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

SinaTools : Open Source Toolkit for Arabic Natural Language Processing

Tymaa Hammouda[a], Mustafa Jarrar[a,*], Mohammed Khalilia[a]

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

Abstract

We introduce *SinaTools* , an open-source Python package for Arabic natural language processing and understanding. *SinaTools* is a unified package allowing people to integrate it into their system workflow, offering solutions for various tasks such as flat and nested Named Entity Recognition (NER), fully-flagged Word Sense Disambiguation (WSD), Semantic Relatedness, Synonymy Extractions and Evaluation, Lemmatization, Part-of-speech Tagging, Root Tagging, and additional helper utilities such as corpus processing, text stripping methods, and diacritic-aware word matching. This paper presents *SinaTools* and its benchmarking results, demonstrating that *SinaTools* outperforms all similar tools on the aforementioned tasks, such as Flat NER (87.33%), Nested NER (89.42%), WSD (82.63%), Semantic Relatedness (0.49 Spearman rank), Lemmatization (90.5%), POS tagging (97.5%), among others. *SinaTools* can be downloaded from (https://sina.birzeit.edu/sinatools).

Keywords: Toolkit; Arabic; Named Entity Recognition; Word Sense Disambiguation; Semantic Relatedness; Synonymy Extraction; Lemmatization; Part-of-speech Tagging; Root tagging; Morphology; NLP; NLU.

1. Introduction

Despite the progress in Arabic NLP [9], there remain a lack of tools and resources that offer solutions for Arabic NLP and NLU tasks. Developing machine learning tools is crucial in democratizing Arabic NLP, as they allow people to incorporate machine learning into their workflows with less technical knowledge. The availability of open-source and advanced AI tools remains limited. Low-code platforms and toolkits can bridge this gap by offering intuitive interfaces and trained models, making it easier for people in industry, research, and education to tap into the power of NLP to develop and deploy NLP applications.

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

Fig. 1: Core modules of *SinaTools*

As will be discussed in this paper, a handful of tools for Arabic NLP have emerged, each offering rich functionalities that contribute to the growing ecosystem of Arabic NLP. Notable examples include The Stanford CoreNLP Toolkit [42], Farasa [10], MADAMIRA [49], CAMeLTools [46], and OCTOPUS [12].

This article presents a new set of tools packaged in *SinaTools* , an open-source toolkit for Arabic NLP and NLU, offering state-of-the-art solutions for various semantic-related tasks developed by SinaLab at Birzeit University. *SinaTools* currently supports flat, nested, and fine-grained Named Entity Recognition (NER), Word Sense Disambiguation (WSD), Semantic Relatedness, Synonymy Extraction and Evaluation, Lemmatization, Part-of-Speech (POS) tagging, and root tagging, among others. *SinaTools* provides a single end-to-end system for these tasks with different interfaces, including a Command Line Interface (CLI), Software Development Kit (SDK), and Application Programming Interface (API), as well as Python Jupyter Notebooks. Our goal is to simplify the development and deployment of Arabic NLP applications. Additionally, this article presents our benchmarking results of similar toolkits on the aforementioned tasks, demonstrating the superiority of *SinaTools* over other tools. The main contributions of this article are:

1. Open-source[1] and Python-based Arabic toolkit for various NLU tasks, with different interfaces.
2. Benchmarking Arabic NLP toolkits, demonstrating *SinaTools* benefits and superior performance on all tasks.

This article is structured as follows: Section 2 overviews related work. Section 3 presents the design and implementation. Section 4 presents all NLU modules. Section 5 concludes the paper and outlines our future work.

2. Related Work

This section overviews different Arabic NLP toolkits, which we also summarize in Table 1. The Stanford CoreNLP [42] is a general-purpose toolkit, supporting nine languages including Arabic, but its Arabic support is limited to basic tasks. MADAMIRA [49] is designed for morphological analysis and disambiguation, including lemmatization, POS tagging, and segmentation. As will be demonstrated later, MADAMIRA is among the top-performing tools in terms of accuracy. However, it is not open-source, is notably slow, and its functionalities are limited to morphology tasks. Farasa [10] focuses also on Arabic morphology tasks, mainly lemmatization, POS tagging, and segmentation. Farasa is designed for speed, offering fast morphological analysis. However, its results are less sophisticated compared to MADAMIRA. For instance, Farasa's POS tagger is limited to 20 tags, whereas MADAMIRA supports 40 tags. Additionally, Farasa's lemmatizer returns ambiguous lemmas as it removes all diacritics.

