Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 00 (2024) 000–000

Procedia Computer Science

www.elsevier.com/locate/procedia

6th International Conference on AI in Computational Linguistics

Alma: Fast Lemmatizer and POS Tagger for Arabic

Mustafa Jarrar[a,*], Diyam Akra[a], Tymaa Hammouda[a]

[a]*Birzeit University, Birzeit, PO Box 14, West Bank, Palestine*

Abstract

We introduce *Alma* (ألمى), an open-source and state-of-the-art lemmatizer, POS tagger, and root tagger for Arabic, boasting both high speed and accuracy. *Alma* relies on a dictionary of morphological solutions ordered by the frequency of these solutions. This dictionary was developed based on the *Qabas* lexicographic database. Unlike many Arabic lemmatizers that return a lemma after stripping diacritics, shadda, and hamza (i.e., ambiguous lemma), *Alma* retrieves unambiguous lemmas (we called *true lemmatization*). Our POS tagger uses a rich tagset of 39 POS tags. Additionally, our root tagger is the first fully-featured tagger since it uses *Qabas*, the largest Arabic lexicographic database. We evaluated *Alma* lemmatizer and POS tagger using the LDC Arabic Treebank (ATB) that contains 339, 710 tokens and achieved 87.82%, 92.7% F1 score, respectively. We additionally evaluated *Alma* lemmatizer using the *Salma* corpus (34*k* tokens) and obtained a 90.48% F1 score. Compared to Farasa, MADAMIRA, and CAMeL Tools lemmatizers and POS taggers, *Alma* outperformed all of them in both tasks, excelling in both speed and accuracy. *Alma* demonstrated superior processing speed, handling 339*k* tokens in 10.00. *Alma* is open-source and publicly available at (https://sina.birzeit.edu/alma).

Keywords: Arabic; Arabic Morphology; morphology tagging; Lemma; Lemmatizer; Part of Speech; POS; POS Tagger; Root; Root Tagger;

1. Introduction

This section overviews the basics of morphology tagging in Arabic, especially lemmatization and part-of-speech (POS) tagging. Lemmatization is the task of determining the lemma of a given word. A lemma (also called *canonical form* or *citation form*) is the dictionary form that is *conventionally* selected from a set of inflected word forms (typically share the same core meaning(s) [23, 19]). In Arabic, the verb lemma is conventionally selected to be in the past, singular, 3rd person, and masculine form, and a noun lemma to be in the singular masculine form [23]. For example, كَتَبَ/ *katab* is the verb lemma for (يَكتُب، يَكتُبون، سَيَكتُب، سَيَكتُبونَها...), and طالب/ *ṭālb* is the noun lemma for (طالبات، طلاب، طلبة، طالبتين).

Lemmatization is a challenging task because a lemma is *conventionally* selected from a dictionary rather than being generated using linguistic rules. Additionally, lemmas are diacritic-sensitive [26]. For example, the lemma بَيَّتَ/ *bayata*

* Corresponding author. Tel.: +970-2-2982000
E-mail address: mjarrar@birzeit.edu

(planned it at night) is not the same as بَيْتٌ/ *baytun* (house). Additionally, different lemmas may have the same spelling. For example, there are two lemmas spelled as بَيْتٌ/ *baytun* , one lemma has the plural بيوت/ *bywt* and means (house), while the other has the plural أبيات/ *ʾabyāt* and means (verse). Most lexicons distinguish between them by adding an index, such as (بَيْتٌ1) and (بَيْتٌ2).

Parts of speech (POS) represent syntactic categories or "word classes" within a language. In Arabic morphology [23], words are traditionally classified into three parts of speech: noun (اِسم), verb (فِعل), and particle (حَرف), which is neither a verb nor a noun. POS tagging is particularly challenging because a word can have multiple POS tags. Additionally, the existence of various Arabic POS tagsets, such as SAMA, CAMeL, Farasa, and *Qabas*, results in incomparable outcomes. Root tagging involves identifying the root for each word in a sentence. In Arabic, roots are usually determined by lexicographers, unlike stems, which are the remaining letters after removing affixes. Furthermore, words in Arabic may have multiple roots. For example, the word صَوْمَعَ has both (ص م ع) and (ص و م ع) as roots.

In this paper, we benchmark several Arabic lemmatizers and POS taggers, including MADAMIRA (good accuracy but slow), Farasa (fast but retrieves ambiguous and undiacritized lemmas), and CAMeL Tools (relatively slow and with low accuracy). To address these gaps, we introduce *Alma*, an open-source, fast, and accurate lemmatizer, POS tagger, and root tagger. *Alma* relies on a dictionary which we computed based on *Qabas* [23] and other resources. *Alma* is implemented as a module within SinaTools [15], an open-source Python toolkit[1] for Arabic NLP developed by SinaLab at Birzeit University.

In summary, the main contributions of this paper are outlined as follows:

1. **Benchmarking the accuracy and speed of four POS tagging tools**, for category, simplified, and True POS tagging using the LDC Arabic Treebank (339, 710 tokens).
2. **Benchmarking the accuracy and speed of lemmatizers,** for full, partial, and non-diacritized lemmas and using two datasets: the Arabic Treebank, and *Salma* corpus.
3. ***Alma*: a fast and accurate lemmatizer, POS tagger, and root tagger**, outperforming all other tools. It achieved an F1 score of 92.7% in $True$ POS tagging using the ATB corpus, and 87.82% in $True$ lemmatization of the ATB corpus (as well as 90.48% using the *Salma* corpus), and with a high speed of lemmatizing $33K$ tokens per second.

The paper is structured as: Section 2 overviews the related work. Section 3 presents the *Alma* system. In Section 4, we detail the experiments conducted and their outcomes. Section 5 concludes the papers and future research.

