Tokenization of Tunisian Arabic: A Comparison between Three Machine Learning Models

ASMA MEKKI, ANLP Research Group, MIRACL Lab., University of Sfax, Tunisia
INÈS ZRIBI, ANLP Research Group, MIRACL Lab., ISIMa, University of Monastir, Tunisia
MARIEM ELLOUZE and LAMIA HADRICH BELGUITH, ANLP Research Group, MIRACL Lab., University of Sfax, Tunisia

Tokenization represents the way of segmenting a piece of text into smaller units called tokens. Since Arabic is an agglutinating language by nature, this treatment becomes a crucial preprocessing step for many Natural Language Processing (NLP) applications such as morphological analysis, parsing, machine translation, information extraction, and so on. In this article, we investigate word tokenization task with a rewriting process to rewrite the orthography of the stem. For this task, we are using Tunisian Arabic (TA) text. To the best of the researchers' knowledge, this is the first study that uses TA for word tokenization. Therefore, we start by collecting and preparing various TA corpora from different sources. Then, we present a comparison of three character-based tokenizers based on Conditional Random Fields (CRF), Support Vector Machines (SVM) and Deep Neural Networks (DNN). The best proposed model using CRF achieved an F-measure result of 88.9%.

CCS Concepts: • **Human-centered computing** → **User models**; • **General and reference** → **Experimentation**; *Computing standards, RFCs and guidelines*

Additional Key Words and Phrases: Word tokenization, Tunisian Arabic, Arabic dialect, deep learning, SVM, CRF

ACM Reference format:
Asma Mekki, Inès Zribi, Mariem Ellouze, and Lamia Hadrich Belguith. 2023. Tokenization of Tunisian Arabic: A Comparison between Three Machine Learning Models. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.* 22, 7, Article 194 (July 2023), 19 pages.
https://doi.org/10.1145/3599234

1 INTRODUCTION

Tokenization is a common task used in **Natural Language Processing** (**NLP**). It is a fundamental step in both traditional NLP methods such as Count Vectorizer and Advanced Deep Learning-based architectures such as Transformers. Tokenization is a way of separating a textual fragment into smaller units called tokens. It is worth noting that tokens can be either words, characters, or sub-words. Hence, tokenization is the foremost step while modeling text data. It is applied on the

Authors' addresses: A. Mekki, M. Ellouze, and L. H. Belguith, ANLP Research Group, MIRACL Lab., University of Sfax, Route de l'Aéroport Km 0.5, Sfax, Sfax, 3029, Tunisia; emails: asma.elmekki.ec@gmail.com, mariem.ellouze@planet.tn, l.belguith@fsegs.rnu.tn; I. Zribi, ANLP Research Group, MIRACL Lab., ISIMa, University of Monastir, Avenue Taher Hadded, Monastir, Monastir, 5000, Tunisia; email: ineszribi@gmail.com.

2375-4699/2023/07-ART194 $15.00
https://doi.org/10.1145/3599234

target corpus to obtain tokens, which represents the input of many NLP tasks ranging from **Part-of-speech** (**POS**) tagging to semantic comprehension. Therefore, we tackle the task of automated word tokenization in this work.

Treatment of TA is a difficult and unclear. For example, **Tunisian Arabic** (**TA**) may contain up to four clitics attached in a single word (e.g., و+قالوا+ها+ثني, w+qAlwA+hA+$y, *And did they say it?*). Despite reducing the number of white-spaces that delimits the words in a script, it increases the size of the corresponding lexicon. Thus, the treatment of these words becomes ambiguous. However, most state-of-the-art NLP work in TA has had to manually tokenize their datasets. For instance, the parser's input must be tokenized in advance as [31, 33] processed to manually tokenize their TA dataset. Moreover, [34] showed that using a tokenized corpus enhances the outcome of sentence boundary detection by 10%. In this article, we focus on the tokenization task for TA. Several tools have been implemented to automate the task of tokenizing clitics for **Modern Standard Arabic** (**MSA**) as well as many Arabic dialects. However, to the best of the researchers' knowledge, there is no system that has been developed for TA. Therefore, researchers working on TA have to resort to experts for the manual annotation of corpora for all tasks that require tokenized input [10, 30, 31, 49, 51]. As a result, we do not find TA tokenization tools to be inspired by or to compare our methods. Moreover, the majority of the textual resources proposed for TA are not tokenized. We started by collecting TA textual data from different sources. Then, we proposed to compare the performance of three different models based on DNN [3], CRF++ [24], and SVM [44] using several TA corpora. These models are widely used for sequential tasks such as word tokenization [1, 5, 11, 38, 39]. Among these three models, CRF produced the best results. We also compared the tokenization outcomes of our model to the MSA tokenization tool[1] to highlight the interest in developing a TA-specific model.

The remainder of this article is structured as follows: Section 2 presents the related work. Section 3 describes TA and the challenges of TA tokenization. Section 4 outlines the data collected for this work. Section 5 explains our proposed method. Finally, Section 6 presents our experimental results, followed by the presentation of our evaluation results in Section 7.

2 RELATED WORK

Tokenization methods used for Arabic language can be classified according to its varieties (MSA or dialectal Arabic). In the remaining part of this section, the state-of-the-art contributions related to these two varieties of Arabic are presented.

2.1 MSA Tokenization

Benajiba and Zitouni [7] proposed a method based on a trigram language model to segment Arabic words. They generated a model based on an algorithm that produces all possible tokenizations by combining a set of prefixes, stem and a set of suffixes. Then, the tokenization that has the highest probability is selected. The probability is estimated using n-grams language model on the proposed morpheme sequences. The training corpus consists of 571,743 words extracted from **Arabic TreeBank** (**ATB**) 1, 2, and 3 [25]. The test corpus contains 42,591 words. It reached an accuracy of 98.1%. However, this method does not handle rewrite cases, and the findings will always contain spelling errors.

Stanford Word Segmenter system [38] is based on **Conditional Random Fields** (**CRF**) classifier and MSA and Egyptian Arabic treebanks [25–27]. The authors recommended a five-character window feature, N-grams including the current character and up to three previous characters and the Unicode of the character, whether the character is a punctuation or a number, and so

[1] https://camel.abudhabi.nyu.edu/madamira/.

on. They took into account only three cases of rewriting: (1) the transformation of the character (ل,[2] l) into (ال, Al) when it is followed by the prefix (ل, l) such as the word (للمدرسة, llmdrsp, *for school*) which transform to (لالمدرسة, l+Almdrsp). (2) the modification of the letter (ت, t) primarily (ة, p). For example, the word (سيارته, syArth, *his car*) which transform to (سيارة+ه, syArp+h) instead of (سيارت+ه, syArt+h). (3) the letter (ي, y) becomes (ى, Y) such as the word (عفينا, EfynA, *forgived*) which transform to (عفي+نا, EfY+nA). The F-measure value was 98.30% using 10,000 samples as a test set. However, the test corpus was relatively small compared to the related work.

Farasa [1] is an Arabic text processing toolkit that uses a dictionary of words and its diacritizations ordered according to their number of occurrences. They suggested to calculate conditional probabilities to predict clitics. Thus, they proposed a lexicon of automatically generated stems to identify valid stems. Farasa [1] used Gazetteer Lexicon [12] to check the possibility of a stem without a final suffix. They used SVM-rank [20] and ATB [25] to sort out possible ways to segment words to prefixes, stems, and suffixes. The accuracy is equal to 98.94%. However, the dictionaries and lexicons used cannot cover all MSA words.