CAMeLTools [46] is an open-source NLP toolkit providing utilities for pre-processing, morphological modeling, dialect identification, NER, and sentiment analysis. Although CAMeLTools covers a broad range of tasks, its support of NLU tasks is limited. For example, its NER module can only detect three flat entity types: PERS, LOC, and ORG.

[1] Download page: https://sina.birzeit.edu/sinatools

OCTOPUS [12] introduces a significant advancement with its AraT5v2 model, specifically designed for Arabic text generation. Meticulously trained on extensive and diverse datasets, the tools outperform competitive baselines across various tasks. OCTOPUS supports eight Arabic generation tasks, including summarization, paraphrasing, grammatical error correction, and question generation, among other generative tasks.

There are other task-specific tools. For example, TURJUMAN [43] is a neural machine translation toolkit that translates from 20 dialects into MSA using the AraT5 model. AraNet [2] supports various social media tasks, including age, dialect, gender, emotion, irony, and sentiment prediction. Mazajak [13] focuses on sentiment analysis.

Table 1: Feature comparison of related Arabic NLP tools.

	SinaTools	**CoreNLP**	**Farasa**	**MADAMIRA**	**CamelTools**	**Octopus**
Language	Python	Java	Java	Java	Python	Python
Command Line Interface (CLI)	✓	✓	✓	✓	✓	
Application Programming Interface (API)	✓	✓	✓	✓	✓	✓
Morphological Modeling				✓	✓	
Morphological Disambiguation		✓	✓	✓	✓	
Diacritization			✓	✓	✓	✓
Tokenization/Segmentation/Stemming		✓	✓	✓	✓	
Lemmatization	✓		✓	✓	✓	
POS Tagging	✓	✓	✓	✓	✓	
Root Tagging	✓					
NER	21 tags (flat, nested)	4 tags flat			3 tags flat	
WSD	✓					
Semantic Relatedness	✓					
Sentiment Analysis					✓	
Dialect ID					✓	
Title Generation						✓
QA						✓
Question Generation						✓
transliteration	✓				✓	✓
grammatical error correction						✓
Paraphrase						✓
Summarization						✓
Synonyms Extraction	✓					

3. Design and Implementation

3.1. Design

SinaTools is an open-source toolkit consisting of a collection of Python application programming interfaces (APIs) and their corresponding command-line tools, which encapsulate these APIs. It adheres to these core design principles:

1. **Modularity**: Each function is encapsulated in its own module, allowing for independent development, testing, and maintenance. This structure facilitates the seamless addition of new features without impacting existing ones.
2. **Extensibility**: The architecture is designed to be extensible, enabling users to easily integrate additional tasks or replace existing components with custom implementations.
3. **User-Friendly APIs**: *SinaTools* provides intuitive and consistent APIs that abstract the complexity of underlying algorithms and data structures. This ensures that users, regardless of their expertise in NLP, can leverage the full capabilities of the tools with a minimal learning curve.
4. **Performance Optimization**: The implementation emphasizes efficient processing of large datasets. Large models and datasets required for various tasks are loaded the first time they are used or through a specific implemented download command. This approach minimizes load times and ensures efficient memory usage during operations.

3.2. Implementation

SinaTools is implemented in Python 3.10.8 and can be installed via pip install. Python was selected due to its ease of use and its widespread adoption for NLP and Machine Learning. *SinaTools* is designed to be compatible with Python version 3.10 on Linux, macOS, and Windows. Currently, *SinaTools* provides both an API and a command-line interface

for each component, in addition to demo pages. It is important to note that *SinaTools* is under continuous development, with ongoing additions of new features and updates to existing ones. This paper reports on the current components, but updates and new components will be continuously published.

