Jordanian Journal of Computers and Information Technology (JJCIT), Vol. 11, No. 04, December 2025.

LIGHT-WEIGHT, SEMI CONTEXT-FREE, RULE-BASED ARABIC TEXT CLASSIFIER FOR POS TAGGING

Bilal Alqduah[1], Mohanad Alhasanat[1], Abdullah Alhasanat[1] and Hatem Alqudah[2]

(Received: 16-Jun.-2025, Revised: 12-Jul.-2025 and 12-Aug.-2025, Accepted: 13-Sep.-2025)

ABSTRACT

In this research, we address the challenges associated with part-of-speech (*POS*) *tagging and morphological classification of Arabic text where word structure is the subject of study.. Our focus is on Classical Arabic* (*CA*) *and Modern Standard Arabic (MSA), where the text is typically vocalized and includes diacritics on most letters. Our proposed classification method does not require a lexicon, stemming processes, or artificial-intelligence techniques. The goal is to minimize the resources needed for classifying Arabic text. This method is based on the principle that each verb in the Arabic language adheres to a specific pattern, we refere to as* (*wazn* وزن *or taf῾īl* تفعيل)*, that can be utilized to identify a word. The classification process is governed by a finite state machine, which is translated into regular expressions. Each verb tense is represented by a set of regular expressions* (*REs*)*. The order in which these regular expressions are processed is crucial for the accuracy of the results. Whenever a match is found, the word is marked to prevent further matches. The proposed method is lightweight and functions as a best-effort classifier, assigning the closest match as a tag. In terms of performance, the proposed classifier's execution time is linear and does not require high processing capabilities.*

KEYWORDS

Part-of-speech (POS) tagging, Arabic rule-based classifiers, Natural languages, Context-free grammars.

1. INTRODUCTION

In Arabic language, words are classified into three main classes: nouns, verbs, and particles or hārf (حرف), with each having their own grammatical functions and structures. Nouns cover names, places, things, and abstract concepts, whereas verbs convey action or state and vary according to tense, person, and gender. Particles, on the other hand, serve as connectors or modifiers, altering the meaning and relationship between words without possessing complete lexical meaning themselves.

Verbs in the Arabic language follow a unique patterning system that allows for the distinction between past, present, and imperative verbs. This patterning is governed by diacritical marks. Under these patterns fall all verbs that are similar in terms of the purpose they denote or the time in which the action occurs. Consequently, the set of patterns that determine the imperative form do not resemble the set of patterns that indicate past tense or the set of patterns for the present tense. These patterns are referred to as verb measures (الاوزان) or *taf῾īl* (تفعيل). The suffixes and prefixes attached to the verb serve other purposes, such as plural forms or the gender of the doer, whether male or female.

In this research, we introduce a part of speech (POS) classification algorithm capable of classifying words from Arabic corpora into verbs, nouns, and particles without the necessity of stemming. This approach eliminates the substantial costs associated with processing all possible prefixes, suffixes, and search time in dictionaries. The proposed classification process aims to facilitate word classification in the Arabic language anonymously, context-free, without requiring artificial-intelligence knowledge bases or dictionaries. In terms of processing capabilities, the method does not demand powerful computers or extensive memory resources.

The proposed algorithm is represented as a set of rules that work like a sieve panel, which is used to classify seeds of different shapes, sizes, and weights through a screen. Using the proposed ordered set of rules, each rule will be matched with potential matches in the provided text. When the shortest match is found, the tag associated with the rule will be assigned to the matched word. The rules are presented as regular expressions covering proper nouns, numbers, special characters, punctuation, pronouns and verbs.

Since regular expressions can be complex to understand, each rule or verb pattern, which we refere to

1. K. Gaashan is with Software Engineering Department, Philadelphia University, Amman, Jordan. Email: khuloodgaashan@gmail.com
2. M. Bani Younes is with Faculty of Information Technology, American University of Madaba, Jordan. Email: m.baniyounes@aum.edu.jo

as a measure, can be represented as a non-deterministic finite automaton (NFA) or multiple deterministic finite automata (DFA). However, after verification, the rules are implemented using regular expressions. Each measure represented by a regular expression (RE) consists of three major parts. Firstly, the rule body, where each letter in the word has specific diacritics sequence for a measure that uniquely identifies a verb. Secondly, the possible prefix, which is a set of possible characters that may precede the verb. Thirdly, the possible or optional set of suffixes or pronouns. In some complex cases, the diacritics change according to grammatical rules, and the vowels might be changed from (Yaa ي) to (Alef ا), … and so on. These cases can be expressed in a rule that matches many verbs without requiring a stemming process using the RE matcher.

Accuracy can be significantly improved if the corpus includes diacritics (tashkeel تشكيل). However, diacritics are not mandatory and are treated as optional components in the rules. Consequently, the proposed rules represent a best-effort algorithm that is not deterministic, meaning that tagging might change according to the level of detail provided in the text. Another aspect is the colloquial words and spoken language; such words are not considered in the proposed rules.

Classification and tagging present challenges due to ambiguity; in the Arabic language, a noun can also function as a verb. For instance, the word (yzeed, يزيد) , which means "increase", serves as a verb. Or it can be used as a proper noun, as noted by Farghaly et al. in [1] and Maamouri and Bies in [2]. Ambiguity increases when diacritics (tashkeel تشكيل) are absent and/or words are removed from their context.

However, the context or sentence in which a word exists makes it easy to identify the word's tag without going into any stages of classification, such as stemming or searching in a lexicon. For example, in the Arabic language, if a word is preceded by the prepositions (jarr particles احرف جر) such as (إلى , من, على, عن…), the subsequent word is a noun by default. Another rule is that if a word is accurately identified as a verb, such as (يسقي), which means to water something, then it cannot be preceded or followed by another verb. Therefore, the words before and after that verb are nouns with 100% accuracy. An exception is to be able to match those words with the patterns of prohibition, negation and affirmation words in the Arabic language or other categories that precede verbs.

The primary challenge is context-free part-of-speech (POS) tagging, which focuses on identifying a word's tag without taking into consideration the word's context. As noted by Eid et al. in [3], any verb should follow specific rules. Therefore, an efficient approach will be matching words with their corresponding measure first, if they can be identified. Otherwise, the word is most likely a noun, if not identified as a particle or any other known category. This approach is the one adopted in this research.

2. LITERATURE REVIEW

Words in the Arabic language are classified into three main categories: nouns, verbs, and particles, correspondingly, (fe'l, ism, and harf) [4]. As stated by B. Weiss in [5], grammarians put two methods to classify words into these categories: the descriptive method and the rational method. The descriptive method focuses on the observable features of each part of speech, such as nunnation, the genitive case, and the vocative case. The properties of a verb are the suffixes such as the letter tā(ت) equivalent to (T) and the letter Yā (ي), and the energetic nūn (ن). The rational method (àqlī) is non-investigative and non-empirical. On one hand, nouns are not tied to time; they possess meaning by themselves. On the other hand, the meanings of verbs are qualified based on a timeline (past, present, or future). This leads us to the classification of particles, hārf (حرف), which convey meaning in a context beyond their own. As stated by Weiss, the principle of classifying speech into these three parts developed out of "ilm al-wad" (علم الوضع) written by Iji, Adud al-Din Abd al-Rahman ibn Ahmad (d. 757/1355) in the fourteenth century, in a work entitled al-Risāla al-wad'iya. The research presented by B. Weiss [5] discusses in detail the states of nouns, particles, and verb, explaining how to differentiate between them based on context, meaning, and by the suffix. The verbs, as stated, can be identified by their radicals.

Alosaimy and Atwell [6] provided a comprehensive list of available part-of-speech (POS) taggers for both Classical Arabic (CA) and Modern Standard Arabic (MSA). In summary, they explained that the tagging process in the surveyed approaches depends on a morphological analyzer (MA) equipped with a lexicon that contains all possible solutions, regardless of the context being studied. They pointed out that no tagger has yet been adopted as a standard. The work presented offers a comparative study for

taggers and discusses their accuracy. Table 1 shows the accuracy for each tagger as presented in the paper of Alosaimy and Atwell [6].