Kastner et al. [22] proposed a Bayesian segmentation model for English and Arabic. In their first experiment, they experimented a Bayesian segmentation model comparing consonant-only representations with complete representations (containing both consonants and vowels). In the second experiment, they tested whether the proto-lexicon[3] helps to learn the real lexicon of the language focusing on consonantal roots or not. They further investigated whether the proto-lexicon supports the acquisition of probabilistic phonological models in Arabic or not. They used Gigaword [15], a newswire corpus (i.e., an electronically transmitted service providing last-minute news). The best F-measure value did not exceed 68.2% with consonant-only representation.

Almuhareb et al. [5] proposed a method for tokenizing Arabic words based on a deep neural network. They presented a method for word tokenization with a rewriting step to infer letters that are deleted or modified when the main word unit is attached to another unit. These letters were rewritten when the two units were separated following the segmentation. They used binary tags as indicators of segmentation positions (i.e., tag 1 is an indicator of the beginning of a new word (split) in a symbol sequence that does not include spaces and tag 0 is an indicator for any other case (unfractionated)). Almuhareb et al. [5] used data from AT [25] and its clitic segmentation scheme. The model was formed with the help of nine additional linguistic resources such as dictionaries, morphological analyzers, or rules. It reached an F1 value of 98.03% for the segmentation of Arabic words and 99% for the segmentation of Arabic words with rewriting. It experienced a low number of epochs between 3 and 10.

Some authors proposed tools that identify suffixes and prefixes during morphological analysis or POS tagging task. For the rule-based method, [4, 6, 18, 23, 46] are based on the use of lexicons and morphological rules. For the statistical method, several machine learning algorithms were employed such as KNN (k-nearest neighbor) algorithm [2], CRF [11], SVM [39], and so on. These tools were mainly proposed for MSA, [39] treated the Egyptian dialect in addition to MSA. However, no tools have been proposed to tokenize TA.

2.2 Dialectal Arabic Tokenization

Mohamed et al. [37] annotated a corpus of 320 Egyptian comments (20,022 words) gathered from social networks to create a tokenization model for Egyptian dialect. They used a letter

[2]The transliteration followed in this article: http://www.qamus.org/transliteration.htm.

[3]The proto-lexicon is built by interpreting everything that appears between the limits of the tokenized words.

classification system in which each letter is classified as a segment boundary or not. It is a memory-based learning method where instances during learning are stored in memory. When a new instance is encountered, the closest instance in memory is returned based on a measure of distance. The obtained result of evaluation is equal to 92% of precision. The dataset used in this work is very small and cannot be considered as a representative sample for the Egyptian dialect.

The morphological analyzer MADAMIRA [39] proposed two tokenization methods based on both treebanks [25, 27]. First, it segmented all clitics except the definite articles. As for the second method, it tokenized QU, CONJ, and PART clitics as well as all articles and enclitics. MADAMIRA disambiguates the analyses produced by the morphological analyzer. Then, the top-scoring analyses are used to tokenize the words under examination using morphological regeneration. The percentage of fully discretized correct words reached 86.3% for MSA and 83.2% for EGY. In addition, when MADAMIRA gave correctly all morphological characteristics (that corresponded exactly to the gold entry), the percentage of words was equal to 84.1% and 77.3% for MSA and EGY, respectively. Eldesouki et al. [14] compared two methods (Bi-LSTM-CRF and SVM) to segment four Arabic dialects (Egyptian, Levantine, Gulf, and Maghrebi). They used a limited learning data composed of 350 tweets for each of the four categories of dialectal Arabic and only 5,495 tokens for the category Maghrebi. We note that Eldesouki et al. [14] grouped Tunisian, Moroccan, and Algerian dialects in the category Maghrebi. The SVM classifier ranks the best possible segmentation for a word using various attributes. For the second method, they determined the best position to segment the words (sequence-to-sequence mapping). Each character is tagged with one of five labels that denote the segmentation decision boundaries: Beginning, Middle, End of a multi-character segment, Single-segment and Word border. Both methods yield between 90% and 95% accurate results for the different dialect categories. The category Maghrebi gives the lowest results (91.2% with SVM and 90.1% with Bi-LSTM-CRF). The size of the dataset used is very limited.

Among the morphological analyzers that have dealt with the tokenization of words, [19] adapted the rule-based morphological analyzer MAGEAD [17] to TA. The proposed analyzer treats only verbs. Furthermore, [51] updated the word tokenization rules as well as the list of affixes and clitics of the MSA morphological analyzer ÂńAl-Khalil» [8]. Moreover, [43] proposed a morphological analyzer based on a dictionary and grammar using NooJ linguistic platform [41].

3 TUNISIAN ARABIC

TA is the spoken dialect in Tunisia while MSA is its official language. Although it is mainly spoken, it is written in social networks, blogs, and some novels, as well as in comics, commercials and some newspapers and popular songs. TA was influenced by other languages such as Berber, French, MSA, Turkish, Italian, Maltese, and so on. [47]. This is the result of the position of Tunisia between the two continents (Africa and Europe) as well as the diversity of civilizations that ruled it and its openness to neighboring cultures. In addition, Tunisians alternate in their speeches between French and TA. Some intellectuals alternate between MSA and TA [28, 32].

With the development of communication technology, internet users increasingly benefit from social media websites by expressing their opinions and interacting with others. These online comments and texts are generally authored in dialectal Arabic. It is generally written using Arabic letters [21, 40]. Code-switching is present in social media where words from several languages (e.g., MSA, French) could be found in the same message. It may even include an intra-word alternation where the word starts in French and ends in TA. The French term could be employed in the word using Arabic letters (e.g., the word (برتاجوا, brtAjwA, *they shared*) starts with (برتج, brtj), which presents the French word "partage", and ends with the plural suffix (وا, wA)). This source is characterized by the existence of onomatopoeia, smileys, non-standard abbreviations, and so on.

TA is also used in radio conversations, news broadcasts, and so on. This source of dialect is generally characterized by a frequent code-switching with MSA with a relatively high frequency of MSA words. It is characterized by the use of a MSA-like lexicon and syntactic structure [40]. Moreover, TA is used in everyday communications, which is marked by a high rate of code-switching between TA, French, and other languages. It is characterized by the presence of disfluencies: filled pauses, self-corrections, repetitions, stuttering, incomplete words, and so on [50].

3.1 MSA vs. TA

Both varieties of Arabic (TA and MSA) share many similarities. They have an inflectional morphology that is distinguished by attachable clitics making it morphologically complex. However, the set of MSA clitics is different from that of TA.

- For TA clitic set, Tunisians do not use the interrogative proclitic (أ, >) and the proclitic of future (س, s). For example, the MSA word (أتعلمت, >tElmt, *did you learned*) is generally transformed into (تعلمتشي, tElmt$y) in TA by adding the interrogation clitic (شي, $y) at the end of the word. However, Tunisians can sometimes use MSA proclitic of future in **intellectualized dialect** (**ID**).
- Indeed, several enclitics are no longer used in TA such as dual enclitics (كما, kmA and ان, An) and the female enclitics (تن, tn, هن, hn and هما, hmA), and so on. For instance,the MSA dual noun (كتابان, ktAbAn, *two books*) turns into (زوز كتب, zwz ktb) in TA.
- Two proclitics (م, m and ع, E) are added to replace the prepositions (من, mn; على, ElY) when followed by a word starting with the definite article (ال, Al). For example, the MSA preposition + noun (من الدار, mn AldAr, from home) turns into (مالدار, mAldAr, *from home*) in TA.
- The third proclitic (خ, x) substitutes the verb (خلّي, xl y, let). Two enclitics are also appended, (ش, $) for negation and (شي, $y) for interrogation.