4. *SinaTools* Modules

4.1. Morphology Module

This module is an implementation[2] of the *Alma* (ألمى) morphology tagger presented in [22], which consists of three sub-modules: (1) lemmatizer (2) POS tagger, and (3) root tagger. These sub-modules utilize a pre-computed memory, consisting of a dictionary containing many wordforms and their morphological solutions. This dictionary is implemented as a Python hashmap, simplifying the lemmatization, POS, and root tagging tasks into straightforward lookup operations. Each entry in the dictionary is a key-value pair, where the key is a wordform and the value is its corresponding morphological solution. A morphological solution consists of a <*lemma*, *POS* , *root*>, and the frequency of this solution. The *Alma* dictionary is frequency-based. We have gathered a large collection of word forms and their morphological solutions from lexicographic resources developed at SinaLab. Using these, we calculated the frequency of each solution (see [22]). The primary resource for building *Alma* is the *Qabas* lexicographic data graph [30], which includes about 58*k* lemmas linked with lemmas in the Arabic Ontology [21, 19], 110 Arabic lexicons [20, 23, 24, 5], and other annotated corpora and resources [45, 35, 15, 28, 27, 26, 18]. The *Alma* dictionary retains the most frequent solution on the top (i.e., default) solution. For example, the wordform (ذهب */dhb*) can be a noun and a verb. However, as the verb form is more common, *SinaTools* consistently tags it as a verb, regardless of the context. Although this method is simple and out-of-context, our evaluations show that it is more accurate and significantly faster than others.

Table 2: Benchmarking Arabic lemmatizers and POS taggers using the ATB and *Salma* datasets

Tool	Lemmatization exact match (F1-Score)		Lemmatization without Numbers (F1-Score)		POS 40 tags (F1-Score)	POS 3 tags (F1-Score)	Speed (seconds)	
	ATB	*Salma*	ATB	*Salma*	ATB	ATB	ATB	*Salma*
MADAMIRA	84.2%	85.5%	88.3%	86.6%	82.3%	97.5%	1661.3	388.57
CamelTools	-	-	80.9%	81.7%	82.7%	98.6%	13080.2	646.06
Farasa	-	-	-	-	62.4%	94.3%	33.22	15.07
SinaTools	87.8%	90.5%	91.5%	91.2%	92.7%	97.5%	11.34	1.19

Evaluation: Table 2 presents the benchmarking results of lemmatization and POS tagging for four tools. We used two datasets: (1) the **LDC's Arabic TreeBank** (ATB) [38], which includes 339, 710 tokens with their morphological annotations, and (2) the ***Salma* dataset** [33], a more recent corpus containing 34, 253 tokens with their morphological and semantic annotations. For lemmatization, we compared the results of all tools in two scenarios: (*i*) with the exact spelling of the lemmas, and (*ii*) after removing numbers from lemmas. Farasa's results did not match in either case, as it provides ambiguous undiacritized lemmas. For POS tagging, we evaluated the tools using the (*i*) full set of 40 tags and (*ii*) a set of 18 tags (see details in [22]). For speed benchmarking, we conducted four runs for each tool on the same machine (24 CPU, 47G Memory, CentOS, 1.3T size) under identical experimental conditions. Excluding the first run, we averaged the speeds of the remaining three runs. *SinaTools* outperformed the other systems in both speed and accuracy across the two corpora. More detailed experimentation and comparisons of the four tools are reported in [22].

Out-of-Vocabulary: As *SinaTools* relies on a dictionary, it cannot provide solutions for wordforms not included in its dictionary. However, our benchmarking showed that out-of-vocabulary (OOV) instances are not a significant issue. To address OOV cases, *SinaTools* integrates a fine-tuned BERT model for POS tagging, ensuring robust performance even when encountering words not present in the dictionary (See [22]).

[2] Demo page (*Alma*): https://sina.birzeit.edu/alma

4.2. Named Entity Recognition Module

This module[3] is based on a BERT model that fine-tuned with our Wojood datasets [32]. The module supports flat, nested, and fine-grain entity types. Since Wojood has 21 entity types, our model includes 21 classification layers, one layer for each entity type. Each layer classifies the token into one of three classes, $C = \{I, O, B\}$. As illustrated in Figure 2, each classifier is an expert in one entity type, which will output one of the three labels in C for each token.