Table 1. POS tagging accuracy for 50 classical words.

Accuracy	**MD**	**MA**	**ST**	**MR**	**WP**	**AM**	**MT**	**FA**
Overall	69.6%	70.6%	78.4%	66.7%	68.6%	79.4%	67.6%	74.5%
No Prop. Nouns	8.0%	78.5%	71.4%	52.8%	58.5%	74.2%	87.1%	74.2%
Prop. Nouns	46.8%	53.1%	93.7%	96.8%	90.6%	90.6%	25.0%	75.0%
MADA+TOKAN suite (MD), MADAMIRA suite (MA), Stanford POS tagger and segmenter (ST), MarMoT (MR), Segmentor and Part-of-speech tagger for Arabic (WP), AMIRA Toolkit (AM), Arabic Toolkit Service POS Tagger (MT), Farasa (FA) [6].								

Lee Y. et al. [7] presented a model for segmenting Arabic words based on the following pattern: prefix*-stem-suffix*, where the * represents the degree of the morpheme, indicating zero or more occurrences. The initial dataset was created manually, with each word segmented by a human, then the corpus is fed to the unsupervised model. The classification result is determined based on the closest probable sequence of morphemes identified. To increase the accuracy, a set of stems derived from 155 million words was imported to the model through an unsupervised algorithm. The accuracy claimed after importing a very large dataset of words and stems to the system is 97%. However, the accuracy is debatable, since stemming is automatic, and the stemming is done for a specific set of words. The described process requires tokenization based on whitespace and punctuation.

To achieve multiple goals within a single process, Habash, N. and Rambow, O. [8] proposed a model that integrates tokenization, part-of-speech tagging, and morphological disambiguation into a three-stage framework. In this proposed solution, tokenization, along with morphological and part-of-speech tagging are considered one process of three stages. In the first stage, all possible analyses are gathered for the sentence subject to study. In the second stage, a classifier consisting of ten morphological features is applied to the words in the text. The features include “POS” for parts of speech, “Conj” for clitic conjunctions, “Part” for particles, “Pron” for pronominal clitics, “Det” for clitic definite determiners such as (ال), “Gen” for gender, “Num” for numbers, “Per” for persons, and “Voice” and “Asp” for aspect (imperfective, perfective, imperative). As explained in the proposed solution, it is possible for one word to match more than one feature. A morphological analyzer will then choose among the returned results by considering two values: agreement and weighted agreement. Agreement represents the number of classifiers matching the analysis. The weighted agreement is the sum of all classifiers agreeing with the analysis. It is noticeable that the provided model is time-consuming and requires intense analysis and pre-knowledge of words and classifications, in addition to a database of prefixes, suffixes, and stems.

In two separate studies, E. Mohammad et al. discussed the feasibility of performing parts of speech tagging without word segmentation [9]. In the second study [10], they provided two methods for parts of speech tagging, adding a new method that depends on artificial intelligence and machine learning to segment words. In the first study, they compared methods of classification accuracy that do not depend on segmentation with other methods that rely on tokenization and segmentation. In their research, they claim an accuracy of 94.74% in tagging words without segmentation, compared to 93.47% accuracy when segmentation is used. However, the proposed solutions depend on having a dataset and trained algorithms ahead; prior knowledge is required.

In [11], Khoja presented PAT, a Part-of-Speech tagger, which utilizes a tagset of (131) tags that are used to manually tag a corpus to produce a lexicon. This lexicon is based on traditional Arabic language grammars. However, the tagset categories extended to include 35 additional tags to account for Arabic clitics. Furthermore, verbs are sub-divided into various sub-categories. Nouns are also classified into categories, distinguishing between singular and plural forms (the latter referring to two or more, according to the numbering system), along with other sub-classifications, such as particles, dates, numbers, and punctuation. The proposed tagger (PAT) performs initial tagging process, which is basically searching the produced lexicon for a match. This means that a pre-processing of stemming, removing prefixes and suffixes is required. Khoja claimed an accuracy rate of 97% using a dictionary containing 4,748 root words.

Mohammad Y. et al. in [12] proposed a tagger based on sentence structure using morphological analysis in conjunction with a Hidden Markov Model. However, the basics are still the same as found in the works of other researchers, including the need to stem, removing suffixes, prefixes, and the need to have a dataset. The difference is that Arabic grammar specifies by its rules and what can follow a verb in a sentence. For instance, if a verb is identified, then it cannot be followed by another verb; instead, it must be followed by a noun. This functions similarly to a context-aware system. In [13], Elhadj Y. attempted to implement the same approach in traditional Arabic text. However, the proposed work does not introduce any new concepts beyond what is presented in [12].

In [14], M. Hjouj et al. presented a tool for Arabic text tagging and named entity recognition depending on a two-phase process. Firstly, the proposed process passes the text through a lexicon recognition phase, followed by a morphological phase. However, the rule development for morphological classification is not explained in the paper. Additionally, some rules proposed in the work to classify words as nouns are flawed. For example, the rule states that any word that ends with the letters (alef- tā ات) is a noun. This is incorrect; for instance, the word (يقتات), which means (feed on), is a present-tense verb, not a noun, despite ending with the (alef- tā) letters.

Transformation-based Learning (TBL) for Part-of-Speech (POS) tagging is a corpus-based method introduced by Algahtani et al. in [15]. This approach claims to reduce the required processing power and offers more flexibility in guessing and classifying tags for unknown words compared to traditional rule-based methods. The proposed method depends on selecting the best-fit tag from a list of candidate tags generated by a morphological analyzer, which derives these tags from previously analyzed text. As described, the TBL POS approach requires a training set, pre-defined rules, and a lexicon containing tag/word combinations. Each word goes through assigning a tagging process where the most frequently matched tag is assigned, followed by a list of rules for further correction to the assigned tag. This approach highly depends on the existence of the word in the prepared corpus and context.

Zeroual et al. [16] aimed to present a hierarchical level of tagset and their relationships to produce more accurate results by navigating deeper in the relations. The proposed method is based on estimating transition probabilities using a decision tree. A tagger named TreeTagger, which is not specifically designed for the Arabic language, utilized the generated tagset used to tag Arabic text.

The use of regular expressions in Arabic-text research is not new; M. Tarawneh and E. AlShawakfa [17] explored the power of regular expressions to improve the accuracy of information retrieval in Quranic text. The use of regular expressions improved the matching process, where prefixes and suffixes can be presented as a group of possibilities surrounding the keyword subject of the search process. In our research, we enhanced regular expressions to represent verb measures.

3. METHODOLOGY AND ALGORITHM

The process for the proposed method is illustrated in Figure 1. Classification and tagging involve several steps, where the possible matches are reduced at each stage to reduce the number of future comparisons and mitigate ambiguities that may arise from partial matching.

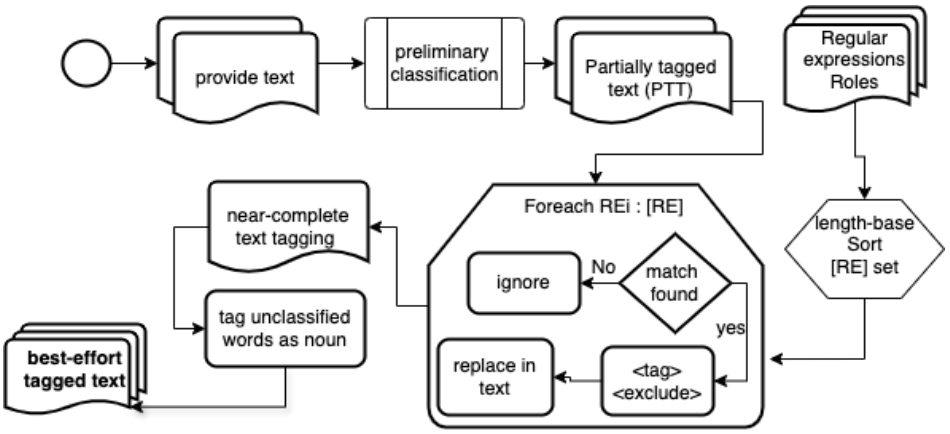

Figure 1. System flow, classification and tagging steps.