3.2 Challenges of TA Tokenization

The main challenges of word tokenization task come from the absence of explicit word boundaries in Arabic writing and the ambiguity of some alphabetic characters. For example, in Arabic language, there are no delimiters between the word and its prefixes (e.g., coordination conjunctions, definite articles, particles) and/or suffixes (e.g., prepositions, negation particles). They are written without any space between them. Therefore, a single word may need to be divided into a stream of textual content made up of prefix(es), stem, and suffix(es). For example, the word (وتعرفهمشي, wtErfhm$y, *and do you know them?*) which is actually a question that conveys a full meaning is composed of a conjunction (و, w), a verb (تعرف, tErf), a pronoun (هم, hm), and an interrogative particle (شي, $y). In addition, affixes might be usually classified into TA affixes and/or MSA affixes. Besides, the stem could be a TA stem or a MSA stem. Tunisians can mix combinations of prefixes in TA and stem in MSA, prefixes and stem in TA and suffixes in MSA, and so on. For instance, the word (سنبرتاجيها, snbrtAjyhA,*we will share it*) is composed of a prefix (س, s) specified for MSA followed by a purely TA stem and two MSA suffixes (ها + ي, y + hA). For example, the word (مالسيارة, mAlsyArp,*of the car*) is composed of the prefix (م, m) used only with TA and the

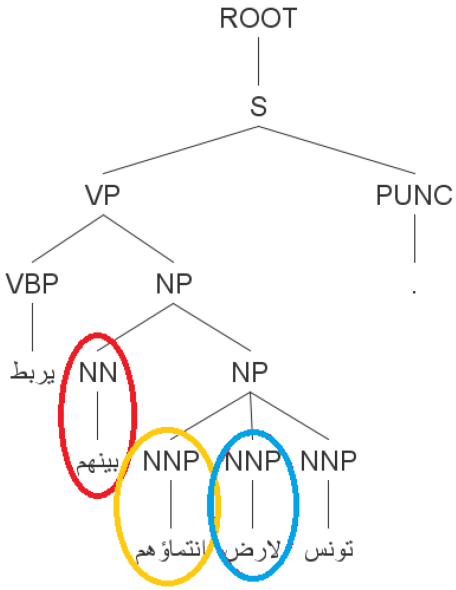

Fig. 1. Example of a non-tokenized sentence annotated by Stanford TUN parser.

noun (سيارة, syArp,*car*) which represents a MSA word. Therefore, using a MSA tokenizer will decrease the quality of tokenization. It is worthy to note that tokenization is considered as a base step for stemming, lemmatization, and further analysis such as parsing, POS tagging, classifying, and counting items for particular comprehension tasks. For instance, we used the Stanford TUN parser [31] to assign a non-tokenized and then a tokenized input to quantify the effectiveness of this task. Figure 1 shows a non-tokenized TA sentence analyzed by Stanford TUN. In fact, we notice that the words (انتماؤهم, AntmA&hm, *their belonging*), (بينهم, bynhm, *between them*) and لارض, lArD, *to the earth*) are all incorrectly annotated since the two clitic pronouns (هم, hm) and the proposition (ل, l) are agglutinated to the aforementioned words. On the other hand, when we input an already tokenized sentence into the Stanford parser, as shown in Figure 2, the system appropriately annotates clitic pronouns and the clause.

However, tokenization is potentially ambiguous, and it is difficult to determine whether the word is composed of several tokens and needs to be segmented or it is just a single token. For example, the preposition (ع, E) is attached to the following word. Despite that, we cannot tokenize all words starting with the character (ع, E). We should make the distinction between words starting with this character (e.g., عبد, Ebd, *person*) and those starting with the preposition which is written attached to the following word in TA (e.g., عالطاولة, EAlTAwlp, *on the table*). Indeed, suffixes are made up of a single-character (e.g., the negation clitic ش, $) or two characters (e.g., the interrogation clitic شي, $y). Moreover, it is extremely common to encounter the co-existence of several prefixes and suffixes in the same word. For example, by tokenizing the word (وكتبهمشي, wktbhm$y, *and did he write them?*) which contains a prefix, a stem, and two suffixes, it turns into و+كتب+هم+شي. Furthermore, internet users tend to accentuate words by repeating a letter several

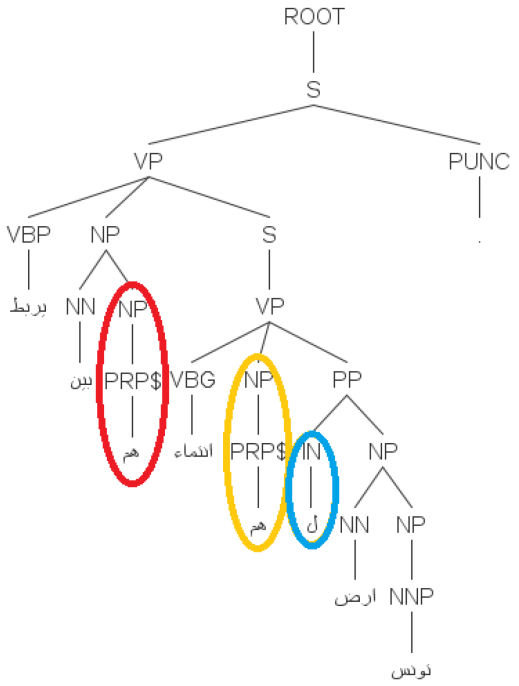

Fig. 2. Example of a tokenized sentence annotated by Stanford TUN parser.

times in succession in dialect derived from social media. This emphasis can cause problems in the detection of tokens. For instance, the word (اعطييينيي, AETyyynyyy,*give me*) contains two repeated letters in the stem (اعطيي, AETyyy) and the suffix (نيي, nyyy) which further complicates our task.

4 OUR PROPOSED METHOD

We proposed an automatic tokenization tool that segments words to generate the most helpful word tokenization model. First, we collected several corpora from multiple sources of TA. Then, a preprocessed step was proceeded to manually tokenize the gathered corpora. Using the prepared corpus, we compared three proposed models: (DNN [3], CRF++ [24], and SVM [44]). These three techniques are devoted for sequence labeling problems such as word tokenization. As noted in several related work [14, 34, 38], the tokenization improves the performance of following processing tasks. For SVM and CRF, we proposed three features to enhance the results. Otherwise, DNN automatically generates the features to use. Figure 3 presents the algorithm steps for DNN, SVM and CRF classification. However, our character-based method is designed for tokenizing words into: prefix(es) + stem + suffix(es). In many cases, the tokenization of words causes spelling errors in stems. For example, we may find these errors in plural verbs that usually end with (وا, wA). However, if the verb is attached to a suffix, we do not write the simple Alif (ا, A). Therefore, by tokenizing the suffix, the verb becomes incorrect according to the CODA-TUN orthographic convention. As far as the verb (كتبوه, ktbwh, *they wrote it*) is concerned, the tokenization of the suffix (ه, h) will generate a malformed verb (كتبو, ktbw) where we should add a simple Alif (ا, A) to be written as

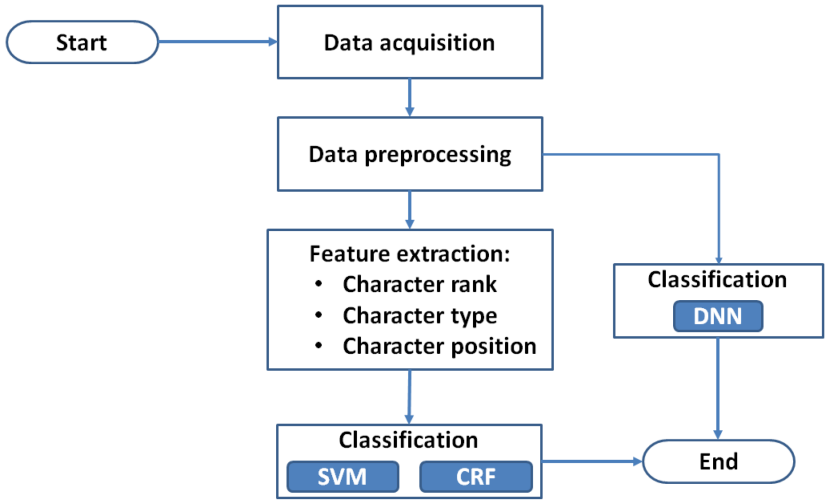

Fig. 3. Algorithm steps for the three models classification.