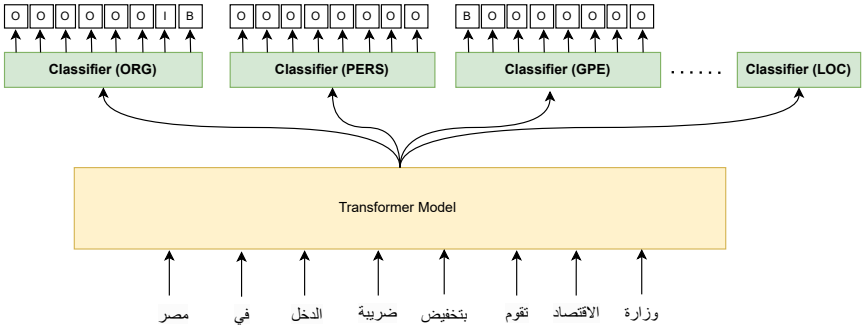

Fig. 2: *SinaTools* NER model

Evaluation: We benchmarked and compared our NER module with CamelTools on three datasets. Table 3 summarizes the results. *Wojood* test set, which covers MSA, Palestinian, and Lebanese dialects, on which *SinaTools* is trained [32]. *Wojood*^{Gaza} provided in subtask-3 in the WojoodNER 2024 shared task [29]. This dataset contains 50k tokens recently collected from five news domains (politics, law, economy, finance, health) related to the Israeli War on Gaza, and annotated using the 21 Wojood tags. *Politics*, a second out-of-domain dataset and contains 12,712 tokens that we collected from Aljazeera news articles two years ago. We note that CamelTools can detect three types of entities only (PERS, LOC, and ORG).

Table 3: Evaluation of *SinaTools* NER module (F1-Score)

Dataset	Tool	Flat (3 tags)	Flat (21 tags)	Nested (21 tags)
Wojood	CamelTools	45.85%	-	-
	SinaTools	**66.40%**	**87.33%**	**89.42%**
Wojood^{Gaza}	CamelTools	29.72%	-	-
	SinaTools	**67.27%**	**55.72%**	**62.68%**
Politics	CamelTools	54.00%	-	-
	SinaTools	**69.00%**	**68.00%**	**74.00%**

4.3. Word Sense Disambiguation Module

This module is an implementation of a novel end-to-end semantic analyzer called *Salma* (سلمى)[4]. Figure 3 illustrates our system architecture, as a pipeline of components: Tokenizer, Lemmatizer, POS Tagger, NER, Target Sense Verification (TSV), and two sense inventories. Given an input sentence, the WSD module conducts semantic analysis, which includes disambiguating single-word and multi-word expressions, and tagging named entities.

Example: Input (وزارة الاقتصاد تقوم بتخفيض ضريبة الدخل في مصر / *Ministry of Economy is reducing income tax in Egypt*). Output: (1) **named entities** (وزارة الاقتصاد/Ministry of Economy)$_{\text{ORG}}$ and (مصر/Egypt)$_{\text{GPE}}$; (2) **multi-word expressions senses** (ضريبة الدخل/income tax) ضَرِيبَةُ دَخْلٌ; and (3) **single-word senses** (تقوم/is)$_{\text{قَامَ-6}}$, (بتخفيض/reducing)$_{\text{تَخْفِيضٌ-2}}$.

The WSD system consists of a pipeline of the following sub-processes:

[3] Demo page (Wojood): https://sina.birzeit.edu/wojood
[4] Demo page (*Salma*): https://sina.birzeit.edu/salma

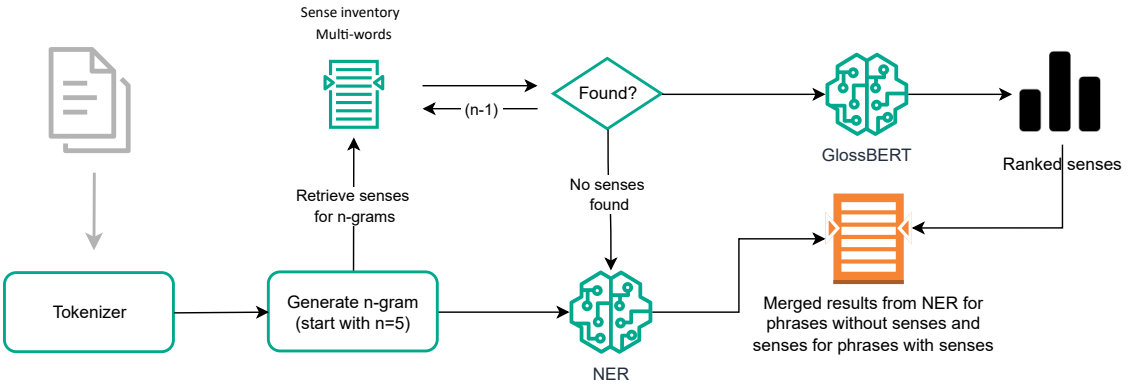

Fig. 3: *SinaTools* end-to-end WSD (*Salma*)