The process of tagging and classifying text involves three stages. The first stage is preliminary classification, during which the provided text goes through a basic classification based on detecting well-known words, such as present verbs preceded by particles of nasb (subjunctive markers). In Arabic grammar, the particles of nasb (subjunctive markers) always come after the present-tense verb. Another category includes nouns that are preceded by prepositions, which occur exclusively before nouns. The first stage is a bulk matching process, where a pair of processes <find, replace> is performed at once. This is the only part of the tagger that investigates partial context, one token before or after, not both, that makes the tagger a semi context-free tagger.

The latter process can be done for all known patterns of a form < particle, word> where the particle precedes a known morphological rule in identifying the successor "word". In Stage 2, a bulk match is performed for each regular expression representing a morphological verb form in the Arabic language, following the same method described in Stage 1. Consequently, the number of possible matches decreases with each step. In Stage 3, the text will contain tagged and untagged words. Any word in the text that remains untagged can be safely marked as a noun, as it does not match any regular expression within the set of regular expressions that defines any rule measure for verb , particle, or pronoun.

3.1 Regular Expressions' Classification Panel (ReCp) Design

In the proposed algorithm, clitics, such as continuous pronouns, are not removed from the words in order to be tagged.For instance, the word (فهمتها) should be segmented, on conventional tagging methods, into (ت + فهم + ها), the verb (fahim فهم/ understood), then the (tā / ت, pronoun referring to the speaker) and the (Haa/ها, pronoun referring to the object). Conjunctions and prepositions that are not part of the word, such as (عن , الى ، من ...), can also be attached to a pronoun. For example, (من + ها) will form the (منها) word, which means "from it". Such pronouns will not go through any stemming process.

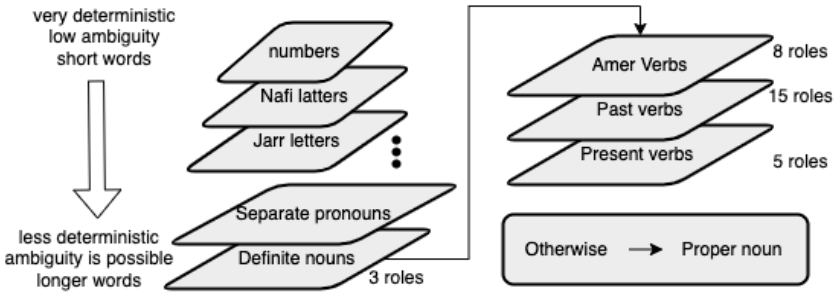

Figure 2. The priorities of regular expressions presented in the sifting screen (matcher).

The word-sieve panel is represented as a set of regular expression rules specifically designed for recognizing patterns in the Arabic language. Each rule is a regular expression (RE) representing a verb measure. In terms of length, the expressions are organized from shortest to longest. For instance, expressions representing particles, such as prepositions, accusatives, conjunctions, and separate pronouns, come first. Next are three-letter verbs, followed by verbs with four-letter roots, … and so on. As the text enters the screen, it will pass over the small expressions first; if it fits a pattern, then it will be classified as a word of that measure or rule. Otherwise, the algorithm will move it further along the screen. Figure 2 shows the screening mechanism, its design, and the number of rules for each stage.

3.2 Regular Expression Design

Representing Arabic verb weights using regular expressions is challenging due to the large number of possibilities and variations that depend on the verb's context within a sentence. In Figure 3, we present a regular expression for the measure tafāal / تفعّل (example سيتقدّمهم). The question mark in the regular expression represents an optional term with at most one occurrence in the word. As illustrated, it is possible to represent an entire family of verbs with this expression without needing to know the verb itself. This identification gives us the flexibility to keep the action represented by the verb anonymous.

As shown in Figure 3, the prefix (سيت) can be expanded as needed to cover additional cases; however, we shortened it to simplify the presentation of the verb weight explained. The RE then continues to match the middle letters of the verb and any possible diacritics. This section is indicated in the figure by the dotted, underlined part. Then the remaining part of the verb weight, which is optional, presents the possible suffixes. In RE, the optional components are followed by (?).

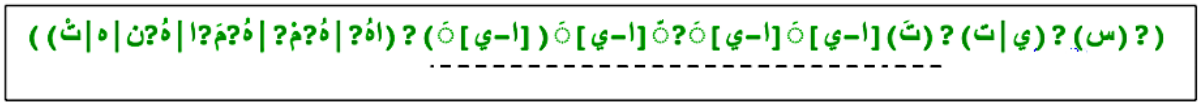

Figure 3. Regular expression for the measure (Tafaal / تفعّل).

Figure 3 shows the DFA or machine for the RE presented in Figure 3 that represents a past-verb measure. Group 2 and group 3 in the figure show the possible optional prefixes such as (سي/ست /…) at the beginning, denoting timing in the future, intention, because of the letter (s/ س) and the (tā / ت) for the opposite person.

However, the possible optional suffixes at the end of the verb, group 6, denote gender, such as (ها ، هن) for female, or the letter (hā/ هاء) for singular male. The figure shows the basic rule or measure where the diacritics (تشكيل) are optional in some cases, such as the second Fat'hā (ً) with the loop over it. But, when diacritics are provided, tagging accuracy will be higher. Groups 2,3 in the figure are optional prefixes, where group 6 is an optional suffix with non-deterministic probability.

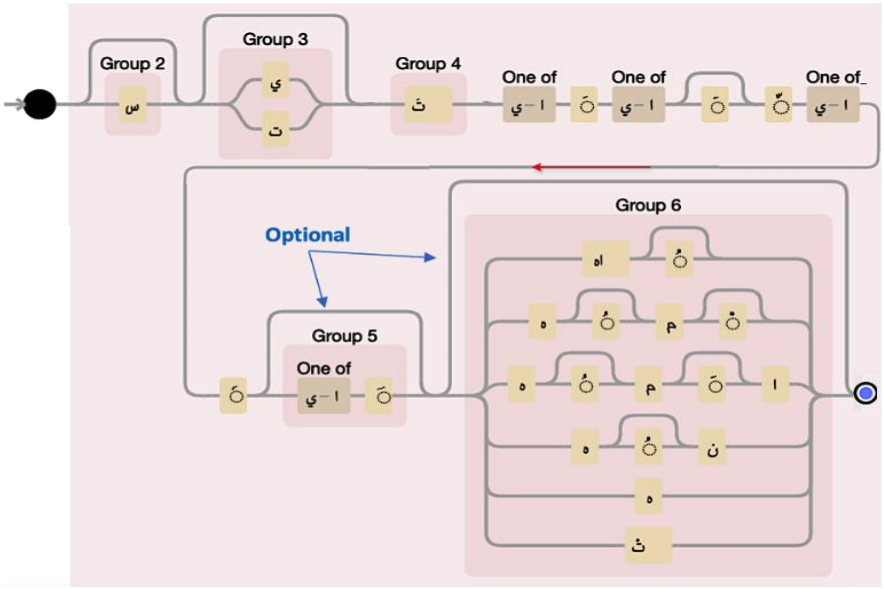

Figure 3. Finite automaton representing the measure (Tafaal / تفعّل).

Each verb can be represented by a finite automaton. However, we can notice similarities between some verb measures in the Arabic language within the same category of verbs, such as the imperative-verb weight. This similarity enables the merging of multiple finite automata to recognize various verb forms or weights.

Table 2 shows an example of imperative verbs where multiple weights are represented by a single regular expression for one verb tense. In this table, imperative verb rule 2 combines two forms into one regular expression, as the objective is just to identify the tense of the verb, not the root or verb meaning. This approach can be applied to other verb tenses, such as present (*mudari*-مضارع) and past (*madi*-ماضي). Regarding nouns, some nouns can be identified when preceded by the definite article (alef-laam ال) or by specific disconnected pronouns or preposition particles.