Table 1. Description of the Used Corpora

Corpus	Number of words
STAC	39,213
TTB	12,372
TMC	47,252
Story	3,740
TAD	151,598
Citations	6,189

(كتبوا, ktbwA). Also, when tokenizing the suffix (ها, hA, *her*) from the word (كرهبتها, krhbthA, *her car*), the stem (كرهبت, krhbt, *car*) will remain malformed. In this case, a rewriting process is needed. Thereby, word tokenization can lead to some errors that we should rewrite. Indeed, in this article, we propose a model for word tokenization of TA that rewrites errors emerging from segmentation.

4.1 Datasets

In this work, we used six available TA textual resources (Table 1). These corpora are written in Arabic letters and include all of the three TA registers as proposed in [34]: ID, **Social Media Dialect** (**SMD**), and **Spontaneous Dialect** (**SD**).

We take the opportunity of the existence of three different corpora for ID. The first corpus is the treebank TTB [31], which contains 12,372 words. The second corpus is «**Tunisian Media Corpus**» (**TMC**) [9]. It is collected from TV news and political debate broadcasts. It contains more than 47,000 words which is equivalent to 5 hours and 20 minutes of transcripts. We also used a story downloaded from the internet. This story is of a rather romantic and social type and different from the other two political corpora. It is composed of 3,740 words.

It should be noted that these corpora are already tokenized and normalized according to CODA-TUN orthographic convention [47].

Table 2. Statistics for TA Tokenization Corpus

	Train	Development	Test
Words	324,262	21,617	86,470
Letters	1,564,519	104,305	417,215

Table 3. List of Predefined TA Prefixes

Prefix	Transliteration	Translation
و	w	*and*
فـ	f	*and*
بـ	b	*with*
مـ	m	*from*
عـ	E	*on*
لـ	l	*to*
الـ	Al	*the*
كـ	k	*like*
هـ	h	*this*
خـ	x	*lets*

For SD, we used the transcribed corpus **Spoken Tunisian Arabic Corpus** (**STAC**) [48]. It contains 4 hours 50 minutes (7,134 sentences) of spontaneous TA speech enriched with disfluencies annotations. These audio records are collected from various TV channels and radio stations that discuss different themes (e.g., politics, social, religion, etc.).

We used the **Tunisian Arabic dialect** (**TAD**) corpus [45]. It is collected from Facebook comments, mobile phone messages, and so on. The corpus contains 151,598 words written in Arabic characters and normalized according to the CODA-TUN convention by [29, 34].

To enrich our corpus, we collected a list of 1,032 quotes[4] generally used by Tunisians to reinforce their ideas.

Our aim is to properly divide training, development, and test sets to evaluate our tokenization models. However, the corpus contains duplicates and it is quite difficult to balance the number of words that must be tokenized as well as the ones that do not contain affixes. We randomly selected 324,262 words for training, 21,617 for development and 86,470 for testing (Table 2).

The collected corpus is already normalized according to CODA-TUN orthographic convention [47]. However, it requires a preprocessing step to separate all tokens of the gathered corpora. TTB [31] does not necessitate this preparation step because it is already tokenized. TA clitics are limited. In the current study, we propose a predefined set of 11 prefixes (Table 3) and 11 suffixes (Table 4). In this step, two TA native speakers have segmented manually all proclitics and enclitics found in the corpus. For example, for the word (ويفهموه, wyfhmwh, *and they understand it*), we add a space after the coordination conjunction (و, w, *and*) and before the suffix (ه, h , his). The word turns into (و +يفهمو +ه, w yfhmw h). Indeed, the example below contains a plural verb, which needs

[4]https://ar.wikiquote.org/wiki/أمثال_تونسية.

Table 4. List of Predefined TA Suffixes

Suffix	Transliteration	Translation
ـش	$	*not*
ـشي	$y	*did you*
ـنا	nA	*our*
ـني	ny	*me*
ـه	h	*his / him*
ـها	hA	*her*
ـهم	hm	*their / them*
ـك	k	*your / you*
ـكم	km	*your / you*
ـي	y	*my*
ـيا	yA	*my*

to end with the letter «simple Alif» (ا, A) according to CODA-TUN [47]. Therefore, we proceed to a step of rewriting to correct the orthography of the stem.

4.2 Features

In this study, we proposed three features that were used for SVM and CRF models:

Character rank: This feature presents the rank of letters in the word. We start by giving 1 for the first letter, 2 for the second one until «n = length of the word» as a rank for the last letter of the word. This feature was applied by Reference [38].

Character type: We propose four types to categorize the characters of words:

— P: for characters that may belong to the list of prefixes (Table 3).
— S1: for characters that may belong to the list of first characters of suffixes (ن, n ; ش, $; ي, y).
— S2: for the simple Alef (ا, A), which can be in the second position of the TA suffixes.
— N: to annotate the rest of characters (ط, T; ر, r; ج, j; etc.).

Character position: We use this feature to define the position of the character in the word. The use of this feature is inspired by Reference [14]. It can be:

— B: the first letter in the word,
— I: a letter in the middle of the word,
— E: the last letter of the word.

4.3 Classes

We propose four classes for the tokenization of TA words with stem rewriting. The first class «0» indicates instances that do not require any processing. Class «1» refers to instances with simple tokenization. We propose two classes for instances that require a rewriting step. In this article, we rewrite two types of errors following the tokenization step. Therefore, we assign class 2 to add a (ا, A) before segmenting the suffix. For example, (جربوه, jrbwh, *they tried it*) is changed to (جربوا + ه, jrbwA + h). Moreover, class 3 segments the prefix «ل» and adds a simple Alif (ا, A)

at the beginning of the next stem (e.g., (للدار , lldAr, *to the home*) and turns into (ل + الدار, l + AldAr). Then, we segment the definite article (ال, Al) by giving class «1» to the instance «ل». In fact, we distinguish different kinds of mistakes occurring because of the tokenization procedure. We may denote the terms that end with (ت, t), (ا, A) and (ئ, }) or (ؤ, &) where the correct forms are respectively (ة, p), (ى, Y) and (ء, '). We ignore the other three possible rewritten forms because of their rarity in the corpus that does not exceed 0.19%. Thus, in the state-of-the-art, there is a gap concerning these three forms of rewriting [5, 13, 38, 39].

5 EXPERIMENTAL SETUP

In this section, we propose several experiments to improve the obtained results for the three applied algorithms (DNN, SVM, and CRF) using the development set. We have configured the parameters effect suggested for each algorithm.

