] https://github.com/google-research/bert/blob/master/multilingual.md


E Output Y (text with punctuations)
D Linguistic Rules
C Contextual Embeddings
B Text Fragment and Token Embeddings
A Input X (text without punctuations)
A greedy strategy for the best subset (punctuations) selection
Downstream layer for Fill-Mask task
Test the score of the predicted punctuations
$out_1$
$out_2$
...
$out_n$
mBERT Multilingual Masked Language Model
$X_{Fragment\ 1}$
$X_{Fragment\ n}$
E[CLS] Eالابتسامة E[MASK] Eكلمة Eطيبة Eبغير Eحروف Eتغني Eمن E[SEP] ... Eانها Eالابتسامة E[SEP]
E[CLS] Eالابتسامة Eكلمة E[MASK] Eطيبة Eبغير Eحروف Eتغني Eمن E[SEP] ... Eانها Eالابتسامة E[SEP]
E[CLS] Eالابتسامة Eكلمة Eطيبة Eبغير Eحروف Eتغني Eمن E[SEP] ... Eانها Eالابتسامة E[MASK] E[SEP]

الابتسامة كلمة طيبة بغير حروف. تغني من يأخذ، ولا تفقر من يعطي. لا تستغرق أكثر من لمح البصر، لكن ذكراها تبقى طويلا. لا يوجد من يستغني عنها، ولا يوجد من لا يملكها. هي التوقيع على ميثاق المحبة بين الأصدقاء، وراحة للمتعب، وشعاع الأمل للبائس. هي أجمل عزاء للمحزون. لا تشترى ولا تستجدى ولا تقترض ولا تسلب. انها الابتسامة.

الابتسامة كلمة طيبة بغير حروف تغني من يأخذ ولا تفقر من يعطي لا تستغرق أكثر من لمح البصر لكن ذكراها تبقى طويلا لا يوجد من يستغني عنها ولا يوجد من لا يملكها هي التوقيع على ميثاق المحبة بين الأصدقاء وراحة للمتعب وشعاع الأمل للبائس هي أجمل عزاء للمحزون لا تشترى ولا تستجدى ولا تقترض ولا تسلب انها الابتسامة

**Figure 1.** Overview of our proposed PDTS that detects missing punctuations in a complex Arabic text $X$ with the purpose of chunking $X$ into concatenated simple sentences $Y$. The translation of the mentioned Arabic text $X$ is *Smile is the right word without letters. It enriches the one who receives but does not diminish the one who gives. It does not last more than a jiffy to behold, but its memory remains long. No one can do without it, and no one does not own it. It is the signing of the charter of respect between friends, a relief from the tiredness, and a ray of hope for the miserable. It is the most beautiful consolation for the dejected. It is neither purchasable, borrowable, nor lootable. It's the smile.*

is a single large complex sentence (i.e., assumed to have no full-stop punctuation). For fine-tuning mBERT on downstream *Mask-fill* task, we consider the output Softmax dense layer and inspect only the contextual token-to-vocab vectors that are mapped with the input token $[MASK]$, see (B and C) of Figure 1. In other words, to predict a masked vocab from contextual-mBERT vectors (i.e., at the output embeddings layers), we implement:

$$p(v|X;\theta) = Softmax(W^T c) \tag{1}$$

where $W \in \mathbb{R}^{|X| \times |V|}$ is the weight matrix of the last staked FFNN layer (i.e., used to project the contextual vector ($c$) of the specified masked token into mBERT vocabulary ($V$)), $v$ is the predicted masked vocab (i.e., has the highest softmax probability score), and $\theta$ is a decision-threshold parameter.

## 3.2. Arabic Text Fragments and Masked Token-to-Vocab Prediction

For tokenizing $X$ into a set of independent-clauses, PDTS requires invoking mBERT $|X|$ times to detect whether the space between every two sequential tokens should include proper punctuation. Let $X = \langle t_1, t_2, t_i, \cdots, t_n \rangle$ be a sequence of $n$ tokens, $i$ is the index of token-sequence, we generate $M = \{t_1^m, t_2^m, t_i^m, \cdots, t_n^m\}$ by invoking mBERT $n$ times, such that $t_i^m$ represents the temporarily predicted masked vocab between ($t_i$ and $t_{i+1}$) when $i < n$ or after $t_i$ when $i = n$. To clarify more, we generate $M$ by (1) inserting $[MASK]$

into $X$ at $i$ index (denoted as $X'_i \leftarrow insertMASK(X, i), \forall i \in X$) and (2) we collect the set of mBERT's outputs, as follows:

$$\begin{aligned} t^m_1 &\leftarrow mBERT(\langle t_1, [MASK], t_2, t_i, \cdots, t_n\rangle) \\ t^m_i &\leftarrow mBERT(\langle t_1, t_2, t_i, [MASK], \cdots, t_n\rangle) \\ t^m_n &\leftarrow mBERT(\langle t_1, t_2, t_i, \cdots, t_n, [MASK]\rangle) \end{aligned} \tag{2}$$