As shown in Figure 4 and its corresponding regular expression, the weights of the two imperative verbs are represented by a single Deterministic Finite Automaton (DFA) and one regular expression, indicated by rule 2. However, rule 1 consolidates five rules due to the high similarity in the structures of the verb weights.

Table 2. Similar verb weights that can be represented in one regular expression.

Imperative verb weights 1	1. "افْعُلْ" 2. "افْعِلْ" 3. "افْعَلْ" 4. "أَفْعِلْ"
RE presentation	(ا|إ|أ)[ُ|ِ|َ]?(ي-ب)[ُ|ِ|َ]?(ي-ب)?(ْ)?
Imperative verb weights 2	1. "تَفَعَّلْ" : example (تَقَدَّمْ Move forward) 2. "تَفَاعَلْ": example (تَسَامَحْ Tolerate)
RE presentation	(ْ)[ي-ب](([ي-ب][ً]{2})|(ا[ي-ب]ً))[ي-ب]ً(ت)

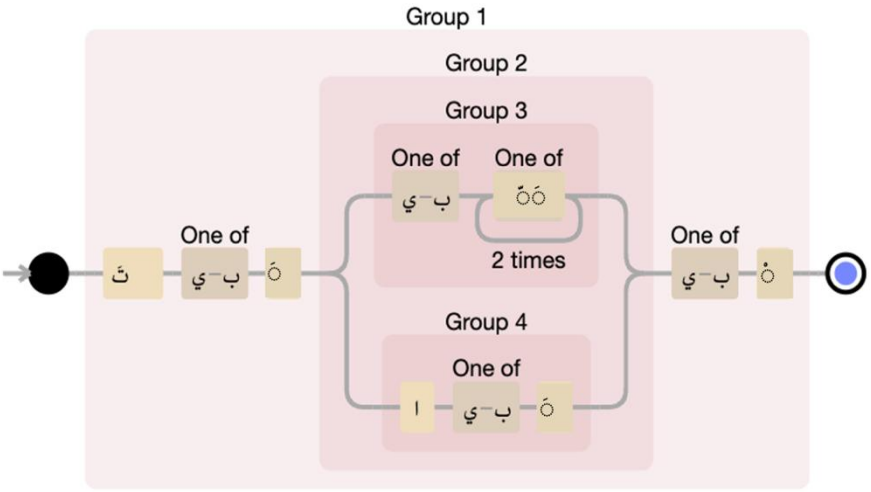

Figure 4. DFA for imperative verb, rule 2 described in Table 2.

3.3 Algorithm

Table 3 shows the proposed algorithm for the classifier. The algorithm starts with considering any Arabic text provided as (T). Two sets of regular expressions defined, CP and R, where CP is used in the initial matching to reduce future work done by R bulk matcher. Time measured before and after tagging is performed as (**τ**).

Table 3. Algorithm for Arabic regex morphological tagging.

Let: 1. **T** ∈ Σ* : Input Arabic text over some alphabet Σ 2. **CP** = { (**C**$_j$, **P**$_j$) }$^m_{j=1}$: a set of category-pattern pairs where **C**$_j$ is a linguistic category name and **P**$_j$ is a regular expression pattern for preliminary matches (obvious cases) 3. **R** = { (**C**$_i$, **P**$_i$) }$^n_{i=1}$: A finite ordered set of category-pattern pairs where **C**$_i$ is a linguistic category name and **P**$_i$ is a regular expression pattern (for bulk match) 4. *counter* ∈ ℕ : Counter for replacements 5. **τ** : Execution time in nanoseconds
Input: • B ∈ Base64 (optional) — Base64-encoded Arabic string Output: Tagged text T', total replacements counter, execution time **τ**
Initialization: 1. Let R ← {(C₁, P₁), (C₂, P₂), ..., (C$_n$, P$_n$)} 2. Initialize counter ← 0 ***Input Handling:*** **3.** If B ≠ ∅: Decode B using Base64 to get T ∈ Σ* 4. Else: Read T from dataset file 5. ***Timing Start:*** Let t_start ← System.nanoTime() ***preliminary classification*** 6. foreach **R**$_j$ ∈ **CP**

```
      if (matcher.find(R_j , T) :
          T ← Replace(T, ' C_j  [T_i ] ')
    Pattern Matching and Replacement:
7.  For each (C_i, P_i) ∈ R:
8.      From the results→ While P_i matches a substring m ⊂ T & m not tagged:
9.          replace m in T with  ' C_i  [m] '
10.         counter ← counter + 1
11. Timing End: Let t_end ← System.nanoTime() , Compute τ ← t_end - t_start
12. Output:  Print transformed text T , counter , τ
```

The first two lines of the algorithm show the initialization step, where all regular expressions are added to the regular expression (RE) hash map. The order in which RE is added to the map is significant to the work of the algorithm, since it should follow the presentation described in sub-section 3.1. In line 6, the preliminary or initial classification starts using the CP set. Each ($\mathbf{R}_j$) is a regular expression that performs one bulk match for a very specific family of verb weights or noun detection in a single command. The number of preliminary rules specified in the panel is eight. Each match will be marked so that it will be excluded from any future processing. The algorithm in lines 7 through 10 is performing a bulk match for each verb, particle or noun weight found in the text. Each rule uses the matcher to perform a bulk match in the hash map for all possible string matches that have not been previously processed. At the end of the algorithm, time is measured again to get Δt, as $\boldsymbol{\tau}$, presenting the consumed time for all tagging performed.

3.4 Complexity

Let (n) be the length of the text (in characters), (m): the number of patterns in the regex patterns, and (k) be the number of matches a single regular expression finds in the text. The initialization process is just the area where regular expressions are added to the matching map, a one-time process. Lines 3 and 4 are getting the input from the user and storing it in the text to analyze. In line 6, the process of bulk match and replace will take O(1) for each regular expression, since it does not loop over the text, or the regular expressions set. For the remaining lines, starting at line 7, in the worst case, a regular expression matching is: O(n) per pattern (can be worse depending on pattern complexity, but typically O(n)), across m patterns, this is O($m \times n$) as it might look in the worst case.

However, a deeper look at the proposed algorithm, the outer loop that passes over all defined regular expressions, is always bounded to 45 times, which is the number of regular expressions defined. The inner loop will execute only once when a match is found for the rule. The matcher is just walking through the input linearly with no backtracking. This means that the matching process does not scan the entire text again as if it were a new scan every time. Instead, it resumes from where the last match ended. So, across the inner loop, the matcher will look at each character at most once per regular expression.

Based on the previous discussion, assume n = length of the text in characters. k = number of regular expression rules (45 constant). Thus, the total work will be O($k \cdot n$)= O($45 \cdot n$)=O(n). In terms of space complexity, the algorithm does not reserve new memory to process the input. Consequently, the space will remain at the boundaries of O(n).

4. RESULTS AND TEST CASES

Hardware specifications of the classifier used in are MacBook Pro 2012, macOS Catalina version 10.15, 16 GB 1600 MHz, 2.3 GHz Quad-Core Intel Core i7. The classifier is programmed using the Java programming language. However, other anonymous servers are used as well to run the classifier, giving a much better time. The dataset used is provided by T. Zerrouki and A. Balla [18] which contains 98.85% classical Arabic text and 1.15% Modern Standard Arabic text . The dataset contains 7701 manually diacritized words. Our classifier is available on the website mentioned in [19] as an experimental release.

As shown in Figure 5 for the relation between the input size and the number of comparisons the algorithm performs, it is noticeable that the increase is almost linear for several input sizes. As shown in Figure 5 (A), for small file sizes, the number of matches or passes is less than the number of tokens, since regular expression matches rules for each measure one time only. When the file size increased to contain more than 100,000 tokens, comparisons are around 39,400, as shown in Figure 5 (B). The

algorithm showed a consistent behavior for incremental input sizes, as shown in the previous figures and as presented in Table 4.