5.1 DNN

In 2000, the term deep learning was used by [3] for the first time in the context of Artificial Neural Networks. In general, **Deep Neural Networks** (**DNN**) is a collection of machine learning methods that attempts to model using articulated architectures of various nonlinear transformations with a high level of data abstraction. It uses several layers to gradually extract features of a higher level from raw data. In this work, we applied Word2Vec [36]. It is a statistical method for efficiently learning a standalone word embedding from a textual data. We used the **Continuous Bag-of-Words** (**CBOW**) model as part of the Word2Vec method, which learns the embedding by predicting the current word based on its context [35]. The structure used of DNN is comparatively simple; it is a feedforward networks with sequential fully-connected hidden layers using the TensorFlow library developed by Google. We employ a Softmax activation function 1 and the default parametric values of Adam optimizer.

$$P(y = j|\Theta^{(i)}) = \frac{e^{\Theta^{(i)}}}{\sum_{j=0}^{k} e_k^{\Theta^{(i)}}}, where\ \Theta = \sum_{i=0}^{k} W_i X_i. \tag{1}$$

In this function, Θ represent a one-hot encoded matrix, where all values are marked 0 and a 1 is used to mark the position for the class label. **X** is a trained set, each with a set of weights **W**, are a class of **j**.

The Effect of the Number of Hidden Layers. In this experiment, we evaluate the effect of adding hidden layers to the neural network. Each time, we successively add hidden layers and compare the accuracy and loss values for word tokenization (Table 5). We compare the proposed networks with different number of layers (up to seven hidden layers) using the TensorFlow backend[5]. Our network relies on dense layers, which means deeply connected neural network layers, ensuring that each neuron receives input from all neurons in its previous layer. To keep a strategic distance from the co-adaptation, the «dropout» is performed on these layers, which randomly hides few neurons and their connections in the middle of the training process to prevent overfitting the network neurons [42]. Results show that four hidden layers give the highest accuracy and the lowest loss values. Therefore, in the remaining experiments, we use this architecture.

The Effect of the Batch Size. To adjust the model, we have configured the learning parameters. First, we start by testing multiple batch sizes (i.e., the number of training instances used in an iteration). The following three options are available:

[5]TensorFlow is an open source, end-to-end deep learning framework developed and maintained by Google.

Table 5. Accuracy and Loss Results for Each Architecture

Hidden layers	Loss	Accuracy
1	0.468	0.867
2	0.339	0.868
3	0.337	0.869
4	**0.336**	**0.870**
5	0.337	0.869
6	0.338	0.869
7	0.344	0.863

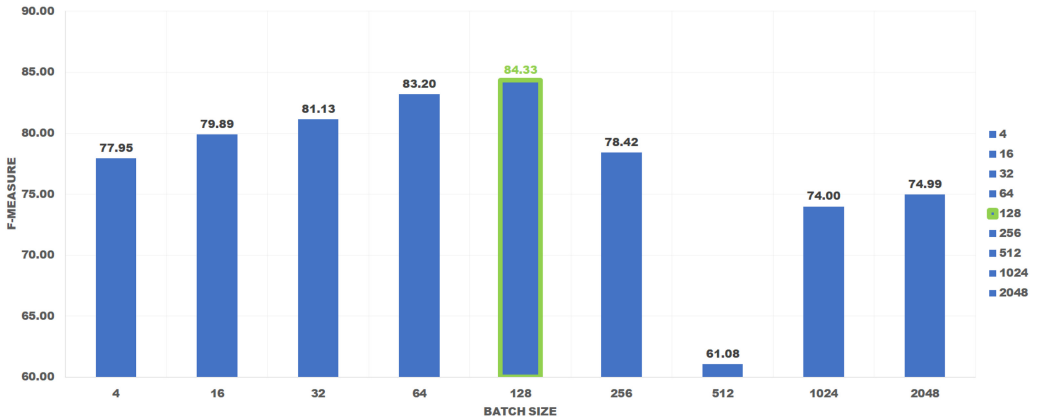

Fig. 4. **The effect of batch size**. The best tokenization results for the DNN experiment was obtained with a batch size of 128, using 100 epochs, which resulted in an F-measure of 84.33%.

(1) **The batch mode** where the batch size equals the total dataset, render iteration, and epoch equivalents.
(2) **The mini-batch mode** in which the batch size is less than the total size of the dataset but greater than one.
(3) **The stochastic mode** where the batch size is equal to one.

The parameters of the neural network and the gradient are adjusted after each sample. However, we tried to find the optimal batch size for word tokenization using our TA dataset.

The Effect of the Number of Epochs. In neural networks, the number of epochs is a hyper parameter of gradient descent that specifies the number of complete passes by the entire dataset. Training a neural network usually takes more than few epochs. Taking into account the fact of having «B» as the number of batches and «I» as the total number of instances in the training dataset, then there would be $\frac{I}{B}$ iterations to complete one epoch. Fortunately, it provides the network the possibility to readjust the model parameters according to previous data. As a result, the model would not be biased toward the final data points of the training. In our case, we made the network learn data up to 300 epochs. However, based on the experiment's findings, the number of epochs needed for our network is 100, with an F-measure of 84.33%.

Table 6 presents the final evaluation results using the development set.

Table 6. Final Evaluation Results Using the Development Set

	Recall	Precision	F-measure
DNN	80.03%	89.11%	84.33%
SVM	77.71%	88.57%	82.79%
CRF	**88.72%**	**87.9%**	**88.13%**

Table 7. The Evaluation Results for Each Kernel Function

Kernel function	Recall	Precision	F-measure
RBF (by default)	76.53%	87.73%	81.75%
Sigmoid	68.71%	78.12%	73.11%
Linear	71.9%	80.37%	75.9%
Polynomial	77.21%	88.04%	82.27%
Polynomial + Keywords	**77.71%**	**88.57%**	**82.79%**

5.2 SVM

The supervised machine learning **Support Vector Machines** (**SVM**) [44] are one supervised machine learning algorithm having the advantages of discriminative preparation, robustness, and the ability to manage a vast number of (overlapping) features with good performance. These machines are also indifferent to the number of examples (positive or negative) of learning data. SVM can be used for both regression and classification tasks [16]. We build an SVM model with a character-based method and the experiment was carried out on the different TA corpora using the function SVC of the Python library SVM. The objective here is to find a hyperplane in 3-dimensional space that distinctly classifies the data points.

The Effect of Parameter C. Parameter «C» regularizes the estimation by exchanging off the right classification of training instances against the maximization of the decision function's margin. We tried different penalty float values of this parameter. We obtained three thresholds (C = 6.8, C = 8.2, and C = 8.3) for the same value of F-measure, which is equal to 81.75%. However, the difference between them is in the confusion matrix of the models. In the first model, the number of instances which are correctly classified for «B-S» class is slightly higher than the second and third models (i.e., using the parameter C = 8.2 and C = 8.3). Therefore, we decided to keep penalty parameter C = 6.8.

The Effect of Kernel Function. We carried out several experiments to improve the SVM model as much as possible. We specified the best kernel function to be used in this algorithm (Table 7). The best evaluation result was the Polynomial kernel function + Keywords (d = 3 and r = 3). These functions are used to map the original dataset into a higher dimensional space to make it a linear dataset. Thus, both **Radial Basis Function** (**RBF**) and Polynomial functions are particularly helpful when the data-points are linearly inseparable. Therefore, as it could be seen from Table 7, these SVM-Kernels give the best F-measure results in our case.