Phase 1 (*n*-gram tokenization): we first generate *n*-grams from the input text for all $2 \leq n \leq 5$. The *n*-grams will be used in later phases to query the sense inventory.
Phase 2 (Lemmatization): Each token in a given *n*-gram is lemmatized using the *SinaTools* Lemmatizer.
Phase 3 (Multi-word WSD): At this phase we retrieve senses for multi-word expressions such as (ضريبة دخل/income tax) and (سنة كبيسة/leap year). *SinaTools* includes a dictionary of such expressions that we collected from our 150 lexicons [23], storing each multi-word expression with its glosses. This dictionary serves as our multi-word sense inventory. For each lemmatized multi-word expression (i.e., *n*-gram, where $2 \leq n \leq 5$), we perform a lookup in the sense inventory, starting with $n = 5$. If a sense is not found, we reduce *n* by 1 and attempt the retrieval again. For example, (ضريبة دخل/income tax) has two glosses (i.e., senses) in the sense inventory. The senses along with the original input sentence are sent to the TSV module for disambiguation (Phase 6).
Phase 4 (NER): As no need to sense-disambiguate all *n*-grams (e.g., text spans such as (مصر/Egypt) and (وزارة الاقتصاد/Ministry of Economy) are proper nouns), the NER module tags each span with entity types.
Phase 5 (single-word WSD): All tokens that were not sense-disambiguated or NER-tagged in the previous steps will undergo an additional step. We disambiguate them as single words (*uni*-grams). For instance, in the example above, three *uni*-grams need to be single-word disambiguated: (تقوم/is), (بتخفيض/reducing), and (في/in). The glosses of these *uni*-grams are retrieved from the single-word sense inventory and sent to the TSV module for disambiguation (Phase 6).
Phase 6 (Target Sense Verification (TSV)): We implemented the TSV approach proposed in [33], which is a core step in the WSD pipeline. Given a pair of sentences (context and gloss), TSV calculates the Positive and Negative probabilities to classify whether this pair is True or False, indicating whether the gloss is the correct sense used in the context based on the higher probability. Thus, for a word *w* in context *C*, and all *w* glosses $g_i \in G$, we generate a set of pairs $\{(C, g_1),(C, g_2),..., (C, g_m)\}$. These pairs are sent to the TSV model for evaluation. The pair with the highest Positive probability is considered the target sense. We fine-tuned the TSV model using AraBERT V2 [7] and the ArabGlossBERT dataset (167*k* pairs) [3]. We experimented with an augmented dataset [40] but it did not enhance the results. We also addressed WSD using the WiC model [4] but the performance was bad.

Evaluation: As *SinaTools* is the first tool for Arabic WSD, we computed its baseline using the *Salma* corpus [33], the only available Arabic sense-annotated corpus. *Salma* is only annotated with NER and single-word expressions, but we extend its annotations to include multi-word expressions. Table 4 shows that *SinaTools* achieves an overall WSD performance of 82.63% accuracy. It's important to note that calculating the accuracy of the single-word WSD is not straightforward because a word might have multiple correct senses in *Salma*, and whether the accuracy should include words with single senses. Thus, we refer the reader to [33] for more details. It is also worth noting that *Salma* was used in the ArabicNLU Shared Task [36], where the best-performing system on single-word WSD achieved 77.8% [50].

4.4. Semantic Relatedness Module

This task is useful in NLP for many scenarios such as evaluating sentence representations [8], document summarization, and question answering. Given two sentences, the semantic relatedness task aims to assess the degree of association

Table 4: Evaluation of *SinaTools* WSD module

	Tokens Count	**Span Count**	**Accuracy**
NER (6 types)	4,389	2,728	85.31%
Multi-word WSD	2,100	519	88.92%
Single-word WSD	27,764	27,764	81.73%
Overall (Micro average)	34,253		**82.63%**

between two sentences across various dimensions, including meaning, underlying concepts, domain-specificity, topic overlap, or viewpoint alignment [1]. *SinaTools* supports MSA semantic relatedness, which represents our participation [41] in the SemRel Shared Task [48], where we achieved the top rank. Unlike the lexical overlap (using dice coefficient) proposed in [47], we extracted the mean-pooling embeddings of the sentences from BERT-based model, then employed cosine similarity as an unsupervised technique to calculate the sentence-pair scores. We evaluated our method using the 595 sentence-pairs test set provided in the SemEval-2024 Task 1 on Semantic Textual Relatedness for African and Asian Languages [48]. Table 5 shows that we outperformed the baseline. We used Spearman rank correlation coefficient, the official evaluation metric used in the shared task, which captures the level to which the system predictions align with the human judgments of the test pairs.