2. Related Works

Lemmatization and POS tagging tasks were initially tackled using a lexicon-based approach, such as the methods using BAMA and SAMA [7, 12], functioned by extracting all out-of-context lemmas and POS from a stem database with affixes compatibility tables. However, this method faces challenges with out-of-vocabulary words and entails ongoing efforts to expand and update the morphology databases due to the dynamic nature of language.

In addition to the stem-based SAMA database, another valuable resource for lemmatization and POS tagging is *Qabas* [23]. *Qabas* is an open-source lexicographic database encompassing Classical Arabic, Modern Standard Arabic, dialects, and transliterated foreign words. It encompasses about $59k$ lemmas and their morphological features such as spelling variations, root(s), POS, gender, number, person, and voice, as well as semantic information like glosses and synonyms. *Qabas* has facilitated the development of tools and resources for synonym extraction [24, 11, 30, 38] and gloss-context pairs for Word-Sense Disambiguation [2, 36, 1]. Utilizing 110 digitized lexicons [17, 20, 21, 3, 4, 19, 18] and 12 morphologically annotated corpora [22, 14, 39, 27]. *Qabas* links about 255,000 of the 297,000 single-word lemmas and integrates 2.4 million tokens.

Several lemmatizers and POS taggers have been developed based on LDC's BAMA and SAMA morphology databases to disambiguate in-context lemmas and POS tags [8]. Notable examples include MADAMIRA [42] and CALIMA_Star [47]. MADAMIRA leverages the BAMA/SAMA morphological analyzer to obtain a list of all possible out-of-context lemmas and POS tags for a given word. These analyses are then passed to a feature modeling component

[1] *SinaTools* : https://sina.birzeit.edu/sinatools

that utilizes an n-gram language model for lemmatization and a Support Vector Machine (SVM) model for POS tagging. While MADAMIRA has achieved good accuracy for both tasks, it is considerably slow.

To address the slow performance issue of MADAMIRA, the Farasa lemmatizer [37] was introduced as a frequency-based alternative. Farasa searches for input words within a dictionary containing lemmas along with their associated frequencies. Farasa was evaluated against MADAMIRA using the WikiNews dataset, where it demonstrated higher accuracy for Modern Standard Arabic (MSA), achieving 97.32% compared to MADAMIRA's 96.61%. However, this evaluation is problematic because the lemmas retrieved by Farasa are ambiguous, despite the high accuracy unambiguous lemmas are important in many NLP tasks such as word sense disambiguation as shown in [25].

FARASA POS tagger [9] is not frequency-based like Farasa lemmatizer; it relies on an SVM model trained on sets of feature vectors for each word for POS tagging. It is evaluated against MADAMIRA on the Wiki-News dataset, FARASA achieved 96.2% accuracy compared to MADAMIRA's 95.3%. However, despite its high results, FARASA's accuracy may not be considered reliable due to its simplified tagset containing (only 18 tags), which is a reduction from the original 39 tags in the SAMA tagset. This reduction can lead to inaccurate results, as demonstrated in Section 4. FARASA's performance is not high when evaluated on the original 39-tagset SAMA dataset.

In the realm of deep-learning-based approaches, deep learning is frequently employed for POS tagging due to its classification nature, as extensively documented in the literature. Conversely, deep learning approaches are rarely applied to lemmatization, which conventionally involves selecting a lemma from a lexicon rather than classifying it. An illustrative example of this approach is Camelira [40] (The disambiguation module of CAMeL Tools [41]), which utilizes deep learning for POS tagging while adopting a different methodology for lemmatization. Camelira is a recent lemmatizer and POS tagger that employs a hybrid approach using CamelBert [16], a pre-trained language model and morphological analyzer. Camelira BERT achieved 98.7% accuracy in POS tagging when evaluated on a blind set from the Arabic Treebank. However, no precise accuracy is provided for Camelira's lemmatization in the literature.

In addition, the AlKhalil2 tool [6] retrieves a list of possible lemmas and POS tags based on lemma patterns, while another version [5] uses a Hidden Markov Model (HMM) to disambiguate the proper lemma from the list retrieved by AlKhalil2. LemmaTag [32] and the Universal Lemmatizer [28] employ deep learning models to generate lemma characters. UDPIPE [45, 46] and UDify [31, 33] treat both POS tagging and lemmatization as classification tasks using pre-trained language models. They generate a set of rules, classify each lemma to a rule using a deep learning classifier, and then apply the rules in reverse to obtain the lemma.

In the following sections, we evaluate the most commonly used lemmatizers and POS taggers discussed in this section (MADAMIRA, Farasa, and CAMeL Tools). We evaluate them in terms of speed, accuracy, and level of lemma ambiguity. We demonstrate in Section 4 that while some of these tools are fast, they often produce inaccurate ambiguous lemmas. Conversely, others may provide accurate, unambiguous lemmas but are considerably slow.

3. *Alma* System and Memory

This section introduces our *Alma* system and how its memory was constructed. The main idea of *Alma* is to simplify the morphological tagging process (lemmatization, POS tagging, and root tagging) by moving the complexity from the morphological tagger itself to the construction of its memory. Unlike typical morphological taggers, which analyze words in context to predict their lemmas and POS, *Alma* streamlines this complexity through pre-computed morphological tagging memory, simplifying the process into a straightforward lookup operation. By doing this, we achieved a remarkably fast lemmatization and POS tagging, and yet outperform all existing tools in terms of accuracy. For example, as shown in Tables 1 and 2, *Alma* was able to lemmatize the Arabic Tree Bank corpus (340K tokens) in 10 seconds with 87.82% accuracy, while MADAMIRA took 1710 seconds with 84.18% accuracy. The *Alma*'s morphological tagging memory, system, and tagging algorithms are described in the following sections.