After selecting «Polynomial» as a kernel function 2, We also tested the effect of parameterizing the two keywords related to this function: «d» as the Degree and «r» as the Coefficient. Using the development set, the best F-measure value obtained employs the same default degree (d = 3) and a coefficient equal to 3 (see Table 6). The best evaluation results was using the CRF model.

$$(\gamma\langle x, x'\rangle + r)^d. \tag{2}$$

Table 8. Detailed F-measure Results According to the Contextual Window and the Number of Features Effects

Context window	One feature	Two features	Three features
0	46.11%	46.11%	46.11%
1	78.35%	77.05%	79.2%
2	81.99%	82.4%	83.21%
3	83.29%	83.06%	84.73%
4	83.89%	84%	85.42%
5	84.29%	83.96%	85.43%
6	**85.85%**	**85.26%**	**87.08%**

5.3 CRF

CRFs are a class of statistical modeling methods often used for sequence prediction. They use contextual information from following and/or previous labels to raise the amount of knowledge that the model requires to make a successful prediction. It is a class of graphical models with an undirected graph between random variables. The structure of this graph specifies the dependence or independence of the random variables. Thus, when we condition the graph on X globally (i.e., when the values of random variables in X are fixed or given) all random variables in set Y follow the Markov property (see Equation (3)). Reference [24] defined a CRF on observations X, random variables Y and $Y_u \sim Y_x$ signifies that Y_u and Y_x are neighbors in the graph. CRFs consider their applications in the identification of named entities, part of speech tagging, segmentation, object detection, and so on. We prepared the template file that includes the features specified for training and testing. It serves in generating n-gram features from the feature columns. The type of the template file is defined according to variables U and B, which indicates Unigram and Bigram, respectively. Our template line starts with B. It defines the current and previous labels generating n*n weights in the model. Indeed, the template is enriched by probabilities between features to further refine the result.

$$p(Y_u/X, Y_v, u \neq v) = p(Y_u/X, Y_x, Y_u \sim Y_x). \tag{3}$$

The Effect of Contextual Window and the Number of Features. We performed a series of experiments to study the effect of increasing the context window while adding features for the tokenization method with a rewriting step (Table 8). The best F-measure result was using a contextual window of 6. We tested the effect of the context window from 0 to 6. For example, for the context window «three», we check the current character, three characters before and three characters after this character. The longest word recorded in our corpus contains 12 characters (الدبلوماسيين, AldblwmAsyyn, *diplomats*). The best F-measure value recorded is 85.85% with a +/- 6 characters context and an improvement of almost 40% compared to a context window of 0.

We added the attribute «Position of the character» to the second experiment. Indeed, the result has not practically changed. Moreover, F-measure values increase or decrease slightly throughout the experiment. In the experiments performed using the two features combined, we noticed that the result was slightly better using the context windows 2 and 4. The results were equal without the use of the context window. For the rest of the experiments, the result using a single feature were higher. However, the best result obtained in this section was using the three attributes proposed in this article. Thus, as for this experiment, F-measure values increased successively until reaching the value of 87.08% with an improvement of 41% compared to a context window equal to 0 and 1.23% and compared to the result of the experiment result using only one feature. The first experiment in Table 9 presents the precision, recall, and F-measure values. The best result found using the development set.

Table 9. Best Evaluation Results for the Contextual Window and the Number of Features Effects for the First Experiment Using the Contextual Window and Number of Features and with Parameter C for the Second Experiment

	Recall	Precision	F-measure
Contextual window and number of features	88.4%	86.05%	87.08%
Parameter C	**88.72%**	**87.9%**	**88.13%**

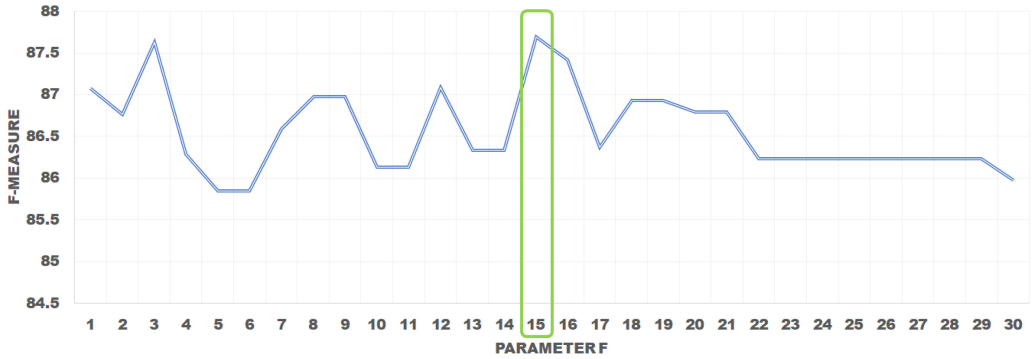

Fig. 5. **Setting the parameter F.** The best tokenization results for the parameter F of the CRF experiment was obtained with a value of F=15, which resulted in an F-measure of 87.67%.

Effect of CRF++ Parameters. In this sub-section, we present our experiments for improving the CRF model using the parameters (F and C) proposed by the toolkit CRF++ [24].

Parameter F. To improve the F-measure result, we experimented the effect of parameter F (see Figure 5). It sets the cut-off threshold for the features. For example, if the parameter F is equal to 5, CRF++ uses the features that occur no less than 5 times in the given training data. We set the value of parameter F to 15 (best F-measure result).

Parameter C. After setting parameter F, we propose to experiment with the effect of parameter C. This parameter helps the overfitting and underfitting to even out. CRF tends to respond more to the training corpus if we increase its value. Indeed, we tested the maximum of float values that are possible between C = 1 and C = 50. We obtained only five values slightly higher than the results of the previous experiments. However, Table 9 (the second experiment) shows the recall, precision, and F-measure values of the best result found. As a result, both parameters contribute to increase the F-measure value with 1.05%. In Table 6, we present the final evaluation results of the CRF model using the development set.

6 EVALUATION RESULTS

We picked the three above-mentioned learning algorithms since we found them interesting and pertinent to our learning issue. However, by examining the results obtained using the three algorithms (see Table 6), we found that the CRF approach is the most useful technique that gives the best result. This approach proves to be the most practical approach to improve word tokenization results, such as the work of Refs. [11, 38].

In this section, the results of recall, precision and F-measure are presented. We used the test set composed of 86,470 tokens. MADAMIRA is one of the most available well-known language analysis platforms. Therefore, we tested our test set on its MSA word tokenization Tool. Indeed, this

Table 10. Final Evaluation Results Using the Test Set

	Recall	Precision	F-measure
TA Baseline	46.07%	47.45%	46.75%
MSA Baseline (MADAMIRA)	63.53%	78.84%	70.36%
Final model	**89.33%**	**88.47%**	**88.9%**

evaluation yielded a 39% error rate. For instance, MADAMIRA tokenizes the personal pronouns (احنا, AhnA and نحنا, nhnA, *we*) to (احـنا, Ah+nA and نحـنا, nh+nA), respectively. Furthermore, it does not recognize TA-specific clitics such as (شي ، ش, $, $y). Our model outperforms this result by 27.47%. Indeed, the result found shows the importance of developing a TA-specific tool (see Table 10). The best result found using the test set.

To the best of the researchers' knowledge, this work is the first attempt to create a tokenization system for TA. Therefore, we cannot compare the performance of our system with a TA baseline from the literature. Thus, we present in Table 10 the evaluation results using the test set applied on the basic fitted CRF model without any contextual window or any additional parameters and using only the first feature (Character Rank). By comparing the two results, we notice that the elaborate experiments attributed to a significant improvement of the final result, reaching more than 42%. Furthermore, we got a comparable tokenization system with the state-of-the-art. For dialectal Arabic works, our method went beyond that of Reference [39] by 11.6%. Otherwise, the findings of Refs. [37] and [14] are higher, respectively, by 2.3% and 3.1%. However, the datasets that they used represent only 2% and 8%, respectively, of our corpus size. This represents a very important factor especially since we (i.e., [14, 37] and our proposed method) use learning methods. Moreover, to the best of the researchers' knowledge, the current study is the first one that proposes a method of tokenization with rewriting, among all works proposed for Arabic dialects. It is frequently used for MSA such as Refs. [5, 38].