Intuitively, invoking mBERT many times can affect the runtime efficiency of PDTS, particularly when applying it for a large-scale $X$ that might contain a long sequence of sentences (e.g., a document containing $|X| > 5k$). We address scaling PDTS by reducing the size of the input sequence from $|X|$ down to $\alpha$ without significantly sacrificing the detection accuracy of $t^m_i$ ($\alpha$ is a positive integer pre-defined parameter). In algorithm 1, we describe the steps of invoking mBERT with a fixed size of input sequence $\alpha$ (i.e., supposed to be smaller than $|X|$), see lines $6 - 11$. To use PDTS, the user must first declare a set $Pun$ containing their desired punctuations to be added in $X$ (see line 2 of algorithm 1) besides a decision-threshold $\theta$ for temporary-accepting/ignoring the predicted punctuation:

$$p(pu^m_i | t^m_i; pun, \theta) = mBERT(X'_i) \tag{3}$$

where $pu^m_i$ represents the accepted mBERT's output, $pun$ and $\theta$ are model parameters used to filter out $t^m_i$ (i.e., obtained by Eq. (2)). In lines $12 - 13$ of algorithm 1, we implement Eq. (3) to filter out the predicted masked tokens ($t^m_i$), and temporarily store $pu^m_i$ in $PunTokens$ to be validated next by our linguistic rules.

## 3.3. A Greedy Strategy for Selecting the Best Subset Punctuations

Up to line 15 of algorithm 1, the discussion concerning the predicted masked tokens has been relatively abstract. Now we introduce our greedy strategy for selecting the best subset punctuations from $PunTokens$ (implemented in algorithm 2 and presented in Figure 1 (D)), such that $Pun^{best} \in PunTokens$. This strategy aims to iteratively validate/accept $pu^m_i$ according to Eq. (3) besides our defined linguistic rules, starting in each iteration with $pu^m_i$ that has the highest accuracy score (i.e., $pu^m_i \leftarrow getPredWithMaxScore(PunTokens)$). Given two text segments $SB$ and $SA$ (i.e., text segment before and after the punctuation $pu^m_i$, see lines $6 - 10$ of algorithm 2), extracted at a particular iteration, our linguistic rules validate if $SB$ and/or $SA$ contains simple sentences to be separated.

More in general regarding the Arabic language grammar, simple Arabic sentences are linguistically categorized into either: nominal-sentence (noun+noun) or verbal-sentence (typically, a verb followed by a noun); each is formulated with a minimum of two words. Depending on the parts-of-speech of the first word, the type of sentence can be determined [31]. In this context and without delving into the complex subcategories of Arabic sentences, we apply four straightforward rules to validate if $SB$ and/or $SA$ includes at least one simple sentence. Let $s$ be the text segment that we want to validate, we define our linguistic rules in Table 2, such that *R1* checks the minimum number of words allowed in a simple sentence; *R2* and *R3* check whether $s$ is a nominal or a verbal sentence; and *R4* is a special case rule applicable when full-stop punctuation precedes $s$.

**Table 2.** Our generic linguistic rules used in PDTS to validate the detected punctuations.

| | |
|---|---|
| **R1** | **if** $\lvert s \rvert \geq 2$ **then** *true* : *false* otherwise. |
| **R2** | **if** nltk[5].pos_tag($s$).numOf($NOUN$) $\geq 2$ **then** *true* : *false* otherwise. |
| **R3** | **if** nltk.pos_tag($s$).numOf($NOUN$) $\geq 1$ **and** nltk.pos_tag($s$).numOf($VERB$) $\geq 1$ **then** *true* : *false* otherwise. |
| **R4** | **if** $t^m_i$ ==nltk.pos($Full\_Stop$) and $s \in \emptyset$ **then** *true* : *false* otherwise. |

As explained, the analysis of the collected punctuations in $PunTokens$ (i.e., obtained in the first step by mBERT) performs in a greedy-like strategy, which iteratively selects a $pu_i^m$ with the highest prediction's score. In principle, this subset selection problem is known to be an NP-hard problem [32](i.e., it cannot be solved in an exact polynomial time). However, a greedy-like strategy often provides a well-approximated solution to the target optimization problem and is practically implemented in many applications, including, e.g., sparse learning and feature data mining selection. In our problem, the size of the best subset selection $|Pun^{best}|$ is unknown, and the search for selection terminates once no $pu_i^m$ is satisfying the rules presented in Table 2. Formally, our PDTS attempts to optimize the subsect selected punctuations (i.e., $Pun^{best}$) as follows:

$$\begin{aligned} \underset{Pun^{best}}{\text{maximize}} \quad & f(Pun^{selected}) \\ \text{subject to} \quad & Pun^{selected} \in PunTokens. \end{aligned} \tag{4}$$

where $f$ represents our objective function (described in algorithm 2 lines $3-14$) that aims to maximize the number of splits in $x$ by adding the best punctuations from $PunTokens$.