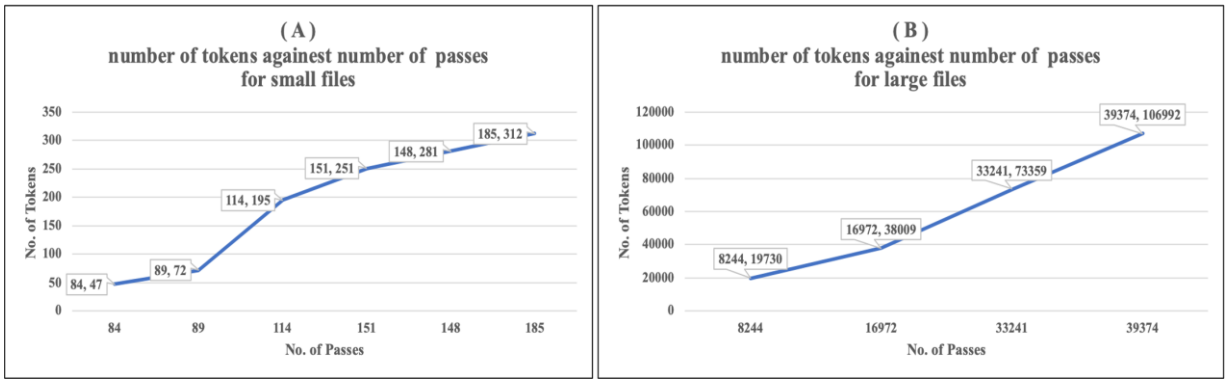

Figure 5. The linear increase of number of comparisons *versus* number of tokens.

Table 4. Test cases and performance metrics.

Tagged Tokens	Size in KB	Time in ms
47	1.34	49
72	1.97	55
195	4.00	60
251	5.65	63
281	6.47	69
312	8.05	79
19730	434.16	911
38009	941.84	1373
73359	1770.54	2245
106992	2183.34	3099

Figure 6 shows the growth in time against the growth in input size for our proposed algorithm. As illustrated in the figure, time complexity increases linearly with input size and is not related to code design, as pointed out earlier in the complexity analysis. For files approximately 1.0 MB in size, the time required to classify and tag words is about 3 seconds.

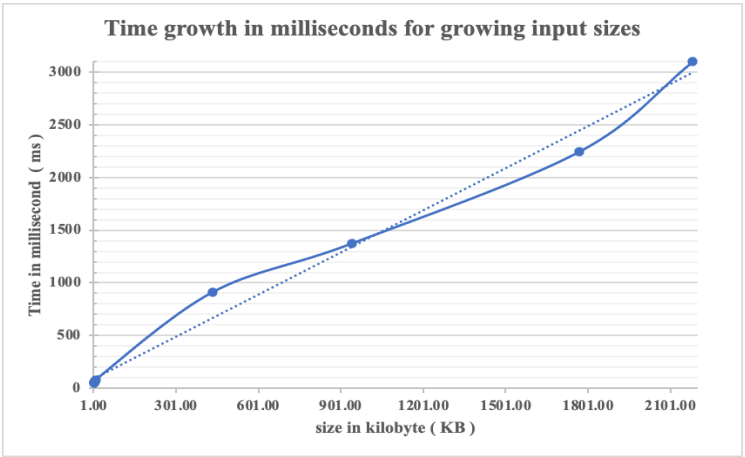

Figure 6. Time growth in relation to input size.

As shown in Table 4, the input size is 47 words, out of which 3 preliminary matches happened before the classifier starts its work. As shown, the complexity of execution relates to the size of the input only. Time taken to perform the tagging is about 42ms, whereas in faster machines it takes about 10ms only, as seen using our classifier provided online at [19].

Table 5. Sample output and analysis from the classifier.

Sample Input			
الْحَمْدُ لِلَّهِ الْغَنِيِّ الْحَمِيدِ الْمُبْدِئِ الْمُعِيدِ، الْفَعَّالِ لِمَا يُرِيْدُ، كَتَبَ الْعِزَّةَ والْكَرَامَةَ لِمَنْ أَطَاعَهُ، وقَضَى بِالذِّلَّةِ والْهَوَانِ عَلَى مَنْ عَصَاهُ وهُوَ الْعَزِيزُ الْحَكِيمُ. وأَشْهَدُ أَنْ لَا إِلَهَ إِلَّا اللَّهُ، أَنْعَمَ عَلَيْنَا بِالْكِتَابِ الْمُبِينِ والرَّسُولِ الصَّادِقِ الْأَمِينِ			
Output			
Noun :الْحَمْدُ LafzAlJalal :لِلَّهِ Noun :الْغَنِيِّ Noun :الْحَمِيدِ Noun :الْمُبْدِئِ Noun :الْمُعِيدِ Noun :الْفَعَّالِ JarrMjror :لِمَا Present :يُرِيْدُ Past :كَتَبَ Noun :الْعِزَّةَ	AtfParticle :وَ Noun :الْكَرَامَةَ AsmaaMusol :لِمَنْ Past :أَطَاعَهُ AtfParticle :وَ Past :قَضَى Noun :بِالذِّلَّةِ Atf Particle :وَ Noun :الْهَوَانِ JarrParticle :عَلَى JazmParticle :مَنْ Past :عَصَاهُ	AtfParticle :وَ DameerMonfasel :هُوَ Noun :الْعَزِيزُ Noun :الْحَكِيمُ AtfParticle :وَ Present :أَشْهَدُ NasbParticle :أَنْ NafiOrNahi :لَا LafzAlJalal :إِلَهَ Estithnaa :إِلَّا LafzAlJalal :اللَّهُ	Past :أَنْعَمَ JarrMjror :عَلَيْنَا Noun :بِالْكِتَابِ Noun :الْمُبِينِ AtfParticle :وَ Noun :الرَّسُولِ Noun :الصَّادِقِ Noun :الْأَمِينِ
Total Tokens: 47 Possible preliminary matches : 3 Matches through loops : 39 Tokens in the text larger than 1 symbol = **42**		Number Iterations: 39 size in bytes: 1368.0 Time to classify: 42.70574 *ms*	

Table 5 shows an example of tagging Arabic text decorated with diacritics. Table 6 shows the pre-classification process results where words preceded by preposition particles are classified by default as nouns. The tag (noun2) is assigned to them to distinguish them from the words tagged based on word structure. The example of present verb preceded by Nasb letter, if the verb is tagged in the pre-classification stage, the verb will be tagged as PresentNasb, not just present.

Table 6. Pre-classification sample results for nouns and present verbs.

Sentence to analyze:	مِنَ الناسِ الطيب و في قلبهِ الرحمة و لَنْ يخذلَ السائل
JarrLetter ← مِنَ Noun2 ←الناسِ Noun1 ←الطيب AtefCONJ ← و JarrLetter ←في Noun2 ←قلبهِ	Noun1← الرحمة AtefCONJ ←و NasbLetter ←لَنْ PresentNasb ← يخذلَ Noun1 ←السائل

Table 7. Fine-grained tagging based on morphological features.

Sentence to analyze:	ضرار بن الأزور، سَافَرَ مَعَ خولة بنت الأزور ! وُلِدَتْ في القرنِ 7 .
NounMale ←ضرار Noun←بن Noun ←الأزور **punctuation** ← ، Past ← سَافَرَ JarrLetterOrDarf ←مَعَ **NounFemail**←خولة Noun ← بنت Noun4 ←الأزور	**punctuation** ← ! Past ← وُلِدَتْ JarrLetter ←في Noun ← القرنِ **Numbers** ← 7 **punctuation** ← .

Regular expression allows fine-grained tagging based on morphological features, such as numbers, punctuation, and gender. As shown in Table 7, some pronouns and names are gender-specific. For example, (بن, bin), which means "son of", and (بنت, bint), which means "daughter of". Such words provide the ability to tag words correctly before they are used correctly based on gender. However, depending on such terms may not be useful to tag words after them with the same accuracy.

Jordanian Journal of Computers and Information Technology (JJCIT), Vol. 11, No. 04, December 2025.