We examined the cases of analysis failure to understand their causes. For example, we found some samples that do not need to be tokenized such as the word (خرج, xrj, *went out*) that is transformed to خر + ج. The model considers the final letter ج as a suffix. However, it does not belong to the list of clitics. Moreover, the word (قالوها, qAlwhA, *they said it*) is transformed into ق + لوا + ها. The word is divided into three parts: a prefix (ق, q), a stem (لوا, lwA) and a suffix (ها, hA). Although the suffix is well annotated, the model assigned the class «tokenized» for the first letter (ق, q) despite the fact that it is not on the list of possible prefixes. Furthermore, almost 40% of the incorrectly classified instances as class 1 (tokenize instance) are for the characters (ل, l) and (و, w). Although these letters are in the prefixes list, the words they contain should not be tokenized. For example, the word (لحق lHk, *catch*) transforms to (لـحق, l+Hk). Actually, this tokenization could be considered correct since the word (حق, Hk, *right*) already exists in TA. However, the word has an entirely different meaning in the context of the sentence proposed in the test set. Therefore, it must not be tokenized.

7 CONCLUSION AND FUTURE WORKS

In this article, we compared three character-based models for automatic tokenization system for TA. To the best of the researchers' knowledge, this system is the first word tokenization system for TA. We collected a varied corpus in order to cover the three registers of dialect (ID, ST, and SMD).

Then, we proceeded to tokenize all prefixes and suffixes of the gathered corpora. All TA models showed an encouraging result with the best F-measure value equal to 88.9% using a CRF model.

We aim in our future work to add morphosyntactic features. Moreover, we will increase the size of the corpus in order to maximize the coverage of TA lexicon. Based on the evaluation results, we aspire to test in the future a hybrid method that can correct the failure cases detected. We would like also to do an extrinsic evaluation by incorporating the model proposed in this article in other linguistic tools (e.g., morphological analyzer).

REFERENCES

[1] Ahmed Abdelali, Kareem Darwish, Nadir Durrani, and Hamdy Mubarak. 2016. Farasa: A fast and furious segmenter for arabic. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*. 11–16.

[2] Muhammad Abdul-Mageed, Mona Diab, and Sandra Kübler. 2013. ASMA: A system for automatic segmentation and morpho-syntactic disambiguation of modern standard Arabic. In *Proceedings of the International Conference Recent Advances in Natural Language Processing RANLP 2013*. INCOMA Ltd. Shoumen, BULGARIA, Hissar, Bulgaria, 1–8.

[3] Igor N. Aizenberg, Naum N. Aizenberg, and Joos Vandewalle. 2000. Multiple-valued threshold logic and multi-valued neurons. In *Proceedings of the Multi-Valued and Universal Binary Neurons*. Springer, 25–80.

[4] Ahmad Al-taani and Salah Abu Al-rub. 2009. A rule-based approach for tagging non-vocalized Arabic words. *The International Arab Journal of Information Technology* 6, 3 (2009), 320–328.

[5] Abdulrahman Almuhareb, Waleed Alsanie, and Abdulmohsen Al-thubaity. 2019. Arabic word segmentation with long short-term memory neural networks and word embedding. *IEEE Access* 7 (2019), 12879–12887. https://ieeexplore.ieee.org/document/8620203.

[6] Shihadeh Alqrainy, Hasan Muaidi AlSerhan, and Aladdin Ayesh. 2008. Pattern-based algorithm for Part-of-Speech tagging Arabic text. In *Proceedings of the 2008 International Conference on Computer Engineering Systems*. 119–124. DOI: https://doi.org/10.1109/ICCES.2008.4772979

[7] Yassine Benajiba and Imed Zitouni. 2010. Arabic word segmentation for better unit of analysis. In *Proceedings of the International Conference on Language Resources and Evaluation, LREC 2010*. Valletta, Malta.

[8] Abderrahim Boudlal, Abdelhak Lakhouja, Azzedine Mazroui, Abdelouafi Meziane, Mohamed Ould Abdallahi Ould Bebah, and Mohamed Shoul. 2010. Alkhalil Morpho Sys: A morphosyntactic analysis system for Arabic texts. In *Proceedings of the ACIT2010*. Riyadh, Saudi Arabia.

[9] Rahma Boujelbane, Mariem Ellouze, Frédéric Béchet, and Lamia Belguith. 2014. De l'arabe standard vers l'arabe dialectal: Projection de corpus et ressources linguistiques en vue du traitement automatique de l'oral dans les médias tunisiens. *TAL. 2. Traitement Automatique du Langage Parlé* 55 (2014), 73–96. https://hal.science/halshs-01193325/.

[10] Rahma Boujelbane, Mariem Mallek, Mariem Ellouze, and Lamia Hadrich Belguith. 2014. Fine-grained POS tagging of Spoken Tunisian Dialect Corpora. In *Proceedings of the Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Springer, 59–62.

[11] Kareem Darwish, Ahmed Abdelali, and Hamdy Mubarak. 2014. Using stem-templates to improve Arabic POS and gender / number tagging. In *Proceedings of the 9th International Conference on Language Resources and Evaluation (LREC'14)*. European Language Resources Association (ELRA), 2926–2931.

[12] Kareem Darwish, Walid Magdy, and Ahmed Mourad. 2012. Language processing for Arabic microblog retrieval. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*.

[13] Mona Diab. 2009. Second generation AMIRA tools for Arabic processing: Fast and robust second generation Amira tools for Arabic processing: Fast and robust Tokenization, POS tagging, and base phrase chunking. In *Proceedings of the 2nd International Conference on Arabic Language Resources and Tools*.

[14] Mohamed Eldesouki, Younes Samih, Ahmed Abdelali, Mohammed Attia, Hamdy Mubarak, Kareem Darwish, and Kallmeyer Laura. 2017. Arabic multi-dialect segmentation: bi-LSTM-CRF vs. SVM. *CoRR* abs/1708.05891 (2017). http://arxiv.org/abs/1708.05891

[15] David Graff. 2003. Arabic gigaword corpus. *Philadelphia, PA: Linguistic Data Consortium* (2003).

[16] Steve R. Gunn. 1998. Support vector machines for classification and regression, technical report. *Southampton, England: Faculty of Engineering, Science and Mathematics, School of Electronics and Computer Science, University of Southampton* 14, 1 (1998), 5–16.

[17] Nizar Habash and Owen Rambow. 2006. MAGEAD: A morphological analyzer and generator for the Arabic dialects. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL*. Sydney, Australia, 681–688.