**Algorithm 1:** The proposed procedure for predicting punctuations in an unpunctuated document using an MLM model.

```
Function: $PunTokens \leftarrow$ predPunctuations($X, \alpha, \theta$)
input   : $X$ an unpunctuated document, consisting of a sequence of words.
          $\alpha$ a positive integer parameter for fragmenting $X$ into smaller
          pieces, such that $(1 \leq \alpha \leq 512)$.
          $\theta$ a probability-threshold for temporary accepting the predicted
          punctuations.
output  : $PunTokens$ a set of punctuations predicted by an MLM model (e.g., $mBERT$) to
          be added into $X$.
1  // Declarer a set of user-defined punctuations in $Pun[]$, for example:
2  $Pun[] \leftarrow \{ colon, question-mark, exclamation-mark, full-stop, comma \}$
3  $PunTokens \leftarrow \emptyset$
4  $iBeginIndex = 0$
5  $wx \leftarrow$ wordTokenizer($X$)
6  if $\alpha > |X|$ then $\alpha \leftarrow |X|$
7  if $\alpha > 512$ then $\alpha \leftarrow 512$  // This is limited by some BERT-based models.
8  for $i \leftarrow 0$ to $|wx|$ do
9      if $\alpha/2 \leq i \leq (|wx| - \alpha)$ then $iBeginIndex += 1$
10     $Fragment^i \leftarrow$ subStringFromTokens ($iBeginIndex, \alpha$)
11     $Fragment^i \leftarrow$ addMaskTokenAtIndex ($Fragment^i, i$)
12     $Pred \leftarrow MLM.$FillMask ($Fragment^i$)  // In PDTS, we use mBERT
13     if $pred.Vocab \in Pun$ and $\theta \leq pred.Score$ then
14         $PunTokens$.append($i, pred.Vocab, pred.Score$)
15 return $X$ and $PunTokens$
```

## 4. Experiments and Discussion

The ultimate overarching purpose behind our proposed PDTS is to tokenize long-given texts into simple candidate sentences, employing (1) linguistic sentence rules along with (2) a punctuation prediction mechanism. Thus, to explore the overall effectiveness of PDTS, we conducted several experiments to evaluate each aspects, elaborated as follows. First, we measured the overall performance of PDTS concerning different input sizes ($|X|$) along with different settings for $\alpha$. Here, we shedded light on the comparison with the closely related state-of-the-art pre-trained base models. Second, we studied the humanistic point of view in eliciting and evaluating the performance of PDTS (i.e., by focusing on

**Algorithm 2:** The proposed procedure for validating the predicted punctuations in algorithm 1 and selecting the best ones to be added in $X$. These added punctuations can be then used as split delimiters for segmenting $X$

```
Function: $Y \leftarrow$ PDTS($X$, $PunTokens$)
input   : $X$ and $PunTokens$ are the return from algorithm 1
output  : $Y$ the text in $X$ with the best detected punctuations added
          (i.e., $Pun^{selected} \in PunTokens$).
1  $Y \leftarrow X$
2  $Pun^{selected}[] \leftarrow \emptyset$
3  repeat
4      $maxPred \leftarrow$ getPredWithMaxScore ($PunTokens$)
5      // Recursively, we extract the text segments before (SB) and after (SA) the predicted
         $maxPred.Vocab$. These surrounding segments are determined based on the added
         punctuations to $Y$ at each iteration
6      $SB, SA \leftarrow$ getSurroundingTextSegments ($Y, maxPred$)
7      $SB^v \leftarrow [R1(SB)$ and $(R2(SB)$ or $R3(SB))]$
8      $SA^v \leftarrow [(R1(SA)$ and $(R2(SA)$ or $R3(SA)))$ or $R4(SA, maxPred.Vocab)]$
9      // Rules (R1-R4) are presented in Table 2.
10     if $SB^v == SA^v == true$ then
11         $Y \leftarrow$ addPunctuation ($Y, maxPred.index, maxPred.Vocab$)
12         $Pun^{selected}$.append ($maxPred$)
13     $PunTokens$.remove ($maxPred$)
14 until $PunTokens == \emptyset$
15 // Return $Y$ that contains $X$ plus the added punctuations.
16 // Here, we can straightforwardly tokenize $Y$ based on $Pun^{selected}$ as split delimiters (i.e.,
     $Y.split(Pun^{selected})$).
17 return $Y$
```

both aspects: linguistic sentence rules and punctuation prediction depending on the MLM method). For reproducibility purposes, we make the code and files of our experiments publicly available under[1].