5. COMPARISONS AND FEATURES

5.1 Comparison with Previous Approaches

Compared to previous work, CAMel tools [20], which are powerful, well-known, and multi-purpose analysis tools; CAMel requires approximately 5.2 GB of information to be stored and processed before the analyzer can be used. However, our proposed method provides POS tagging with predictable space compared to CAMel. The required storage for our proposed method is less than 0.5 MB of rules. Since CAMel provides many services, the comparison will be just for the POS tagging feature. As shown in Figure 7, using the same hardware used for our algorithm, CAMel behaves differently for variable input sizes. In both small and large inputs, our algorithm takes much less time than CAMel. For 2MB of input size, our algorithm took about 3 seconds, whereas in CAMel, using trained AI model, it needed 140.25 seconds to finish tagging. This shows that the rule-based algorithm is about 97.86% faster than CAMel.

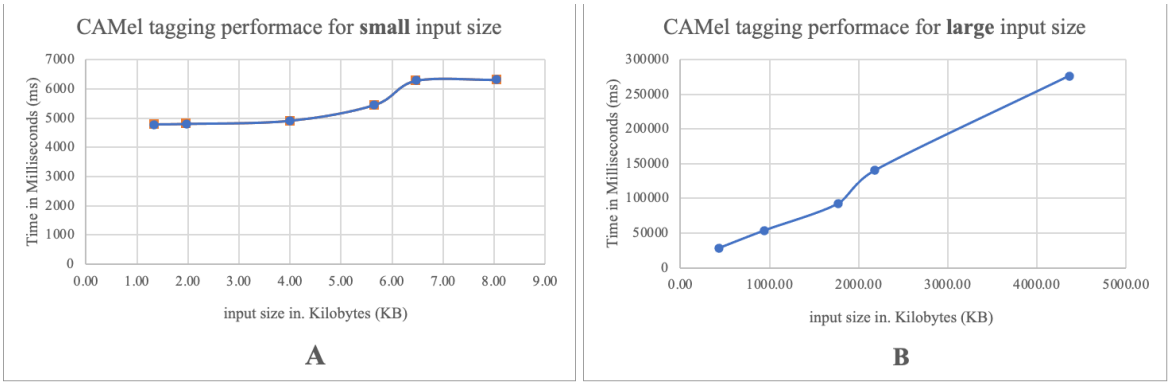

Figure 7. CAMel performance for small (A) and large (B) datasets.

Using the same input data shown previously in Table 5, CAMel took 6.6 seconds to process the sentence, whereas our proposed light-weight algorithm took 42.70574 *ms* for the same input. In terms of accuracy, CAMel incorrectly tagged some words, as shown in Table 8 marked with *. The error ratio for the provided text is about 11% for 47 words.

Table 8. Results for CAMel POS tagging for fixed benchmark input.

الْفَعَّالِ ← adj	الْحَمِيدِ ← noun_prop	أَشْهَدُ ← verb
لِمَا ← verb. *	[لا يوجد تحليل] ← الْمُبْدِئِ *	أَنْ ← conj_sub
يُرِيْدُ ← verb	الْمُعِيدِ ← noun	لَا ← part_neg
لِمَنْ ← verb *	كَتَبَ ← noun *	إِلَهَ ← verb *
أَطَاعَهُ ← verb	الْعِزَّةَ ← noun	إِلَّا ← verb

Figure 8 shows the performance for Farasa tagger using small datasets of a size of 4 kilobytes up to 180 kilobytes as shown in part (A) of the figure. Part (B) shows the performance using large datasets starting at 1500 kilobytes up to 5000 kilobytes. As illustrated in the figure, Farasa demonstrates a linear growth in execution time as the size of the input files increases.

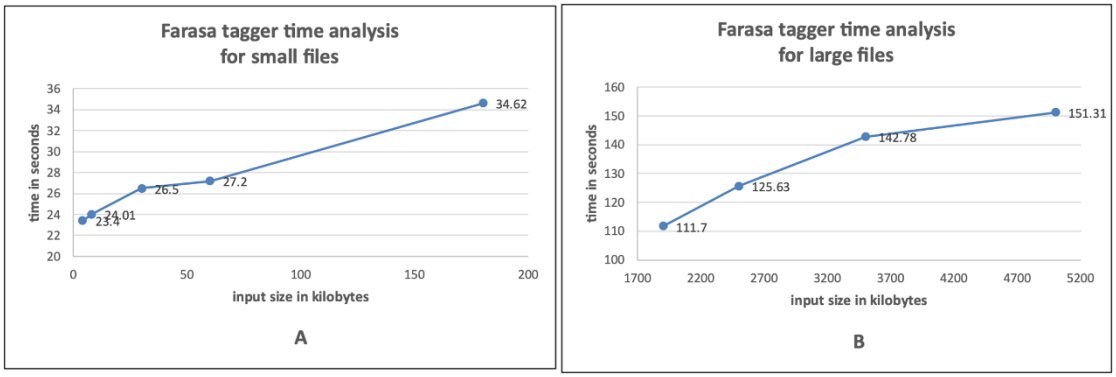

Figure 8. Farasa performance analysis using small and large datasets.

Table 9. Comparison with Farasa POS tagger, Golden Standard POS dataset tags and the proposed light-weight tagger.

Word	Farasa POS		Proposed Light-wight algorithm		Gold Standards POS tags
	Tag	result	Tag	accuracy	
الْحَمْدُ	DET+NOUN	OK	Noun	OK	DET+NOUN
لِلَّهِ	PREP NOUN	OK	LafzAlJalal_Noun	OK	PREP+NOUN
الْغَنِيِّ	DET+NOUN	OK	Noun	OK (Ignored)	DET+ADJ
الْحَمِيدِ	DET+NOUN	OK	Noun	OK (Ignored)	DET+ADJ
الْمُبْدِئِ	DET+ADJ	OK	Noun	OK	DET+ADJ
الْمُعِيدِ	DET+NOUN	OK	Noun	OK (Ignored)	DET+ADJ
،	PUNC	OK	punctuation	OK	PUNC
الْفَعَّالِ	DET+ADJ	OK	Noun	OK	DET+ADJ
لِمَا	PREP PART	OK	JarrMjror	OK	PREP+PRON
يُرِيْدُ	V	OK	PresentVerb	OK	V
،	PUNC	OK	punctuation	OK	PUNC
كَتَبَ	V	OK	PastVerb	OK	V
الْعِزَّةَ	DET+NOUN+NSUFF	OK	Noun	OK	DET+NOUN+NSUFF
والْكَرَامَةَ	CONJ DET +NOUN+NSUFF	OK	AtefCONJ	OK	CONJ
			Noun	OK	DET+NOUN+NSUFF
لِمَنْ	PREP PART	OK	AsmaaMusol	OK	PREP+PRON
أَطَاعَهُ	V PRON	OK	PastVerb	OK	V+PRON
،	PUNC	OK	punctuation	OK	PUNC
وقَضَى	CONJ V	OK	AtefCONJ	OK	CONJ
			PastVerb	OK	V
بِالذِّلَّةِ	PREP DET+ NOUN+NSUFF	OK	Noun	OK	PREP+DET+NOUN+NSUFF
والْهَوَانِ	CONJ DET+NOUN	OK	AtefCONJ	OK	CONJ
			Noun	OK	DET+NOUN
عَلَى	PREP	OK	JarrLettersPREP	OK	PREP
مَنْ	PREP	NO	LetterJazm	NO	REL.PRON
عَصَاهُ	**NOUN PRON ***	NO	PastVerb	OK	VERB+PRON
وهُوَ	CONJ PRON	OK	AtefCONJ	OK	CONJ
			DameerMonfasel	OK	PRON
الْعَزِيزُ	DET+NOUN	OK	Noun	OK (Ignored)	DET+ADJ
الْحَكِيمُ	DET+ADJ	OK	Noun	OK	DET+ADJ
.	PUNC	OK	punctuation	OK	PUNC
وأَشْهَدُ	CONJ V	OK	AtefCONJ	OK	CONJ
			PresentVerb	OK	V
أَنْ	PART	OK	NasbLetters	OK	PART
لَا	PART	OK	NafiOrNahi	OK	NEG.PART
إِلَهَ	NOUN	OK	LafzAlJalal_Noun	OK	NOUN
إِلَّا	PART	OK	Estithnaa	OK	EXCEPT.PART
اللَّهُ	NOUN	OK	LafzAlJalal_Noun	OK	NOUN
،	PUNC	OK	punctuation	OK	PUNC
أَنْعَمَ	V	OK	PastVerb	OK	V
عَلَيْنَا	PREP PRON	OK	JarrMjror	OK	PREP+PRON
بِالْكِتَابِ	PREP DET+NOUN	OK	Noun	OK	PREP+DET+NOUN
الْمُبِينِ	DET+ADJ	OK	Noun	OK	DET+ADJ
والرَّسُولِ	CONJ DET+NOUN	OK	Noun	OK	CONJ+DET+NOUN
الصَّادِقِ	DET+ADJ	OK	Noun	OK	DET+ADJ
الْأَمِينِ	DET+NOUN	OK	Noun	OK (Ignored)	DET+ADJ