[18] Meryeme Hadni, Said Alaoui Ouatik, Abdelmonaime Lachkar, and Mohammed Meknassi. 2013. Hybrid part-of-speech Tagger for non-vocalized Arabic text. *International Journal on Natural Language Computing (IJNLC)* 2, 6 (2013), 1–15.
[19] Ahmed Hamdi, Rahma Boujelbane, Nizar Habash, and Alexis Nasr. 2013. The effects of factorizing root and pattern mapping in bidirectional Tunisian - standard Arabic machine translation. In *Proceedings of the MT Summit 2013*. France.
[20] Thorsten Joachims. 2016. Training linear SVMs in linear time. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD)*.
[21] Anna Kashina. 2020. Case study of language preferences in social media of Tunisia. *Advances in Social Science, Education and Humanities Research* 489 (2020), 111–115. https://www.atlantis-press.com/proceedings/icdatmi-20/125948610.
[22] Itamar Kastner and Frans Adriaans. 2018. Linguistic constraints on statistical word segmentation: The role of consonants in Arabic and English. *Cognitive Science* 42, S2: Special Issue: Word Learning and Language Acquisition (2018), 1–25. DOI : https://doi.org/10.1111/cogs.12521
[23] Shereen Khoja. 2001. APT : Arabic part-of-speech tagger. In *Proceedings of the Student Work. NAACL*. 20–25.
[24] John Lafferty and Andrew Mccallum. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data conditional random fields: Probabilistic models for segmenting and. In *Proceedings of the 18th International Conference on Machine Learning, ICML*, Vol. 1. 282–289.
[25] Mohamed Maamouri, Ann Bies, and Tim Buckwalter. 2004. The Penn Arabic Treebank: Building a large scale annotated Arabic corpus. In *Proceedings of the NEMLAR Conference on Arabic Language Resources and Tools*. Cairo, Egypt.
[26] Mohamed Maamouri, Ann Bies, and Seth Kulick. 2012. Expanding Arabic Treebank to speech: Results from broadcast news. In *Proceedings of the LREC*. Citeseer, 1856–1861.
[27] Mohamed Maamouri, Ann Bies, Seth Kulick, Sondos Krouna, Dalila Tabassi, and Michael Ciul. 2012. Egyptian Arabic treebank DF Part 2 V2.0. In *Proceedings of the LDC Catalog Number LDC2012E98*.
[28] Salah Mejri, Mosbah Said, and Inès Sfar. 2009. Pluringuisme et diglossie en Tunisie. *Synergies Tunisie* 1 (2009), 53–74. https://gerflint.fr/Base/Tunisie1/salah1.pdf.
[29] Asma Mekki, Inès Zribi, Mariem Ellouze, and Lamia Hadrich Belguith. 2022. Sarcasm detection in Tunisian social media comments: Case of COVID-19. *Language Resources and Evaluation* 56, 1 (2022), 44–51. https://doi.org/10.1007/s10579-021-09538-4
[30] Asma Mekki, Inès Zribi, Mariem Ellouze, and Lamia Hadrich Belguith. 2022. Sarcasm detection in Tunisian social media comments: Case of COVID-19. In *Foundations of Intelligent Systems*. Michelangelo Ceci, Sergio Flesca, Elio Masciari, Giuseppe Manco, and Zbigniew W. Raś (Eds.), Springer International Publishing, Cham, 44–51.
[31] Asma Mekki, Inès Zribi, Mariem Ellouze, and Lamia Hadrich Belguith. 2020. Treebank creation and parser generation for Tunisian social media text. In *Proceedings of the 17th ACS/IEEE International Conference on Computer Systems and Applications AICCSA 2020*. IEEE.
[32] Asma Mekki, Inès Zribi, Mariem Ellouze Khmekhem, and Lamia Hadrich Belguith. 2018. Critical description of TA linguistic resources. In *Proceedings of the 4th International Conference on Arabic Computational Linguistics (ACLing 2018) & Procedia Computer Science, November 17-19 2018*. Dubai, United Arab Emirates.
[33] Asma Mekki, Inès Zribi, Mariem Ellouze Khemakhem, and Lamia Hadrich Belguith. 2017. Syntactic analysis of the Tunisian Arabic. In *Proceedings of the International Workshop on Language Processing and Knowledge Management*.
[34] Asma Mekki, Inès Zribi, Mariem Ellouze Khemakhem, and Lamia Hadrich Belguith. 2021. Sentence boundary detection of various forms of Tunisian Arabic. *Language Resources and Evaluation* (2021).
[35] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems* 26 (2013). https://papers.nips.cc/paper_files/paper/2013/hash/9aa42b31882ec039965f3c4923ce901b-Abstract.html.
[36] Tomáš Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 746–751.
[37] Emad Mohamed, Behrang Mohit, and Kemal Oflazer. 2012. Annotating and learning morphological segmentation of Egyptian colloquial Arabic. In *Proceedings of the Language Resources and Evaluation (LREC 2012)*. 873–877.
[38] Will Monroe, Spence Green, and Christopher D. Manning. 2014. Word segmentation of informal Arabic with domain adaptation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics. DOI : https://doi.org/10.3115/v1/P14-1
[39] Arfath Pasha, Mohamed Al-badrashiny, Mona Diab, Ahmed El Kholy, Ramy Eskander, Nizar Habash, Manoj Pooleery, Owen Rambow, and Ryan M. Roth. 2014. MADAMIRA : A fast, comprehensive tool for morphological analysis and disambiguation of Arabic. In *Proceedings of the 9th International Conference on Language Resources and Evaluation (LREC'14)*. 1094–1101.
[40] Lotfi Sayahi. 2014. *Diglossia and Language Contact: Language Variation and Change in North Africa*. Cambridge University Press.

[41] Max Silberztein. 2005. NooJ: A linguistic annotation system for corpus processing. In *Proceedings of HLT/EMNLP 2005 Interactive Demonstrations*. 10–11.

[42] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15, 1 (2014), 1929–1958.

[43] Roua Torjmen and Kais Haddar. 2018. Morphological aanalyzer for the Tunisian dialect. In *International Conference on Text, Speech, and Dialogue (TSD 2018)*. Springer, Cham, 180–187. https://doi.org/10.1007/978-3-030-00794-2_19

[44] Vladimir N. Vapnik. 1995. *The Nature of Statistical Learning Theory*. Springer-Verlag New York, Inc., New York, NY, USA.

[45] Jihene Younes, Hadhemi Achour, and Emna Souissi. 2015. Constructing linguistic resources for the Tunisian dialect using textual user-generated contents on the social web. In *Current Trends in Web Engineering - 15th International Conference, ICWE 2015 Rotterdam, The Netherlands*. 3–14.

[46] Chiraz Ben Othman Zribi, Aroua Torjmen, and Mohamed Ben Ahmed. 2007. A multi-agent system for POS-tagging vocalized Arabic texts. *The International Arab Journal of Information Technology* 4, November 2007 (2007), 322–329.

[47] Inès Zribi, Rahma Boujelbane, Abir Masmoudi, Mariem Ellouze, Lamia Hadrich Belguith, and Nizar Habash. 2014. A conventional orthography for Tunisian Arabic. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014), Reykjavik, Iceland, May 26-31, 2014*. European Language Resources Association (ELRA), 2355–2361.

[48] Inès Zribi, Mariem Ellouze, Lamia Hadrich Belguith, and Philippe Blache. 2015. Spoken Tunisian Arabic corpus STAC: Transcription and annotation. *Research in Computing Science* 90 (2015).

[49] Inès Zribi, Mariem Ellouze, Lamia Hadrich Belguith, and Philippe Blache. 2017. Morphological disambiguation of Tunisian dialect. *Journal of King Saud University - Computer and Information Sciences* 29, 2 (2017), 147–155. Arabic Natural Language Processing: Models, Systems and Applications.

[50] Inès Zribi, Inès Kammoun, Mariem Ellouze, Lamia Hadrich Belguith, and Philippe Blache. 2016. Sentence boundary detection for transcribed Tunisian Arabic. In *12th Edition of the Konvens Conference*. Bochum, Germany.

[51] Inès Zribi, Mariem Ellouze Khemakhem, and Lamia Hadrich Belguith. 2013. Morphological analysis of tunisian dialect. In *Sixth International Joint Conference on Natural Language Processing, IJCNLP 2013, Nagoya, Japan, October 14–18, 2013*. 992–996.

Received 15 May 2021; revised 30 August 2022; accepted 19 May 2023