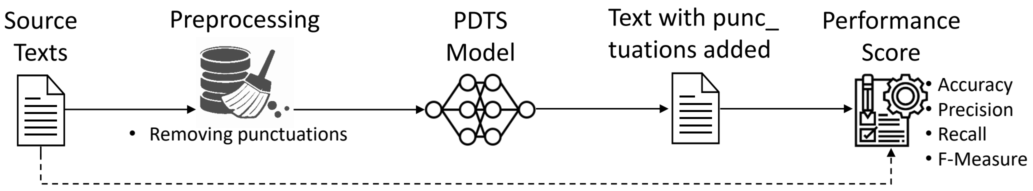


**Figure 2.** Description of our quantitative evaluation protocol.

## 4.1. Experiment Setup

### 4.1.1. Experiment Protocol and Metrics:

In the context of this paper, a proper protocol to experiment is to generate a representative test set from any available well-written corpus (i.e., contains grammatically correct punctuations). The thought is to treat the original texts, typically written by expert writers, as gold-standard references while removing all punctuations from them to form tests' inputs (i.e., mimic the transcribed audio-to-text documents). Then, comparing the tests' outputs (i.e., produced by PDTS) with the original gold references to estimate the generalization error. We have implemented this protocol (illustrated in Figure 2) and used four standard evaluation metrics (*accuracy, precision, recall* and *f-measure*) to measure the qualify performance of PDTS, expressed as follows:

$$\begin{aligned} accuracy &= \frac{(TP+TN)}{(TP+FP+FN+TN)}, \\ precision &= \frac{TP}{(TP+FP)}, \\ recall &= \frac{TP}{(TP+FN)}, \\ f\text{-}measure &= 2.\frac{precision\ .\ recall}{precision+recall} \end{aligned}$$

where TP (true-positive) represents correct predicted cases of punctuations; FP (false-positive) represents wrong predicted cases due to a mismatch of predicted punctuations with the already existing ones in the gold-standard references; TN (true-negative) represents wrong predicted punctuations due to their absence in the gold-standard references; and FN (false-negative) represents missed cases to predict the existing punctuations in the gold-standard reference.

### 4.1.2. Arabic Corpora:

We choose to experiment on a combination of two public Arabic corpora. They provide a variety of short-to-long texts, written in different styles and formats (i.e., in terms of the presence of diacritics and punctuations). The first corpus is ABMC[6] (Arabic in Business and Management Corpora), comprising 1200 documents (400 statements written by chairpersons and chief executive managers, besides other 800 news articles on the economic and stock market). The second corpus[7] comprises 55,000 formal and diacritized statements. Overall, we test our PDTS on 16,281 text examples (out of 56,200) after excluding documents that contain less than 50 words. We have categized these 16,281 text examples into short, mid (i.e., number of words between 168 and 512), and long based on their length, see their statistical descriptions in Table 3.

**Table 3.** Statistical descriptions of text examples used for evaluation.

| Document size | # documents | # sentences | # words |
|---|---|---|---|
| Small | 5427 | 6639 | 328872 |
| Medium | 5427 | 6807 | 491693 |
| Large | 5427 | 13404 | 1057131 |
| Total | 16281 | 26850 | 1877696 |

### 4.1.3. Baselines:

We compared the pre-trained mBERT model (i.e., used as a base model for constructing PDTS) against the state-of-the-art Arabic monolingual as well as multilingual MLM models. ArBERT [30], AraBERT [18], and CAMeLBERT [19] are Arabic-specific Transformer-based language models, trained with the same architecture as BERT$_{Base}$ configuration but on different datasets. ArBERT is trained on 61GB of MSA text, while AraBERT is trained on around 23GB of Arabic text (containing 70M sentences and 3B words). CAMeLBERT is trained[8] on the largest Arabic dataset, containing different dialectal corpora with a size of about 107GB. XLM-RoBERTa [33], however, is a state-of-the-art multilingual version of

[6] https://metatext.io/datasets/arabic-in-business-and-management-corpora-(abmc)
[7] https://github.com/AliOsm/arabic-text-diacritization
[8] The model is trained for 1m epochs, which took approximately 4.5 days.

**Table 4.** Performance comparison of PDTS with respect to different document sizes: small ($|X| < 168$), medium ($168 \leq |X| \leq 512$) and large ($512 < |X|$). The base model is *mBERT*, and $\alpha = 40$. For $\theta$, the best value found in all our experiments is around 0.4.

| | Accuracy | Precision | Recall | F-Measure |
|---|---|---|---|---|
| small | 0.86 | 0.74 | 0.87 | 0.79 |
| medium | 0.87 | 0.67 | 0.67 | 0.65 |
| large | 0.95 | 0.78 | 0.92 | 0.82 |
| avg. | 0.89 | 0.73 | 0.82 | 0.75 |

RoBERTa, which is pre-trained on 2.5TB of filtered CommonCrawl data that consists of 100 languages, including Arabic.