3.1. Alma's Memory

Alma's memory is a table that we pre-computed and used inside *Alma* as a lookup table. *Alma*'s memory contains a large number of word forms and their possible morphological solutions. For each word, we collected its morphological solutions and stored them ordered by frequency. A morphological solution consists of a (*lemma*, *POS* , and *root*), and

the frequency of this solution. We implemented this table as a Dictionary (i.e., hashmap) in Python, where the key of the dictionary is the wordform, and the value is a set of morphological solutions ordered by frequency (See Figure 1).

[{"Word Form": "بيت",
"Solutions": [{"lemma":"3 بَيْت", "pos":"NOUN", "root": "ب ي ت", "frequency":596095}, {"lemma":"1 بَيْت", "pos":"NOUN", "root": "ب ي ت",
"frequency":140578}, {"lemma":"4 بَيَت", "pos":"NOUN", "root": "ب ي ت", "frequency":9634}, {"lemma":"1 بَيَّت", "pos":"PV", "root": "ب ي ت",
"frequency":3153}]},
{"Word Form": "بلال",
"Solutions": [{"lemma":"1 بلال", "pos":"'NOUN_PROP'", "root": "ب ل ا ل", "frequency":null}]},
{"Word Form": "احلام",
"Solutions": [{"lemma":"1 حُلم", "pos":"NOUN", "root": "ح ل م", "frequency":212926}, {"lemma":"1 حِلم", "pos":"NOUN", "root": "ح ل م",
"frequency":304}]}]

Fig. 1: *Alma* Memory: example of three wordforms and their solutions

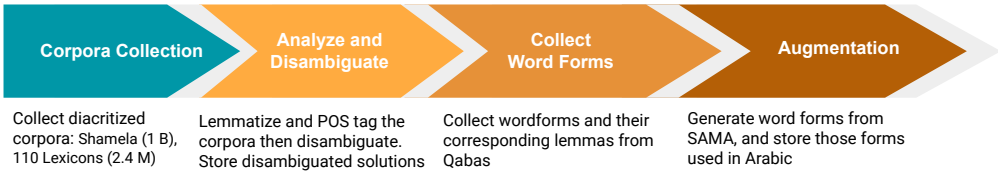

Fig. 2: *Alma* Memory Construction Methodology

Constructing *Alma*'s memory and computing its frequencies was carried out through these phases (See Figure 2):

First Phase (Corpora Collection): We collected two corpora: the Shamela [44] is a collection of books, and the 150 lexicons [20] digitized by the SinaLab at Birzeit University. These lexicons consist of about 2.4 million tokens. About 110 of these lexicons were mapped and integrated with *Qabas* [23], an open-source Arabic lexicon containing 59*k* lemmas. Both Shamela and the 150 lexicons are in diacritized forms of MSA and Classical Arabic.

Second Phase (Morphological Analysis and Disambiguation): Each word in the collected corpora was lemmatized and POS-tagged using the SAMA analyzer [35]. The SAMA analyzer provided all potential solutions for each word, including lemma, POS, diacritized form, and gloss, without disambiguation. To disambiguate the SAMA solutions, we employed the following strategy: (i) we calculated the frequency of each diacritized form in each SAMA solution based on its occurrence in the collected corpora. (ii) we attempted to find a match between one of the diacritized forms of SAMA solutions and the analyzed diacritized word. This process involved multiple attempts. Initially, if there was a single match, it was stored as a new entry in *Alma*'s memory. In the case of multiple matches, the most frequent one was selected and stored as a new entry in *Alma*'s memory and ignored the others. In the case of no match, we stripped the last diacritic, made another matching attempt to find a match, and stored it in *Alma*'s memory.

Third Phase (Collecting Wordfroms from Lexicons): This phase focuses on leveraging the *Qabas* lexicographic graph to gather an extensive collection of wordforms and their corresponding lemmas. As previously mentioned, the 59*k* lemmas in *Qabas* have been manually mapped to 298*k* lemmas across 110 lexicons. This mapping creates a rich Arabic lexicographic graph, enriched further by the inclusion of various wordforms and morphological features. We have gathered these wordforms, both diacritized and undiacritized, along with their POS, root and *Qabas* lemmas, and integrated them into Alma's memory. However, these solutions are integrated into Alma's memory with null frequency.

Fourth Phase (Augmentation): To enhance *Alma*'s memory with words and solutions that were not discovered in the previous phases, we utilized a morphology generator. Specifically, we employed ALMOR [13] to generate word forms from the SAMA database [35]. To ensure that only forms used in Arabic were selected from the generated forms, we employed the Abu-ElKhair corpus [10], which contains 1.5 billion tokens. If the generated form was used in the Arabic language, we then checked if it had only one solution and did not appear in our collected corpora. If all conditions were met (i.e., used in Arabic, had one solution, and did not appear in our collected corpora), we added it as a new entry with a single solution and a null frequency into *Alma*'s memory.

All SAMA lemmas in phases 2 and 4 are replaced with *Qabas* lemmas, which is straightforward [23].

3.2. Alma's System Description

This section describes how *Alma* works (see Algorithm 1), which goes through these steps: **(1) exact-match-search**, **(2) cleaning-search-again-loop**, **(3) out-of-vocabulary-lemmatization**, **(4) out-of-vocabulary-POS-tagging**, and **(5) out-of-vocabulary-root-tagging**.