Table 5: Performance of Semantic Relatedness of *SinaTools* on the SemEval-2024 Task

Test Pairs	Spearman	
	Baseline	*SinaTools*
595	0.42	**0.49**↑

4.5. Synonyms Module

Arabic is low-resourced in terms of synonymy resources and tools [44, 31, 17]. *SinaTools* includes an implementation of the synonymy extraction and evaluation[5] algorithm introduced in [14], which was also tested in extracting Welsh synonyms [37]. The algorithm is designed to extract synonyms from mono and multilingual lexicons. It leverages synonymy and translation pairs from these resources to generate a synonymy graph, where nodes involved in cyclic paths are deemed synonyms. The extracted synonyms are assigned fuzzy values to indicate their degree of belonging to the synonym set. The algorithm is evaluated using the Arabic WordNet, achieving 98.7% accuracy at 3rd level and 92% at 4th level. We utilized about 100 mono and bilingual lexicons [23], including the Arabic Ontology [21, 19], Qabas [30], 40 ALECSO lexicons, 11 from the Arabic Academy, WordNet, and others. These resources were used to compute two synonymy graphs: a 2nd level graph ($75MB$) and a 3rd level graph ($1.1GB$). *SinaTools* features two main methods: (i) ***SynExtract*** for synonym extraction, and (ii) ***SynEval*** for synonym evaluation. The $SynExtract$ method allows users to give one or more terms as input and retrieve their synonyms (See example 1). Each synonym is given a fuzzy value. The more terms are provided in the input the better the accuracy. The $SynEval$ method enables users to input a set of terms and receive a fuzzy score for each term, reflecting its synonymy strength (See example 2). In both methods, users can choose between the faster 2nd level graph or the richer but slower 3rd level graph.

Example 1: the function $SynExtract$(طريق , ممر), returns {مسلك 61%, سبيل 50%, نهج 23%, زقاق 20%, سكة 15% , ...}

Example 2: the function $SynEval$(طريق , ممر, مسلك), returns {مسلك 70%, ممر 50%, طريق 30%}.

4.6. Other Modules and Tools

In addition to its core functionalities, *SinaTools* includes a variety of utility modules designed for text processing capabilities. The **Sentence Splitter** module offers the capability to segment text into sentences, with the flexibility to specify separators (e.g., periods, question marks, exclamation marks, and line breaks), thereby accommodating varied text structures. Notably, this feature selectively incorporates chosen separators while disregarding unselected ones,

[5] Demo Page (Synonymy) : https://sina.birzeit.edu/synonyms

thereby enhancing the precision of text segmentation. The **Diacritic-Based Matching of Arabic Words** module[6] compares two Arabic words to find out whether they are the same, taking into account their partial or full diacratizition [34]. For example, the two words (فَعلَ */faʕla* ,فعَل */fʕal*) are compatible and the (فَعلَ */faʕla* ,فِعلَ */fiʕla*) are not. The **Text Duplicate Removal** module employs cosine similarity to eliminate redundant sentences from input text under a *similarity threshold*, which is useful for corpora pre-processing. The **Arabic Jaccard** module computes union, intersection, and similarity metrics between sets of Arabic words, taking into account partial diacritization. Moreover, the **Arabic Diacritic Removal (arStrip)** module designed to cleanse Arabic words by selectively removing diacritics, shaddah, digits, alif, and special characters, according to user-specified parameters. Lastly, the **Transliteration Module** facilitates seamless conversion between Arabic and Buckwalter transliteration schemas, thereby ensuring interoperability across different linguistic representations.

5. Conclusions and Future Work

We presented *SinaTools* , an open-source Python package for Arabic NLP, offering solutions for various tasks such as NER, WSD, semantic relatedness, synonymy extraction and evaluation, lemmatization, POS tagging, among others. Our benchmarking results highlight that *SinaTools* consistently outperforms similar tools across all tasks.

Looking ahead, we plan to expand *SinaTools* by incorporating additional Arabic NLU modules. These enhancements will focus on areas currently underserved by existing tools, such as intent detection [25, 39], relationship extraction [6], detection of hate speech [16], bias and propaganda [11, 51], among others.