4.1.4. Implementation Details:

All base models (i.e., mBERT and other pre-trained baselines) are publicly available at the Hugging Face[9], under the model names: *'bert-base-multilingual-uncased'*, *'UBC-NLP/ARBERT'*, *'aubmindlab/bert-base-arabertv02'*, *'bert-base-arabic-camelbert-mix'*, *'xlm-roberta-large'*. We have used the PyTorch[10] framework to construct them and utilized Python-based NLP toolkits for text preprocessing, including NLTK[11] and CAMeL[12]. For computation, we used a PC machine equipped with Intel i9-CPU, 64G-RAM, and a single NVIDIA GeForce RTX3070 GPU.

### *4.2. Performance Evaluation*

The performance of PDTS concerning different document sizes is illustrated in Table 4. The overall prediction score looks generally high but relatively resembling between small vs. large document categories. Besides the lower scores observed with the medium document category, the obtained scores indicate that document size as a factor is independent of quality performance. This means that document size can affect only computational performance. Further, it should be pointed out that the *Accuracy* score presents the average height score of 0.89 (see Table 4), which results from having a high $TN$'s rate, not $TP$'s rate.

As mentioned, we introduced the parameter $\alpha$ to reduce computation costs when processing large document sizes. Thus, we conducted several experiments to analyze the sensitivity of $\alpha$ with respect to both the quality as well as the computational performance, reported in Table 5 and visualized in Figure 3.

Table 5 illustrates the performance of PDTS on six runs, each *with* and *without* using our linguistic rules (discussed in Table 2). Through those achieved scores, while paying more attention to the highest *f-measure* (i.e., $FM = 0.75$ ), we can observe that PDTS performs well and demonstrates the best results at $\alpha = 40$. The practical feasibility of applying our linguistic rules is evidently anticipated by an average of around 16% improvement (see the *acc, recall* and *f-measure* with using vs. without using rules). Moreover, while it is obvious that the execution time would increase by increasing $\alpha$, this notable improvement (i.e., anticipated to be achieved when applying our rules) is worth sacrificing a few seconds of performance. This is indicated by giving insight into the computational overhead (see execution time columns). In our experiments, the maximum computational overhead has reached 35 seconds at $\alpha = 160$ for processing 16281 documents.

In Table 6, we compare different pre-trained transformer-based models (i.e., alternative options rather than mBERT for basing our PDTS). Surprisingly, our considered mBERT outperforms all the state-of-the-art competitor models in the *f-measure* score, including monolingual Arabic models, for processing Arabic texts. Moreover, while ArBERT [30] is

9 https://huggingface.co/
10 https://pytorch.org
11 https://www.nltk.org/
12 https://camel-tools.readthedocs.io/

**Table 5.** Ablation study of our PDTS to assess the effectiveness of applying our linguistic rules, descried in Table 2. The base model is *mBERT* and $\theta = 0.4$. The best performance found is indicated by the asterisk *.

| | With excluding linguistic rules | | | | | With applying linguistic rules | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| $\alpha$ | Acc | P | R | FM | ET(Sec) | Acc | P | R | FM | ET(Sec) |
| 5 | 0.81 | 0.66 | 0.36 | 0.44 | 865 * | 0.93 * | 0.55 | 0.90 * | 0.64 | 886 |
| 10 | 0.82 | 0.73 | 0.44 | 0.51 | 973 | 0.91 | 0.61 | 0.84 | 0.69 | 983 |
| 20 | 0.80 | 0.77 | 0.44 | 0.54 | 1213 | 0.91 | 0.71 | 0.74 | 0.70 | 1234 |
| 40 * | 0.77 | 0.78 | 0.44 | 0.54 | 1980 | 0.89 | 0.73 | 0.82 | 0.75 * | 1987 |
| 80 | 0.77 | 0.80 * | 0.48 | 0.57 | 3181 | 0.87 | 0.69 | 0.74 | 0.68 | 3213 |
| 160 | 0.77 | 0.77 | 0.47 | 0.55 | 3507 | 0.86 | 0.63 | 0.69 | 0.64 | 3542 |

**Table 6.** Performance comparison of PDTS concerning the closely related state-of-the-art pre-trained base models (the asterisk * indicates the best performance), under the settings ($\alpha = 40$ and $\theta = 0.4$). Here, $\alpha$ is set according to the best *f-measure* score found; see Figure 3.

| Base model | Language | F-Measure | ET(Sec) |
|---|---|---|---|
| ArBERT [30] | Monolingual (Arabic) | 0.6235 | 1286.81 |
| AraBERT [18] | Monolingual (Arabic) | 0.3500 | 1173.98 |
| CAMeLBERT [19] | Monolingual (Arabic) | 0.2121 | 986.17 * |
| XLM-RoBERTa [33] | Multilingual | 0.3035 | 4916.51 |
| mBERT [16] | Multilingual | 0.7523 * | 1986.91 |

trained on the least amount of data compared to AraBERT [18] and CAMeLBERT [19], it gives next to the best *f-measure*. Concerning the execution time, CAMeLBERT [19] gives the best performance but performs poorly in *f-measure*.