Algorithm 1 *Alma* algorithm

```
Require: Sentence
  Words ← tokinze_sentence_into_words
  for word in Words do
      solution = Lookup_most_frequent_solution_from_dict(word, dict)
      if no_solution_found then
          five_norm_steps = [Remove_AL, Replace_h_t,Remove_ diacritics, Remove_ shadda]
          for norm_step in five_norm_steps do
              norm_word = apply_norm_step
              solution = Lookup_most_frequent_solution_from_dict(norm_word, dict)
          end for
          if no_solution_found then
              Return word_0
          end if
      end if
  end for
```

When a user inputs a word to be tagged, *Alma* does the following:

1. Exact-Match-Search: *Alma* initially checks if the input exists in its memory. If there is an exact match, *Alma* retrieves the most frequent lemma and its corresponding POS tag. If not found, it goes to the next phase.

2. Stripping-Search-Again-Loop: in this phase, *Alma* sequentially applies several steps to the input word, including (1) removing prefixes like (ال) (e.g., السِياسَة becomes سِياسَة), (2) replacing final (ه) with (ة) (e.g., سِياسه), (3) unifying Alef variants, (4) removing diacritics (e.g., سياسه), and (5) shadda. After each normalization step, *Alma* searches its memory again. If none of these steps yield a match, the word is considered out-of-vocabulary (OOV).

3. Out-of-Vocabulary-Lemma: if a word is not found (e.g., ترامب/Trump), *Alma* handles this by appending a "_0" suffix to the word to denote its OOV (i.e., (e.g., ترامب_0/Trump_0). As discussed in Section 4.4, such OOV cases are typically rare, less than 1%. We plan to continuously add such cases to *Alma*'s memory.

4. Out-of-Vocabulary POS: If a word is not found, *Alma* tries to predict its POS using a BERT model that we fine-tuned using the Arabic Treebank. As shown in Table 3, this model achieved 98.1% F1-score when evaluated on a blind set of ATB and 94.66% when evaluated on *Salma* as a test set. We tested the integration of this model with *Alma* in two scenarios. First, when *Alma* encounters an OOV word in a sentence, it uses the model to predict *all remaining words* in the sentence, as detailed in row (ALMA+BERT) in Table 3. In the second scenario, *Alma* uses the model to predict *only* the OOV word, as detailed in row (ALMA+BERT(OOV)) in Table 3.

5. Out-of-Vocabulary Root: if a word is not found, *Alma* considers the wordform itself as root, for example, the root of ترامب is (ت ر ا م ب).

4. Evaluation

This section benchmarks the accuracy and speed of four tools using two datasets. We used the default settings in all tools for lemmatization and POS tagging, without changing any configuration **Farasa**: we used the jar file found at (https://farasa.qcri.org/POS/), with the parameter (*–lemma true*) for lemmatization. **MADAMIRA**: we used the jar file (MADAMIRA-release-20190603-2.1) found at (https://inventions.techventures.columbia.edu/downloads); its default configuration includes *Aramorph* (*almor − msa − r*13) database, see Pasha et al. [43]. **CAMeL Tools**: we used the disambiguation module, which leverages Camelira (see section 4.2 in Obeid et al. [40]). The disambiguation process invokes the *disambiguate* method from the MSA pretrained model which is part of *BERTUnfactoredDisambiguator*. We evaluated each token, in context, in both datasets using each of the four tools. The reported accuracy excluded digits and punctuation.

4.1. Benchmarking Datasets

We selected two datasets for benchmarking, the Arabic TreeBank (ATB) and the *Salma* corpus, especially as these datasets are MSA and utilize diacritized SAMA lemmas [12]. The LDC's **Arabic Treebank (ATB)** [34] was compiled from news wires between 2001 and 2011. The corpus is tokenized and annotated with features like lemma, POS, segmentation, and gloss. We used Part 3 *V*3.2, which contains 339, 710 tokens, including 32, 135 verbs, 202, 377 nouns, and 105, 198 particles. It has 13, 031unique lemmas (10, 602 nouns, 2, 229 verbs, and 214 particles). The ***Salma* dataset** [25] was recently collected from MSA media sources, between 2021 and 2023. The corpus is tokenized and annotated with various features including lemmas and POS. *Salma* contains 34, 253 tokens, including 2, 763 verbs, 19, 030 nouns, and 12, 460 particles. It has 3, 875 unique lemmas (2, 904 nouns, 677 verbs, and 294 particles). Using *Salma* for benchmarking is important because some tools, including MADAMIRA and CAMeL Tools, are trained on ATB; thus their benchmarking with ATB may not reflect real-world performance.

4.2. Lemmatization Benchmarking

4.2.1. Benchmarking Methodology

Our lemmatization benchmarking methodology comprises five types of lemmatizations: True, Ambiguous, Ambiguous undiacritized, Loose, and Baggy. To illustrate these variations, the word (بيت/ *byt*) has four possible lemmas {بَيْت 1, بَيْت 3, بَيْت 4, بَيَّت 1}, each referring to a different lexeme (i.e., different set of inflections). For example, the lemma (بَيْت 1) refers to the proper noun "Beit", (بَيْت 3) refers to (بيت بيوت) meaning "house", (بَيْت 4) refers (بيت أبيات) meaning "verse", and (بَيَّت 1) refers to the verb with meaning "housing". Notice that the first three lemmas have the same spelling except the digits. Notice also that if digits, diacritics, and Shadda are removed, the four lemmas would have the same spelling.

Since both corpora use SAMA lemmas but lemmatizers produce different lemma spellings, comparing them is challenging. MADAMIRA retrieves SAMA lemmas, and *Alma* lemmas are mapped into SAMA lemmas, making them comparable. However, lemmas in CAMeL Tools very close to SAMA lemmas without digits. Farasa drops digits, diacritics, Shadda, and Hamza. To compare the four tools, we conducted the following lemmatization scenarios:

(1) True Lemmatization: is the exact matching of lemmas, as appeared in both corpora (i.e., SAMA lemmas).