As shown in Table 9, the same text used with CAMel was classified and tagged using the Farasa POS tagger, a complete stack for Arabic language processing. It was produced by [21] and is available online at [22]. Farasa tagger accuracy was 95.56% when compared to the golden standard POS tags using the Penn Arabic Treebank [23]. In our proposed semi context-free tagging approach, we do not consider the

tags derived from nouns. For example, the word (الْغَنِيّ), which is a noun, when put in the sentence context, it is an adjective (DET+ADJ) according to the gold standard. The accuracy percentage of our proposed algorithm, based on the test cases, is slightly over 97%. As demonstrated in Table 9, the Farasa tagger incorrectly classified the word (عَصَاهُ) to be a noun meaning (his stick). Whereas, our rule-based classifier correctly identified it as a verb in the past tense, as it matches a specific verb rule. However, our tagger will not be as accurate as other taggers when diacritics are partially provided on the text.

We compared the accuracy of our algorithm with that of other taggers using a dataset provided by the Universal Dependencies Treebank of Arabic, developed at New York University Abu Dhabi (NYUAD). Following the extraction of results from each tagger, we normalized the assigned tags to establish a unified tagset, allowing systematic comparison across all outputs. Table 10 shows an accuracy comparison using precision, recall and F1 for CAMel, Farasa and Stanford POS taggers against our provided tagger. Farasa tagger presented a 94.37% accuracy, followed by our proposed algorithm with an accuracy of 93.07%. Table 11 summarizes the architecture and the key models used in each tagger.

Table 10. Accuracy comparison for standard Arabic text.

Taggers	**Precision**	**Recall**	**F1**	**Accuracy**
Farasa	89.31%	76.77%	81.53%	94.37%
Our poropsed tagger	94.86%	89.3%	90.07%	93.07%
CAMeL	65.85%	65.51%	62.48%	87.01%
Stanford	70.7%	70.53%	68.11%	81.82%

Table 11. Summary of under-laying AI architecture for used taggers.

Tagger	**Model Architecture**	**Key Model(s)**	**Note**
Farasa	Deep Learning	RNNs, LSTMs	Designed for speed and accuracy in
CAMeL	Machine Learning/Deep Learning	SVMs, RNNs	Uses a mix, often SVMs for reliability.
Stanford	Machine Learning (Classic) / Deep Learning (New)	MaxEnt (Classic) LSTMs/Transformers (New)	The classic version is not deep learning. The modern Stanza version is AI based tagger.

Table 12 shows the results of another comparison performed using 3358 words from Quranic text. CAMeL tagger scored 66.85% in accuracy, while our proposed algorithm scored just above 50%. The downside of our tagger is the fact that it does not cover some special Quranic diacritics and some word measures. For example, words like book (كِتَٰب) and devil (شَيۡطَٰن) in Quranic writing where the letter alef is superscripted unlike in standard Arabic (كتاب) and (شيطان), respectively.

Table 12. Accuracy test using Quranic text.

Taggers	**Precision**	**Recall**	**F1**	**Accuracy**
Our poropsed tagger	32.72%	14.3%	19.67%	50.01%
CAMeL	25.89%	15.23%	12.45%	66.85%

Table 13. Meaning manipulation using diacritics in Arabic language.

Sentence : مَنْ مَنَّ مِنْ مَنِّ مُنَّ مِنَ المنانِ Approximate translation: Who blesses from his blessings, will be bestowed by the generous one					
Word	Proposed Algorithm	CAMeL	Farasa	Stanford	Golden tags
مَنْ	Letterjazm/JUSPART	PREP*	PREP*	PREP*	PREP
مَنَّ	PASTVERB	PREP*	PART*	PRON*	VERB
مِنْ	JARRLETTER_PREP	PREP	PART*	PREP	PREP
مَنِّ	NOUN/PRON	PREP*	PREP*	NOUN/PRON	NOUN/PRON
مُنَّ	PASTVERB	PREP*	PART*	PREP*	VERB
مِنَ	JARRLETTER_PREP	PREP	PART*	PRON*	PREP
المنانِ	NOUN	X *	NOUN	NOUN	NOUN

An extreme case is the sentence presented in Table 13, where six words with five different meanings are provided by changing diacritics. As shown in the table, tagging based on rules presented better results than by other taggers. This behaviour is expected, since rules are deterministic more than taggers that depend on machine learning or deep learning. Bad tags are marked with *.

4.2 Features of the Proposed Algorithm

In summary, the proposed light-weight, rule-based tagging system offers several advantages over traditional tagging methods and modern AI approaches.

Hardware Requirements: Our proposed algorithm operates effectively on any hardware capable of running Java or Python code, without demanding high memory or CPU resources. While there may be a slight performance degradation on weaker hardware, our proposed algorithm is designed to be an in-place algorithm, requiring no extra datastructures or memory. Furthermore, the execution time of the algorithm can be anticipated to be lower than expected due to its linear growth in execution time, as demonstrated in the performance analysis provided earlier.

Accuracy: The accuracy of the proposed tagger depends on two factors: 1) the definition of rules and 2) the presence of diacritics on text. This means that the tagger can offer a best-effort tagging based on diacritics accuracy. Comparing to machine-learning and deep-learning approaches, AI-based algorithms require retraining with a substantial amount of pre-classified words to cover all possibilities and optimize the results. This leads us to the third feature of our rule-based classifier: low maintenance cost.

Low Maintenance Cost: Modifying or adding a rule will alter the results or add a new tagging category. However, ambiguity in certain words, such as nouns in the form of verb (i.e. يزيد yazeed), remains a challenge for all taggers. This issue can be solved by looking into the context in which the word resides. Such cases are not solved in the provided algorithm, as it discusses the context-free approach. It is worthy noticing that adding new measures to the tagger may require revising the sieve design and the priorities of applying those new measures when tagging text.

Finally, the execution time. As shown by the provided comparisons, the rule-based algorithm is efficient more than traditional and AI-based approaches, since: 1) It does not perform prefix and suffix processing. 2) It eliminates dictionary search time, and 3) No databases for training are required.

6. CONCLUSIONS

Tagging and morphological analysis using regular expressions show that the Arabic language is sensitive to context, diacritics, suffixes, and prefixes. As shown by this paper, it is possible to develop lightweight, fast, and effective classifiers and taggers using regular expressions. However, the way regular expressions are used, in terms of order in the matching panel and the reduction of ambiguity, is crucial.

When designing regular expressions, the following points should be considered to minimize the cost of the matching process. First, use possessive quantifiers whenever possible to reduce backtracking (i.e., use ++ instead of +). For example, when matching the regular expression “(a+)+b” with the string “aaaaaaax”, the matcher will fail after a very long time because of backtracking, since there is no letter “b” at the end. By modifying the RE to possessive RE “(a++)+b”, the matcher will fail faster, since there is no “b” at the end of the input string. This will improve RE performance in cases where no matches are found in the text.