### 4.3. Human Evaluation

To further evaluate the performance of PDTS, we conducted a qualitative investigation by eliciting human assessments with 54 text documents. These documents were randomly sampled from the 16,281 documents, where the punctuations were added by PDTS using mBERT besides the other competitor base models. We asked three expert consultants in Arabic linguistics (not authors of this paper) to assess these documents on three criteria using a five-point Likert scale (1-5):

- Adequacy (preservation of the source meaning),
- Contextual soundness (in terms of the avoidance of error cases, including (1) the generation of unimportant punctuations and/or (2) the failure in not generating important punctuations), and

**Table 7.** Human evaluation results for the three criteria: **A**dequacy, **C**ontextual soundness, and **G**rammaticality. Base pre-trained models with † are significantly different from PDTS's base model ‡, depending on a two-tailed independent student t-test, at $p < .05$. The asterisk * indicates the best result.)

| Base model | A | C | G | Avg. | p-value (t-value) |
|---|---|---|---|---|---|
| ArBERT [30] | 4.17 $*$ | 3.17 | 3.67 $*$ | 3.67 | .925156 (0.11) |
| AraBERT [18] | 3.33 | 2.50 | 3.17 | 3.00 | .069934 (2.46) |
| CAMeLBERT [19] | 2.67 | 1.50 | 2.17 | 2.11 | .002153 (7.03) † |
| XLM-RoBERTa [33] | 1.67 | 1.83 | 1.67 | 1.72 | .003126 (6.36) † |
| mBERT [16] | 3.83 | 3.33 | 3.30 | 3.50 | .468527(0.81) |
| mBERT [16] + Linguistic rules | 3.83 | 3.83 $*$ | 3.50 | 3.72 $*$ | ‡ |

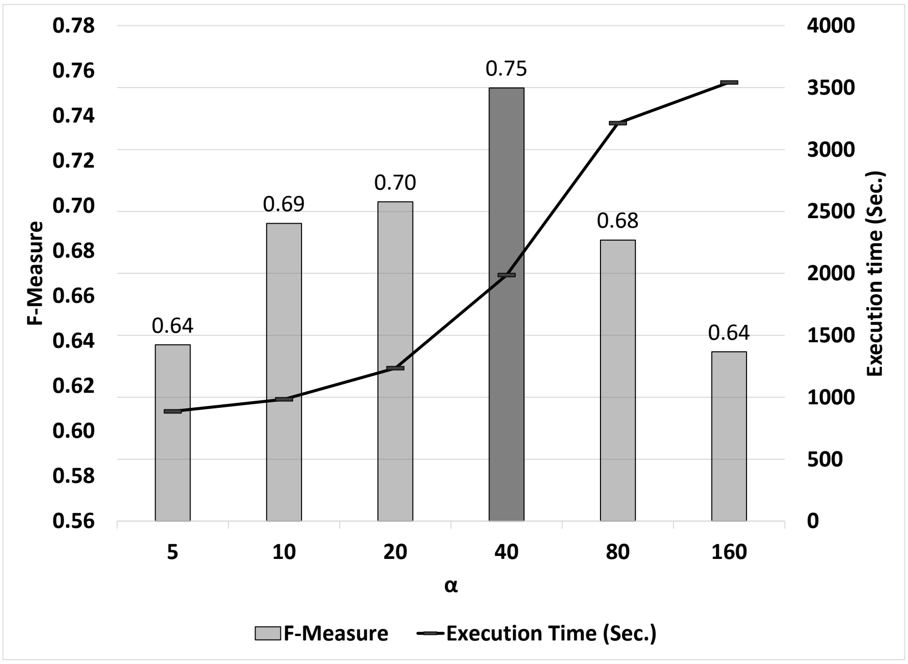


**Figure 3.** Impact of the parameters $\alpha$ (while $\theta = 0.4$) on the effectiveness of PDTS w.r.t the trade-off between the *f-measure* score and execution time.

- Grammaticality (the correctness of the generated punctuations).