(2) Ambiguous Lemmatization: is the True lemmatization after removing digits. For example, the four lemmas shown in the previous example would be two lemmas after removing digits, as {بَيْت, بَيَّت}.

(3) Ambiguous undiacritized lemmatization: is the True lemmatization after removing digits and all diacritics except Shadda. The four lemmas shown in the previous example would be two lemmas, as {بيت, بيّت}.

(4) Loose lemmatization (stripped): is the True lemmatization after removing digits, all diacritics including Shadda. The four lemmas shown in the previous example would be one lemma, as {بيت}.

(5) Baggy lemmatization (stripped and normalized): is the True lemmatization after removing digits, diacritics, Shadda, as well as normalizing all Hamza forms. That is, the three Hamza forms (أ , إ , آ) are normalized into Alf (ا), e.g., the lemmas { أناء , إناء } would be one lemma {اناء}.

4.2.2. Lemmatization Benchmarking Results

Table 1 shows that *Alma* outperforms all other tools on the *Salma* evaluation and on the True and Ambiguous lemmatization for ATB. Although MADAMIRA outperforms *Alma* in the last three ATB experiments, we believe this is because MADAMIRA was trained on ATB (see [42]).

4.2.3. Speed Benchmarking

We evaluated the lemmatization speed of the four tools on the same machine (24 CPU, 47G Memory, CentOS, size 1.3T) using the same setup (reading input from a file and writing output to a file). The experiment was repeated six times, excluding the first run to account for the initial memory loading time of some tools. This is especially true for CAMeL Tools, which loads a BERT model into memory, unlike other tools that don't rely on neural models. That is, the reading and writing of input/output files were considered, while the initial model loading time was not. Table 2 shows the averages and standard deviations. The results indicate that *Alma* is significantly faster than the others, processing 34K tokens per second. This speed is attributed to ALMA's architecture, which uses a Python dictionary for fast memory access and processing.

Table 1: Evaluation of four lemmatizers using two datasets

Experiment	ATB	SALMA
Exp1: True Lemmatization (Exact Match)		
ALMA	**87.82%**	**90.48%**
MADAMIRA	84.18%	85.53%
CAMeL Tools	-	-
Farasa	-	-
Exp2: Ambiguous lemmatization (without Numbers)		
ALMA	**91.50%**	**91.17%**
MADAMIRA	88.35%	86.60%
CAMeL Tools	80.88%	81.73%
Farasa	-	-
Exp3: Ambiguous undiacritized lemmatization		
ALMA	93.25%	**93.40%**
MADAMIRA	**96.08%**	93.24%
CAMeL Tools	90.65%	90.83%
Farasa	68.92%	70.78%
Exp4: Loose Lemmatization (striped)		
ALMA	94.42%	**94.97%**
MADAMIRA	**96.73%**	94.48%
CAMeL Tools	93.64%	93.29%
Farasa	86.73%	91.28%
Exp5: Baggy lemmatization (striped and normalized)		
ALMA	95.01%	**95.88%**
MADAMIRA	**96.94%**	94.55%
CAMeL Tools	94.81%	94.73%
Farasa	94.86%	95.33%

Table 2: Lemmatization speed with standard deviation (seconds) calculated after six runs excluding the first run.

Experiment	ATB (seconds)	SALMA (seconds)	Speed (tokens per second)
ALMA	$10^{\pm 1}$	$1^{\pm .11}$	33,997
MADAMIRA	$1710^{\pm 99}$	$370^{\pm 49}$	180
CAMeL Tools	$14,398^{\pm 1234}$	$677^{\pm 28}$	25
Farasa	$35^{\pm 2}$	$14^{\pm 2}$	7632

4.3. POS Tagging Benchmarking

4.3.1. Benchmarking Methodology

Benchmarking POS taggers is challenging as they use different tagsets. Specifically, *Alma* and MADAMIRA use the SAMA tagset (39 tags). CAMeL Tools uses these 39 tags but with different naming [29]. However, Farasa uses a tagset of 18 tags [9]. For example, (مصطفى) is NOUN_PROP for MADAMIRA, and NOUN for Farasa, and (يذهب) is *IV* for all taggers but it is *V* for Farasa. To address this problem, we defined three types of POS tagging scenarios:

1. True POS tagging: We use the full 39 tags used in the ATB . We only normalize tag names. For example, we normalize "PRON_REL" in CAMeL Tools into "REL_PRON".

2. Simplified POS tagging: We map the tagsets of all tools into a *simplified* tagset of 13 tags. For example, we mapped {NOUN, NOUN_PROP, NOUN_QUANT, NOUN_NUM, NUM} into NOUN. See the table in the appendix.

3. POS Category tagging: We categorize the tagsets of all tools into three tags {NOUN, VERB, FUNC_WORD}.

4.3.2. POS Benchmarking Results

We evaluated all POS taggers, and evaluated *Alma*, with and without the use of the BERT model that we fine-tuned earlier (see Section 3.2). As shown in Table 3, *Alma* alone (without the use of any BERT model) outperformed all tools in True and Simplified POS tagging. Additionally, *Alma* with BERT outperformed all tools in all settings.

Table 3: Evaluation of *Alma* POS tagging accuracy compared with different morphological analyzers

	ATB		
	True POS	POS Simplified	POS Category
Alma	92.7%	93.1%	97.5%
Alma +BERT	**98.2%**	**98.6%**	**99.5%**
Alma +BERT(OOV)	92.7%	93.1%	97.6%
BERT	98.1%	98.6%	99.5%
MADAMIRA	82.3%	90.8%	97.5%
CAMeL Tools	82.7%	91.9%	98.6%
Farasa	62.4%	84.9%	94.3%

4.4. Discussion and Error Analysis

In this section, we will discuss the cases where *Alma* produced the wrong lemmatization and POS tagging.