Second, it is important to avoid nested quantifiers such as (.*)* or (a+)+ to constrain matching and avoid unexpected results. Additionally, use anchoring (^, $, \b) . Finally, try to avoid using look heads and look behinds when possible to reduce complexity. The tagger can be improved by building a better context-aware analyzer to solve the problem of ambiguity and wrong tagging, and by integrating machine-learning and deep-learning techniques with the current rule-based approach.

REFERENCES

[1] A. Farghaly and K. Shaalan, "Arabic Natural Language Processing: Challenges and Solutions," ACM Transactions on Asian Language Information Processing, vol. 8, no. 4, pp. 1–22, Dec. 2009.

[2] M. Maamouri and A. Bies, "Developing an Arabic Treebank: Methods, Guidelines, Procedures and Tools," Proc. of the Workshop on Computational Approaches to Arabic Script-based Languages, pp. 2–9, Geneva, Switzerland, Aug. 2004.

[3] M. Eid, Alnaho Almosaffa, النحو المصفى, vol. 1, ISBN: 9772324660, Deposit 1975/4427, Alshabab Library at Cario, 1971.

[4] B.-E. A. Ibn Aqeel, Sharh Ibn 'Aqeel' Ala Alfiyyah Ibn Malik, 1st Edn., vol. 1, Alresalah Center for Heritage Studies , Cairo, 1962.

[5] B. Weiss, "A Theory of the Parts of Speech in Arabic (Noun, Verb and Particle): A Study in 'ilm al-wad," Arabica, vol. 23, no. 1, pp. 23–36, Feb. 1976, Accessed: Feb. 04, 2024.

[6] A. Alosaimy and E. Atwell, "Tagging Classical Arabic Text Using Available Morphological Analysers and Part of Speech Taggers," JLCL, vol. 32, no. 1, pp. 1–26, 2017.

[7] Y.-S. Lee, K. Papineni, S. Roukos, O. Emam and H. Hassan, "Language Model-based Arabic Word Segmentation," Proc. of the 41st Annual Meeting on Association for Computational Linguistics (ACL '03), pp. 399–406, DOI: 10.3115/1075096.1075147, Morristown, NJ, USA, 2003.

[8] N. Habash and O. Rambow, "Arabic Tokenization, Part-of-speech Tagging and Morphological Disambiguation in One Fell Swoop," Proc. of the 43rd Annual Meeting on Association for Computational Linguistics (ACL '05), pp. 573–580, Ann Arbor, Ed., Morristown, NJ, USA, Jun. 2005.

[9] E. Mohamed et al., "Is Arabic Part of Speech Tagging Feasible Without Word Segmentation?" The Association for Computational Linguistics, pp. 704–708, doi: 10.13140/2.1.3631.8402, 2010.

[10] E. Mohamed and S. K. Ubler, "Arabic Part of Speech Tagging," Proc. of the 7th Int. Conf. on Language Resources and Evaluation (LREC'10), pp. 2537–2543, [Online], Available: http://www.lrecconf.org/proceedings/lrec2010/pdf/384_Paper.pdf, 2010, Accessed: Feb. 04, 2024.

[11] S. Khoja, "APT: Arabic Part-of-speech Tagger," Proc. of the Student Workshop at the Second Meeting of the North American Chapter of the Association for Computational Linguistics, Carnegie Mellon University, Pennsylvania, 2001.

[12] Y. O. Mohamed et al., "Arabic Part-of-speech Tagging Using the Sentence Structure," Proc. of the 2nd Int. Conf. on Arabic Language Resources and Tools, pp. 241–245, Cairo, Egypt, 2009.

[13] Y. O. M. Elhadj, "Statistical Part-of-speech Tagger for Traditional Arabic Texts," Journal of Computer Science, vol. 5, no. 11, pp. 794–800, 2009.

[14] M. Hjouj, A. Alarabeyyat and I. Olab, "Rule-based Approach for Arabic Part of Speech Tagging and Name Entity Recognition," Int. J. of Advanced Computer Science and Applications, vol. 7, no. 6, 2016.

[15] S. Algahtani, W. Black and J. McNaught, "Arabic Part-of-speech Tagging Using Transformation-based Learning," Proc of the 2nd Int. Conf. on Arabic Language Resources and Tools, Cairo, Egypt: The MEDAR Consortium, Apr. 2009.

[16] I. Zeroual et al., "Towards a Standard Part of Speech Tagset for the Arabic Language," Journal of King Saud University - Computer and Information Sciences, vol. 29, no. 2, pp. 171–178, Apr. 2017.

[17] M. Tarawneh and E. AlShawakfa, "A Hybrid Approach for Indexing and Searching the Holy Quran," Jordanian Journal of Computers and Information Technology (JJCIT), vol. 1, no. 1, p. 41, 2015.

[18] T. Zerrouki and A. Balla, "Tashkeela: Novel Corpus of Arabic Vocalized Texts, Data for Auto-diacritization Systems," Data in Brief, vol. 11, pp. 147–151, DOI: 10.1016/j.dib.2017.01.011, Apr. 2017.

[19] B. I. Alqudah, "Context-free Rule-based Arabic Text Tagger and Classifier," [Online], Available: http://bilal-qudah.com/arabic/index.php, [Accessed May 5, 2025].

[20] O. Obeid et al., "CAMeL Tools: An Open Source Python Toolkit for Arabic Natural Language Processing," Proc. of the 12th Language Resources and Evaluation Conf., pp. 7022–7032, Marseille, France, 2020, Accessed: Jun. 16, 2025.

[21] K. Darwish et al., "Multi-dialect Arabic POS Tagging: A CRF Approach," Proc. of the 11th Int. Conf. on Language Resources and Evaluation (LREC 2018), pp. 93–98, Miyazaki, Japan, May 2018.

[22] Arabic Language Technology Group, "Farasa POS Tagger," [Online]. Available: https://farasa.qcri.org/POS/ 2020, Accessed: Jun. 16, 2025.

[23] M. Maamouri, A. Bies, T. Buckwalter and H. Jin, "Arabic Treebank: Part 3 v 1.0, LDC2004T11," Linguistic Data Consortium, The Trustees of the University of Pennsylvania, DOI: https://doi.org/10.35111/jf6e-hm83, Accessed: May 21, 2004.

"An Enhanced Word Level Arabic OCR Based on Dual Encoder Transformer Architecture", K. Gaashan and M. Bani Younes.

ملخص البحث:

في هذا البحث، نُعالج التّحديات المرتبطة بتحديد أنواع أجزاء الكلام وتصنيف كلمات نُصوص اللُّغة العربية حيث يكون تركيب الكلمات هو موضوع البحث. وسيكون تركيزنا على العربية الكلاسيكية والعربية القياسية الحديثة.

وتجدر الإشارة إلى أنّ طريقتنا المقترحة لا تحتاج إلى مُعجم أو عمليات استخراج جذور الكلمات أو تقنيات ذكاء اصطناعي. فالهدف هو تقليل المصادر اللّازمة لتصنيف الكلمات في النّصوص العربية. وترتكز طريقتنا على أنّ كُلّ فِعلٍ في اللُّغة العربية يتبع نمطاً معيّناً نشير إليه بالوزْن أو التّفعيل ، ومن الممكن استغلاله لتحديد نوع الكلمة. وكلّ صيغةٍ من صِيَغ الأفعال يتم تمثيلها بمجموعةٍ من التّعبيرات المنتظمة، ويُعدّ التّرتيب الّذي تتمّ به معالجة هذه التّعبيرات المنتظمة أمراً حاسماً بالنّسبة لدقّة النّتائج. فإذا وُجِد توافق، يتمّ وَسْم الكلمة لمنع التّوافقات الأخرى.

الطّريقة المقترحة تتّسم بخفّة الوزن، وأمّا من حيث الأداء، فإنّ زمن التّنفيذ للمصنِّف هو زَمَن خطّي ولا يتطلّب قُدرات معالجةٍ عالية.