Each expert is given 18 sample documents to be compared with their three original documents (denoted as gold-standard references). The average number of words in each sample is around 115 words. The experimental results, reported in Table 7, confirm that our PDTS (indicated by ‡) produces the best average ratings of about 3.72 and outperforms all competitor bases models in *contextual soundness* criteria.

We observe that employing ArBERT as an alternative-based model to mBERT is quite convenient, as ArBERT achieves the highest *Adequacy* and *Grammaticality* ratings (see 4.17 and 3.67 in the first row of Table 7). This observation is also indicted by the insignificant p-values obtained with ArBERT (and also AraBERT) against mBERT (including linguistic rules). Apart from the strengths and weaknesses of these base models, this experiment revealed that fine-tuning a comprehensive pre-trained multilingual model (such as mBERT) for the needed Arabic tasks (i.e., in such a low-resource language) is more straightforward and could produce significantly better performance than employing existing monolingual Arabic BERT-style models (particularly, CAMeLBERT that has been trained on the largest Arabic dataset). Nevertheless, not all existing multilingual models are suitable for Arabic as unexpectedly, the worst performance observed was with XLM-RoBERTa [33].

## 5. Conclusion and Future Work

This paper investigated whether fine-tuning a pre-trained language model for the *Fill-Mask* task can be beneficially applied in punctuation detection approaches. We proposed PDTS, a punctuation detector for text splitting, built on top of a multilingual BERT-based model, four generic linguistic rules, and a greedy algorithm for the best subset punctuation selection. We have showcased how PDTS is employed as a text tokenizer for unpunctuated documents, which can be highly useful to text simplification approaches that concentrate on handling large generated audio-to-text documents. Depending on our considered standard quantitative metrics and qualitative human-based evaluation protocols, experimental findings across two well-written Arabic corpora demonstrated that PDTS is practically effective in both performance quality and computational cost. For future work, we are striving to expand our PDTS toward supporting automatic text simplification and summarization problems.