4.4.1. Lemmatization Error Analysis

The number of tokens that *Alma* did not lemmatize correctly is 34466 in ATB and 2679 in *Salma*. We categorize these lemmatization errors into three categories and present statistics about each category (see Table 4). About 60.7% of the errors are Ambiguous lemmatization and mostly with noun. It is also notable that the OOV errors are marginal.

Ambiguous Lemmatization: In this type of error, both the wrong lemma and the correct lemma are considered lemmas for the input word, but in different contexts. The wrong lemma retrieved by *Alma* is because it is more frequent. Examples of this category include *Alma* retrieving (1 سِياسَة) which refers to 'politics', while the correct lemma is (2 سِياسَة) which refers to 'policy'. Another example is *Alma* retrieving (1 سَعُودِيّ) which refers to 'Saudi' people, while the correct lemma is (1 سَعُودِيَّة) which refers to 'Saudi Arabia'. Another example is *Alma* retrieving (1 لٰكِنَّ) which is used for emphasis, while the correct lemma is (1 لَكِنْ) which is used as a conjunction. The error in this category can be considered partially incorrect since *Alma* retrieved an ambiguous lemma.

Wrong Lemmatization: The lemma retrieved by *Alma* is completely wrong. This is mostly because the corpora are not diacritized, and therefore *Alma* cannot match them with any diacritized disambiguated entry in its memory. As a result, *Alma* resorts to non-diacritized entries that contain multiple solutions and selects the most common one. For example, consider the word (فهم) in *Salma*, which refers to 'understanding' was lemmatized with the lemma (1 هُم ١) meaning "and them," since it is more frequent.

Out_of_vocabulary Lemmatization: *Alma* failed to retrieve a lemma for words not found in its memory, using the exact match or after applying all the normalization steps (discussed in Section 3.2). In such cases, *Alma* appends a "_0" suffix to the word and retrieves it to indicate the absence of a lemma.

4.4.2. POS Error Analysis

Alma (without BERT) made 20,676 POS tagging mistakes in ATB. Table 5 presents the percentage of each type of mistake. More than half of the POS mistakes are General POS errors. This means that *Alma* can retrieve the correct POS category in most cases. The OOV errors are few.

General POS: As discussed in section 4.3, the 39 tags are categorized into 13 simplified POS tags which are also categorized into 3 categories. For example, ADJ is a type of NOUN. The General POS error means that *Alma* could not determine the exact POS tag but it was able to determine its upper POS. For example, the word آليات is tagged as ADJ by *Alma*, while the correct POS is NOUN.

Table 4: Statistics about the lemmatization errors

Error	ATB				*Salma*			
Category	Noun	Verb	Particle	**All**	Noun	Verb	Particle	**All**
Ambiguous	39.2%[13501]	9.7%[3347]	11.9%[4077]	**60.7%[20925]**	36%[975]	7.0%[192]	18.0%[490]	**61.2%[1657]**
Wrong	23.8%[8208]	6.5%[2236]	8.9%[3063]	**39.2%[13495]**	20.6%[557]	11.7%[316]	6.4%[173]	**38.7%[1046]**
OOV	0.10 %[35]	0.009%[3]	0.02%[8]	**0.13%[46]**	0.18%[5]	0.04%[1]	-	**0.22%[6]**

Wrong POS: *Alma* could not tag a word with its specific POS. For instance, the word (أحب) is assigned the *IV* by *Alma*, while the correct is ADJ_COMP.

Out of Vocabulary: *Alma* fails to retrieve a POS for a word because the word is not found either in its exact match or after applying all the normalization steps in *Alma*'s memory.

Table 5: Statistics about the POS tagging errors

	ATB			
	Noun	Verb	Particle	All
General POS	48.4%[10012]	4.5%[938]	6.2%[1281]	**58.9%[12172]**
Wrong POS	14.2%[2933]	14.9%[3075]	11.8%[2428]	**40.9%[8444]**
OOV	0.22%[47]	0.03%[8]	0.02%[5]	**0.29%[60]**

5. Conclusions and Future works

In this paper, we introduce *Alma*, a fast and accurate Arabic lemmatizer, POS tagger, and root tagger. *Alma* is built upon a large memory that contains both diacritized and non-diacritized Arabic words along with their lemmas, POS, root and frequencies. We benchmarked *Alma* and compared it with other tools, including MADAMIRA, Farasa, and CAMeL Tools, using the *Salma* and ATB corpora under various experimental configurations. Our results demonstrate that *Alma* outperforms all existing systems in both speed and accuracy. In the future, we aim to further enhance *Alma* by enlarging its memory and improving its algorithm. We also plan to extend *Alma* to support Arabic dialects.

References

[1] Al-Hajj, M., Jarrar, M., 2021a. ArabGlossBERT: Fine-Tuning BERT on Context-Gloss Pairs for WSD., in: Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021), INCOMA Ltd., Online. pp. 40–48.

[2] Al-Hajj, M., Jarrar, M., 2021b. LU-BZU at SemEval-2021 Task 2: Word2Vec and Lemma2vec Performance in Arabic Word-in-Context Disambiguation., in: Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021), ACL, Online. pp. 748–755.

[3] Alhafi, D., Deik, A., Jarrar, M., 2019. Usability evaluation of lexicographic e-services, in: The 2019 IEEE/ACS 16th International Conference on Computer Systems and Applications (AICCSA), IEE. pp. 1–7.

[4] Amayreh, H., Dwaikat, M., Jarrar, M., 2019. Lexicon digitization-a framework for structuring, normalizing and cleaning lexical entries. Technical Report, Birzeit University .