References

[1] Abdalla, M., Vishnubhotla, K., Mohammad, S.M., 2023. What Makes Sentences Semantically Related? A textual relatedness Dataset and Empirical Study, in: Proceedings of EACL 2023, ACL. pp. 782–796.
[2] Abdul-Mageed, M., Zhang, C., Hashemi, A., et al., 2020. AraNet: A Deep Learning Toolkit for Arabic Social Media, in: OSACT, pp. 16–23.
[3] 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.
[4] 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.
[5] 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.
[6] Aljabari, A., Duaibes, L., Jarrar, M., Khalilia, M., 2024. Event-Arguments Extraction Corpus and Modeling using BERT for Arabic, in: Proceedings of ArabicNLP'2024, ACL, Bangkok, Thailand.
[7] Antoun, W., Baly, F., Hajj, H., 2020. AraBERT: Transformer-based Model for Arabic Language Understanding, in: OSACT, pp. 9–15.
[8] Asaadi, S., Mohammad, S.M., Kiritchenko, S., 2019. Big BiRD: A Large, Fine-grained, Bigram Relatedness Dataset for Examining Semantic Composition, in: Proceedings of the NAACL-HLT, ACL. pp. 505–516.
[9] 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.
[10] Darwish, K., Mubarak, H., 2016. Farasa: A New Fast and Accurate Arabic Word Segmenter, in: Proceedings of LREC, pp. 1070–1074.
[11] Duaibes, L., Jaber, A., Jarrar, M., Qadi, A., Qandeel, M., 2024. Sina at FigNews 2024: Multilingual Datasets Annotated with Bias and Propaganda, in: Proceedings of ArabicNLP'2024, ACL, Bangkok, Thailand.
[12] Elmadany, A., Nagoudi, E.M.B., Abdul-Mageed, M., 2023. Octopus: A Multitask Model and Toolkit for Arabic Natural Language Generation .
[13] Farha, I.A., Magdy, W., 2019. Mazajak: An Online Arabic Sentiment Analyser, in: Proceedings of WANLP Workshop, pp. 192–198.
[14] 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.
[15] Haff, K.E., Jarrar, M., Hammouda, T., Zaraket, F., 2022. Curras + Baladi: Towards a Levantine Corpus, in: Proceedings of the International Conference on Language Resources and Evaluation (LREC 2022), Marseille, France.
[16] Hamad, N., Jarrar, M., Khalilia, M., Nashif, N., 2023. Offensive Hebrew Corpus and Detection using BERT, in: Proceedings of the 20th ACS/IEEE AICCSA, IEEE.
[17] Helou, M.A., Palmonari, M., Jarrar, M., 2016. Effectiveness of automatic translations for cross-lingual ontology mapping. Journal of Artificial Intelligence Research 55, 165–208.
[18] Jarrar, M., 2008. Towards Effectiveness and Transparency in e-Business Transactions, An Ontology for Customer Complaint Management. IGI Global. chapter 7. pp. 127–149.

[6] Demo Page (Diacritic-Based Matching) : https://sina.birzeit.edu/resources/Implication.html