## References

1. Alshanqiti, A.; Namoun, A.; Alsughayyir, A.; Mashraqi, A.M.; Gilal, A.R.; Albouq, S.S. Leveraging DistilBERT for Summarizing Arabic Text: An Extractive Dual-Stage Approach. *IEEE Access* **2021**, *9*, 135594–135607. doi:10.1109/ACCESS.2021.3113256.
2. Martin, L.; Fan, A.; de la Clergerie, É.; Bordes, A.; Sagot, B. MUSS: multilingual unsupervised sentence simplification by mining paraphrases. *arXiv preprint arXiv:2005.00352* **2020**.
3. Maddela, M.; Alva-Manchego, F.; Xu, W. Controllable text simplification with explicit paraphrasing. *arXiv preprint arXiv:2010.11004* **2020**.
4. Niklaus, C.; Cetto, M.; Freitas, A.; Handschuh, S. Context-Preserving Text Simplification. *arXiv preprint arXiv:2105.11178* **2021**.
5. Hao, T.; Li, X.; He, Y.; Wang, F.L.; Qu, Y. Recent progress in leveraging deep learning methods for question answering. *Neural Computing and Applications* **2022**, pp. 1–19.
6. Alonzo, O. The Use of Automatic Text Simplification to Provide Reading Assistance to Deaf and Hard-of-Hearing Individuals in Computing Fields. *SIGACCESS Access. Comput.* **2022**. doi:10.1145/3523265.3523268.
7. Gamal, D.; Alfonse, M.; Jiménez-Zafra, S.M.; Aref, M. Survey of Arabic Machine Translation, Methodologies, Progress, and Challenges. 2022 2nd International Mobile, Intelligent, and Ubiquitous Computing Conference (MIUCC). IEEE, 2022, pp. 378–383.
8. Zhou, M.; Duan, N.; Liu, S.; Shum, H.Y. Progress in Neural NLP: Modeling, Learning, and Reasoning. *Engineering* **2020**, *6*, 275–290. doi:https://doi.org/10.1016/j.eng.2019.12.014.
9. Khalifa, I.; Feki, Z.; Farawila, A. Arabic discourse segmentation based on rhetorical methods. *Int. J. Electric Comput. Sci* **2011**, *11*, 10–15.
10. Monroe, W.; Green, S.; Manning, C.D. Word segmentation of informal Arabic with domain adaptation. Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), 2014, pp. 206–211.
11. Souri, A.; Al Achhab, M.; El Mouhajir, B.E. A proposed approach for Arabic language segmentation. 2015 First International Conference on Arabic Computational Linguistics (ACLing). IEEE, 2015, pp. 43–48.
12. Elmadany, A.A.; Abdou, S.M.; Gheith, M. Turn Segmentation into Utterances for Arabic Spontaneous Dialogues and Instance Messages. *arXiv preprint arXiv:1505.03081* **2015**.
13. Abdelali, A.; Darwish, K.; Durrani, N.; Mubarak, H. Farasa: A fast and furious segmenter for arabic. Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: Demonstrations, 2016, pp. 11–16.
14. Eldesouki, M.; Samih, Y.; Abdelali, A.; Attia, M.; Mubarak, H.; Darwish, K.; Laura, K. Arabic multi-dialect segmentation: bi-LSTM-CRF vs. SVM. *arXiv preprint arXiv:1708.05891* **2017**.
15. Cheragui, M.A.; Hiri, E. Arabic Text Segmentation using Contextual Exploration and Morphological Analysis. 2020 2nd International conference on mathematics and information technology (ICMIT). IEEE, 2020, pp. 220–225.
16. Devlin, J.; Chang, M.; Lee, K.; Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR* **2018**, *abs/1810.04805*, [1810.04805].
17. Pires, T.; Schlinger, E.; Garrette, D. How multilingual is Multilingual BERT?, 2019, [arXiv:cs.CL/1906.01502].
18. Antoun, W.; Baly, F.; Hajj, H. AraBERT: Transformer-based Model for Arabic Language Understanding. LREC 2020 Workshop Language Resources and Evaluation Conference 11–16, 2020, p. 9.
19. Inoue, G.; Alhafni, B.; Baimukan, N.; Bouamor, H.; Habash, N. The Interplay of Variant, Size, and Task Type in Arabic Pre-trained Language Models. Proceedings of the Sixth Arabic Natural Language Processing Workshop; Association for Computational Linguistics: Kyiv, Ukraine (Online), 2021; pp. 92–104.
20. Li, J.; Sun, A.; Joty, S.R. SegBot: A Generic Neural Text Segmentation Model with Pointer Network. IJCAI, 2018, pp. 4166–4172.
21. Li, J.; Chiu, B.; Shang, S.; Shao, L. Neural text segmentation and its application to sentiment analysis. *IEEE Transactions on Knowledge and Data Engineering* **2020**.
22. Niklaus, C.; Cetto, M.; Freitas, A.; Handschuh, S. Transforming Complex Sentences into a Semantic Hierarchy. Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, 2019. 57th Annual Meeting of the Association for Computational Linguistics ; Conference date: 28-07-2019 Through 02-08-2019, doi:10.18653/v1/P19-1333.
23. Lukasik, M.; Dadachev, B.; Simoes, G.; Papineni, K. Text segmentation by cross segment attention. *arXiv preprint arXiv:2004.14535* **2020**.
24. Hananto, V.R.; Serdült, U.; Kryssanov, V. A Text Segmentation Approach for Automated Annotation of Online Customer Reviews, Based on Topic Modeling. *Applied Sciences* **2022**, *12*. doi:10.3390/app12073412.
25. Pak, I.; Teh, P.L. Text segmentation techniques: a critical review. *Innovative Computing, Optimization and Its Applications* **2018**, pp. 167–181.
26. Daroch, S.K.; Singh, P. An Analysis of Various Text Segmentation Approaches. Proceedings of International Conference on Intelligent Cyber-Physical Systems; Agarwal, B.; Rahman, A.; Patnaik, S.; Poonia, R.C., Eds.; Springer Nature Singapore: Singapore, 2022; pp. 285–302.

27. Lattisi, T.; Farina, D.; Ronchetti, M. Semantic Segmentation of Text Using Deep Learning. *Computing and Informatics* **2022**, *41*, 78–97.
28. Pasha, A.; Al-Badrashiny, M.; Diab, M.; El Kholy, A.; Eskander, R.; Habash, N.; Pooleery, M.; Rambow, O.; Roth, R. Madamira: A fast, comprehensive tool for morphological analysis and disambiguation of arabic. Proceedings of the ninth international conference on language resources and evaluation (LREC'14), 2014, pp. 1094–1101.
29. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. Advances in neural information processing systems, 2017, pp. 5998–6008.
30. Abdul-Mageed, M.; Elmadany, A.; Nagoudi, E.M.B. ARBERT & MARBERT: Deep Bidirectional Transformers for Arabic. Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers); Association for Computational Linguistics: Online, 2021; pp. 7088–7105. doi:10.18653/v1/2021.acl-long.551.
31. Alosh, M. *Using Arabic: A Guide to Contemporary Usage*; 'Using' Linguistic Books, Cambridge University Press, 2012.
32. Liu, H.; Yu, L. Toward integrating feature selection algorithms for classification and clustering. *IEEE Transactions on knowledge and data engineering* **2005**, *17*, 491–502.
33. Conneau, A.; Khandelwal, K.; Goyal, N.; Chaudhary, V.; Wenzek, G.; Guzmán, F.; Grave, E.; Ott, M.; Zettlemoyer, L.; Stoyanov, V. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116* **2020**.