[5] Boudchiche, M., Mazroui, A., 2019. A hybrid approach for arabic lemmatization. International Journal of Speech Technology .

[6] Boudchiche, M., Mazroui, A., Bebah, M.O.A.O., Lakhouaja, A., Boudlal, A., 2017. Alkhalil morpho sys 2: A robust arabic morpho-syntactic analyzer. Journal of King Saud University - Computer and Information Sciences , 141–146.

[7] Buckwalter, T., 2004. Buckwalter arabic morphological analyzer version 2.0. (LDC2004L02) .

[8] Darwish, K., Habash, N., Abbas, M., Al-Khalifa, H., Al-Natsheh, H.T., Bouamor, H., Bouzoubaa, K., Cavalli-Sforza, V., El-Beltagy, S.R., El-Hajj, W., Jarrar, M., Mubarak, H., 2021. A Panoramic Survey of Natural Language Processing in the Arab Worlds. Com. ACM 64, 72–81.

[9] Darwish, K., Mubarak, H., Abdelali, A., Eldesouki, M., 2017. Arabic pos tagging: Don't abandon feature engineering just yet. WANLP .

[10] El-Khair, I.A., 2016. 1.5 billion words arabic corpus.

[11] Ghanem, S., Jarrar, M., Jarrar, R., Bounhas, I., 2023. A Benchmark and Scoring Algorithm for Enriching Arabic Synonyms, in: Proceedings of the 12th International Global Wordnet Conference (GWC2023), Global Wordnet Association. pp. 215–222.

[12] Graff, D., Maamouri, M., Bouziri, B., Krouna, S., Kulick, S., Buckwalter, T., 2010. Standard arabic morphological analyzer(sama) v3.1(ldc2010l01) .

[13] Habash, N., 2007. Arabic morphological representations for machine translation, in: Arabic computational morphology: Knowledge-based and empirical methods, Springer. pp. 263–285.

[14] Haff, K.E., Jarrar, M., Hammouda, T., Zaraket, F., 2022. Curras + Baladi: Towards a Levantine Corpus, in: LREC 2022, Marseille, France.

[15] Hammouda, T., Jarrar, M., Khalilia, M., 2024. SinaTools: Open Source Toolkit for Arabic Natural Language Understanding, in: Proceedings of the 2024 AI in Computational Linguistics (ACLING 2024), ELSEVIER, Dubai.

[16] Inoue, G., Khalifa, S., Habash, N., 2022. Morphosyntactic tagging with pre-trained language models for arabic and its dialects. Findings of the Association for Computational Linguistics: ACL 2022 .

[17] Jarrar, M., 2011. Building a formal arabic ontology (invited paper), in: Proceedings of the Experts Meeting on Arabic Ontologies and Semantic Networks, ALECSO, Arab League.

[18] Jarrar, M., 2020. Digitization of Arabic Lexicons. UAE Ministry of Culture and Youth. pp. 214–217.

[19] Jarrar, M., 2021. The Arabic Ontology - An Arabic Wordnet with Ontologically Clean Content. Applied Ontology Journal 16, 1–26.

[20] Jarrar, M., Amayreh, H., 2019. An arabic-multilingual database with a lexicographic search engine, in: The 24th International Conference on Applications of Natural Language to Information Systems (NLDB 2019), Springer. pp. 234–246.

[21] Jarrar, M., Amayreh, H., McCrae, J.P., 2019. Representing arabic lexicons in lemon - a preliminary study, in: The 2nd Conference on Language, Data and Knowledge (LDK 2019), CEUR Workshop Proceedings. pp. 29–33.

[22] Jarrar, M., Habash, N., Alrimawi, F., Akra, D., Zalmout, N., 2017. Curras: An annotated corpus for the palestinian arabic dialect. Journal Language Resources and Evaluation 51, 745–775.

[23] Jarrar, M., Hammouda, T.H., 2024. Qabas: An Open-Source Arabic Lexicographic Database, in: Proceedings of LREC-COLING 2024, ELRA and ICCL, Torino, Italy. pp. 13363–13370.

[24] Jarrar, M., Karajah, E., Khalifa, M., Shaalan, K., 2021. Extracting Synonyms from Bilingual Dictionaries, in: Proceedings of the 11th International Global Wordnet Conference (GWC2021), Global Wordnet Association. pp. 215–222.

[25] Jarrar, M., Malaysha, S., Hammouda, T., Khalilia, M., 2023a. SALMA: Arabic Sense-annotated Corpus and WSD Benchmarks, in: Proceedings of ArabicNLP, Part of the EMNLP 2023, ACL. pp. 359–369.

[26] Jarrar, M., Zaraket, F., Asia, R., Amayreh, H., 2018. Diacritic-based matching of arabic words. ACM Asian and Low-Resource Language Information Processing 18, 10:1–10:21.

[27] Jarrar, M., Zaraket, F., Hammouda, T., Alavi, D.M., Waahlisch, M., 2023b. Lisan: Yemeni, Irqi, Libyan, and Sudanese Arabic Dialect Copora with Morphological Annotations, in: Proceedings of the 20th ACS/IEEE AICCSA, IEEE.

[28] Kanerva, J., Ginter, F., Salakoski, T., 2021. Universal lemmatizer: A sequence-to-sequence model for lemmatizing universal dependencies treebanks. Natural Language Engineering 27, 545–574.

[29] Khalifa, S., Habash, N., Eryani, F., Obeid, O., Abdulrahim, D., Kaabi, M.A., 2018. A morphologically annotated corpus of emirati arabic. LREC 2018 .