[19] 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.
[20] Jarrar, M., 2020. Digitization of Arabic Lexicons. UAE Ministry of Culture and Youth. pp. 214–217.
[21] Jarrar, M., 2021. The Arabic Ontology - An Arabic Wordnet with Ontologically Clean Content. Applied Ontology Journal 16, 1–26.
[22] Jarrar, M., Akra, D., Hammouda, T., 2024a. ALMA: Fast Lemmatizer and POS Tagger for Arabic, in: Proceedings of the 2024 AI in Computational Linguistics (ACLING 2024), ELSEVIER, Dubai.
[23] 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.
[24] 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.
[25] Jarrar, M., Birim, A., Khalilia, M., Erden, M., Ghanem, S., 2023a. ArBanking77: Intent Detection Neural Model and a New Dataset in Modern and Dialectical Arabic, in: Proceedings of ArabicNLP, Part of the EMNLP 2023, ACL. pp. 276–287.
[26] Jarrar, M., Deik, A., Faraj, B., 2011. Ontology-based data and process governance framework -the case of e-government interoperability in palestine, in: Proceedings of the IFIP International Symposium on Data-Driven Process Discovery and Analysis (SIMPDA'11), pp. 83–98.
[27] Jarrar, M., Habash, N., Akra, D., Zalmout, N., 2014. Building a corpus for palestinian arabic: a preliminary study, in: Proceedings of the EMNLP 2014, Workshop on Arabic Natural Language, ACL. pp. 18–27.
[28] 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.
[29] Jarrar, M., Hamad, N., Khalilia, M., Talafha, B., Elmadany, A., Abdul-Mageed, M., 2024b. WojoodNER 2024: The Second Arabic Named Entity Recognition Shared Task, in: Proceedings of ArabicNLP'2024, ACL, Bangkok, Thailand.
[30] 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.
[31] 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.
[32] Jarrar, M., Khalilia, M., Ghanem, S., 2022. Wojood: Nested Arabic Named Entity Corpus and Recognition using BERT, in: Proceedings of the International Conference on Language Resources and Evaluation (LREC 2022), Marseille, France.
[33] Jarrar, M., Malaysha, S., Hammouda, T., Khalilia, M., 2023b. SALMA: Arabic Sense-annotated Corpus and WSD Benchmarks, in: Proceedings of ArabicNLP, Part of the EMNLP 2023, ACL. pp. 359–369.
[34] 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.
[35] Jarrar, M., Zaraket, F., Hammouda, T., Alavi, D.M., Waahlisch, M., 2023c. Lisan: Yemeni, Irqi, Libyan, and Sudanese Arabic Dialect Copora with Morphological Annotations, in: Proceedings of the 20th ACS/IEEE AICCSA, IEEE.
[36] Khalilia, M., Malaysha, S., Suwaileh, R., Jarrar, M., Aljabari, A., Elsayed, T., Zitouni, I., 2024. ArabicNLU 2024: The First Arabic Natural Language Understanding Shared Task, in: Proceedings of ArabicNLP'2024, ACL, Bangkok, Thailand.
[37] 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).
[38] Maamouri, M., Bies, A., Kulick, S., Krouna, S., Gaddeche, F., Zaghouani, W., 2010. Arabic Treebank: Part 3 V 3.2 LDC2010T08. LDC.
[39] Malaysha, S., El-Haj, M., Ezzini, S., Khalilia, M., Jarrar, M., Nasser, S., Berrada, I., Bouamor, H., 2024a. AraFinNLP 2024: The First Arabic Financial NLP Shared Task, in: Proceedings of ArabicNLP'2024, ACL, Bangkok, Thailand.
[40] Malaysha, S., Jarrar, M., Khalilia, M., 2023. Context-Gloss Augmentation for Improving Arabic Target Sense Verification, in: Proceedings of the 12th International Global Wordnet Conference (GWC2023), Global Wordnet Association.
[41] Malaysha, S., Jarrar, M., Khalilia, M., 2024b. NLU-STR at SemEval-2024 Task 1: Generative-based Augmentation and Encoder-based Scoring for Semantic Textual Relatedness, in: Proceedings of the SemEval 2024 Shared Task 1 (Semantic Relatedness), ACL.
[42] Manning, C.D., Surdeanu, M., Bauer, J., Finkel, J.R., Bethard, S., McClosky, D., 2014. The Stanford CoreNLP Natural Language Processing Toolkit, in: Proceedings of ACL, pp. 55–60.
[43] Nagoudi, E.M.B., Elmadany, A., Abdul-Mageed, M., 2022. TURJUMAN: A Public Toolkit for Neural Arabic Machine Translation .
[44] 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.
[45] 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.
[46] 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 LERC, pp. 7022–7032.
[47] Ousidhoum, N., Muhammad, S.H., 2024. SemRel2024: A Collection of Semantic Textual Relatedness Datasets for 14 Languages.
[48] Ousidhoum, N., Muhammad, S.H., Abdalla, M., 2024. SemEval-2024 Task 1: Semantic Textual Relatedness for African and Asian Languages, in: Proceedings of SemEval, ACL.
[49] Pasha, A., Al-Badrashiny, M., Diab, M.T., El Kholy, A., Eskander, R., Habash, N., Pooleery, M., Rambow, O., Roth, R., 2014. MADAMIRA: A Fast, Comprehensive Tool for Morphological Analysis and Disambiguation of Arabic, in: LREC, pp. 1094–1101.
[50] Rajpoot, P.K., Jindal, A.K., Parikh, A., 2024. Upaya at ArabicNLU Shared-Task: Arabic Lexical Disambiguation using Large Language Models, in: Proceedings of ArabicNLP, Part of ACL 2024.
[51] Zaghouani, W., Jarrar, M., Habash, N., Bouamor, H., Zitouni, I., Diab, M., El-Beltagy, S., AbuOdeh, M., 2024. The FIGNEWS Shared Task on News Media Narratives, in: Proceedings of ArabicNLP'2024, ACL, Bangkok, Thailand.