[30] Khallaf, N., Arfon, E., El-Haj, M., Morris, J., Knight, D., Rayson, P., Jarrar, T.H.M., 2023. Open-source thesaurus development for under-resourced languages: a welsh case study, in: The 4th Conference on Language, Data and Knowledge (LDK2023).

[31] Kondratyuk, D., 2019. Cross-lingual lemmatization and morphology tagging with two-stage multilingual bert fine-tuning, in: Proceedings of the 16th workshop on computational research in phonetics, phonology, and morphology, pp. 12–18.

[32] Kondratyuk, D., Gavenčiak, T., Straka, M., Hajič, J., 2018. Lemmatag: Jointly tagging and lemmatizing for morphologically rich languages with brnns. EMNLP 2018 .

[33] Kondratyuk, D., Straka, M., 2019. 75 languages, 1 model: Parsing universal dependencies universally, in: EMNLP-IJCNLP.

[34] Maamouri, M., Bies, A., Kulick, S., Krouna, S., Gaddeche, F., Zaghouani, W., 2010a. Arabic treebank: Part 3 v 3. (LDC2010T08) .

[35] Maamouri, M., Graff, D., Bouziri, B., Krouna, S., Bies, A., Kulick, S., 2010b. Standard arabic morphological analyzer(sama) v3.1(ldc2010l01) .

[36] Malaysha, S., Jarrar, M., Khalilia, M., 2023. Context-Gloss Augmentation for Improving Arabic Target Sense Verification, in: GWC2023.

[37] Mubarak, H., 2018. Build fast and accurate lemmatization for arabic. LREC 2018 .

[38] Naser-Karajah, E., Arman, N., Jarrar, M., 2021. Current Trends and Approaches in Synonyms Extraction: Potential Adaptation to Arabic, in: Proceedings of the 2021 International Conference on Information Technology (ICIT), IEEE, Amman, Jordan. pp. 428–434.

[39] Nayouf, A., Jarrar, M., zaraket, F., Hammouda, T., Kurdy, M.B., 2023. Nâbra: Syrian Arabic Dialects with Morphological Annotations, in: Proceedings of ArabicNLP, Part of the EMNLP 2023, ACL. pp. 12–23.

[40] Obeid, O., Inoue, G., Habash, N., 2022. Camelira: An arabic multi-dialect morphological disambiguator, in: EMNLP-Demos, pp. 319–326.

[41] Obeid, O., Zalmout, N., Khalifa, S., Taji, D., Oudah, M., Alhafni, B., Inoue, G., Eryani, F., Erdmann, A., Habash, N., 2020. Camel tools: An open source python toolkit for arabic natural language processing, in: Proceedings of the twelfth LREC, pp. 7022–7032.

[42] Pasha, A., Al-Badrashiny, M., Diab, M., El Kholy, A., Eskander, R., Habash, N., Pooleery, M., Rambow, O., Roth, R.M., 2014. Madamira: A fast, comprehensive tool for morphological analysis and disambiguation of arabic, in: LREC 2014, pp. 1094–1101.

[43] Pasha, A., Al-Badrashiny, M., Diab, M., Habash, N., Pooleery, M., Rambow, O., Roth, R., 2019. Madamira v2.1 user manual.

[44] Shamela Website, . https://shamela.ws/. Accessed: 2023-01-01.

[45] Straka, M., 2018. Udpipe 2.0 prototype at conll 2018 ud shared task, in: CoNLL 2018, pp. 197–207.

[46] Straka, M., Straková, J., Hajic, J., 2019. Evaluating contextualized embeddings on 54 languages in pos tagging. Lemmatization and Dependency Parsing. .

[47] Taji, D., Khalifa, S., Obeid, O., Eryani, F., Habash, N., 2018. An arabic morphological analyzer and generator with copious features, in: Proceedings of the fifteenth workshop on computational research in phonetics, phonology, and morphology, pp. 140–150.

Appendix A. POS Tagging Benchmarking

Table A.6: Categorization and mapping between POS tagsets. The (SAMA-based) means that is used in (*Alma*, MADAMIRA, and Camel). Some Camel tags have slightly different names that we unified in this table

POS Category	POS Simplified	POS
NOUN	NOUN	NOUN (All)
		NOUN_PROP (SAMA-based)
		NOUN_QUANT (SAMA-based)
		NOUN_NUM (SAMA-based)
		NUM (Farasa)
	ADJ	ADJ (ALL)
		ADJ_COMP (SAMA-based)
		ADJ_NUM (SAMA-based)
VERB	Verb	IV (SAMA-based)
		PV (SAMA-based)
		CV (SAMA-based)
		V (Farasa)
		IV_PASS (SAMA-based)
		PV_PASS (SAMA-based)
		PSEUDO_VERB (SAMA-based)
		VERB (SAMA-based)
		Verb * (MADAMIRA and Camel, with IV,PV Aspects/Passive, Active Voice)
FUNC_WORD	Adverb	ADV (All)
		INTERROG_ADV (SAMA-based)
		REL_ADV (SAMA-based)
	Pronoun	pron (All)
		DEM_PRON (SAMA-based)
		EXCLAM_PRON (SAMA-based)
		INTERROG_PRON (SAMA-based)
		REL_PRON (SAMA-based)
	Particle	PART (All)
		DET (SAMA-based)
		FOCUS_PART (SAMA-based)
		FUT_PART (SAMA-based)
		INTERROG_PART (SAMA-based)
		NEG_PART (SAMA-based)
		RESTRIC_PART (SAMA-based)
		VERB_PART (SAMA-based)
		VOC_PART (SAMA-based)
	Conjunction	CONJ (All)
		SUB_CONJ (SAMA-based)
	Preposition	prep
	Interjection	interj
	Abbreviation	abbrev
	Foreign	Foreign