An Arabic Morphological Analyzer and Generator with Copious Features

Dima Taji, Salam Khalifa, Ossama Obeid, Fadhl Eryani, and Nizar Habash
Computational Approaches to Modeling Languages Lab
New York University Abu Dhabi
{dima.taji, salamkhalifa, oobeid, fadhl.eryani, nizar.habash}@nyu.edu

Abstract

We introduce CALIMA$_{Star}$, a very rich Arabic morphological analyzer and generator that provides functional and form-based morphological features as well as built-in tokenization, phonological representation, lexical rationality and much more. This tool includes a fast engine that can be easily integrated into other systems, as well as an easy-to-use API and a web interface. CALIMA$_{Star}$ also supports morphological reinflection. We evaluate CALIMA$_{Star}$ against four commonly used analyzers for Arabic in terms of speed and morphological content.

1 Introduction

Work on Modern Standard Arabic (MSA) morphological modeling has been ongoing for the past thirty years resulting in many resources with high degrees of accuracy for analysis, generation, and tokenization (Beesley et al., 1989; Al-Sughaiyer and Al-Kharashi, 2004; Habash, 2010; Pasha et al., 2014; Abdelali et al., 2016). These previous efforts addressed many of the important challenges of Arabic morphology such as its high degree of ambiguity resulting from optional diacritization and templatic morphemes. However, while there are several commonly used systems for Arabic morphology, we observe that there are still some unresolved challenges.

First, some aspects of Arabic's rich morphology are not fully or consistently modeled. Examples include the discrepancy between form and function (in gender, number, case and state) as well as the rationality feature. The commonly used Penn Arabic Treebank (PATB) (Maamouri et al., 2004) and Buckwalter Arabic Morphological Analyzer (BAMA) (Buckwalter, 2002) do not model nominal functional features or rationality. Some previous attempts did not cover all these phenomena or focused on limited data sets (Smrž, 2007; Alkuhlani and Habash, 2011).

Second, the different existing tools do not all provide the same kind of information, which often led researchers to improvise extensions to accommodate their downstream task needs. One example is the phonological representation mappings that Biadsy et al. (2009) devised on top of the MADA disambiguation system (Habash et al., 2009) instead of using Elixir-FM (Smrž, 2007), which already included phonology. This is partially because Elixir-FM was not connected to a disambiguation system. Another example is the work by Habash et al. (2009) to provide generation capability on top of the BAMA (Buckwalter, 2002) algorithm and databases because BAMA, which was used to annotate the PATB, was analysis focused, unlike the finite-state solutions to Arabic morphology (Beesley et al., 1989).

Third, many of the existing tools have different use requirements (operating system, programming language, etc.), and some have no easy-to-use APIs.

In this paper, we introduce CALIMA$_{Star}$,[1] a very rich Arabic morphological analyzer and generator that includes functional features, built-in tokenization, phonological representation, and numerous other features. CALIMA$_{Star}$ comes with a fast engine that can be easily integrated into other systems, and an easy-to-use web interface. CALIMA$_{Star}$ also supports morphological reinflection. While in this paper we focus on MSA only for the database discussion, the engine itself is independent of the variant choice. CALIMA$_{Star}$ will be made publicly available as part of a large suite of tools to support research on Arabic natural language processing (NLP).[2]

[1]In Arabic, كلمة /*kalima*/ means 'word'. We follow and extend the naming convention from Habash et al. (2012) who developed CALIMA$_{\text{EGY}}$, and Khalifa et al. (2017) who developed CALIMA$_{\text{GLF}}$. The *Star* designation in CALIMA$_{Star}$ is intended to eventually represent all Arabic variants (MSA and dialects), and all possible features.

[2]`http://resources.camel-lab.com/`.

2 Related Work

In this section, we discuss previous work on Arabic morphological analysis and generation in terms of (a) algorithms and representations, (b) morphological knowledge, and (c) morphological disambiguation and tokenization. Table 1 compares the features supported by CALIMA$_{Star}$ and a number of analyzers discussed below.

2.1 Algorithms and Representations

There are a number of dimensions over which solutions to Arabic morphology modeling have varied (Beesley et al., 1989; Beesley, 1996; Habash and Rambow, 2006; Smrž, 2007; Altantawy et al., 2010, 2011). One important aspect is the degree of explicitness of representing morphological rules and their interactions. Some approaches use very abstract and linguistically rich representations and rules to derive surface forms of the words (Beesley et al., 1989; Beesley, 1996; Habash and Rambow, 2006; Smrž, 2007). Other approaches pre-compile representations of the different components needed by the system: BAMA (Buckwalter, 2002), SAMA (Graff et al., 2009), and ALMORGEANA (ALMOR for short) (Habash, 2007) are examples of such systems. They use a six-table representation consisting of three lexical tables (for prefixes, suffixes, and stems), and three compatibility tables (prefix-suffix, prefix-stem, and stem-suffix). Altantawy et al. (2011) described a method to bridge between these two types of solutions. The type of representation used naturally needs to synchronize with the appropriate algorithms for analysis and generation. CALIMA$_{Star}$ is of the second category (tabulated pre-compiled solutions), and it builds on the popularly used BAMA, SAMA, and ALMOR morphological analyzers.

2.2 Morphological Knowledge

Previous efforts show a wide range for the depth that morphological analyzers can produce. Some efforts include very shallow analyses such as in the Temple Translator's Workstation Project (Vanni and Zajac, 1996) which only provided English glossing. Others include a range of form-based features, functional features, and morpheme forms, in addition to lexical features (Buckwalter, 2002; Smrž, 2007; Boudlal et al., 2010; Alkuhlani and Habash, 2011; Boudchiche et al., 2017).

Alkuhlani and Habash (2011) extended part of the PATB to include functional gender and number, and rationality, but did not cover the entire database used by BAMA or SAMA. ElixirFM (Smrž, 2007) includes functional gender and number, as well as full case and state modeling, but not rationality. ElixirFM, MAGEAD (Altantawy et al., 2010; Habash and Rambow, 2006) and AlKhalil Morpho Sys (Boudlal et al., 2010; Boudchiche et al., 2017) include roots, with varying degrees of accuracy. ElixirFM includes phonological forms; but ALMOR does not. However, Biadsy et al. (2009) presented orthography-to-phonology rules that can be used on automatically diacritized text to generate pronunciation dictionaries.

Our goal is to make all these features be built-in as part of our CALIMA$_{Star}$ databases, and to fill in any gaps that were left by other efforts.

2.3 Analysis, Disambiguation and Tokenization

We distinguish between analysis and disambiguation: analysis refers to identifying all of the different readings (analyses) of a word out of context; while disambiguation is about identifying the specific analysis in context. Tokenization is the process of segmenting a word into different units for downstream applications. There are many possible tokenization *schemes* and techniques to apply them (Habash and Sadat, 2006). The tokenized form of a word varies depending on the specific analysis of the word. Systems such as MADA (Habash et al., 2009), AMIRA (Diab et al., 2004), and MADAMIRA (Pasha et al., 2014) handle disambiguation and tokenization differently. Both MADA and MADAMIRA disambiguate the analyses that are produced by a morphological analyzer. The chosen analyses are then used to tokenize the words using morphological regeneration. AMIRA, on the other hand, has a different two step process in which a toeknization component is followed by part-of-speech (POS) tagging.

The FARASA system (Abdelali et al., 2016) relies on probabilistic models of stems, prefixes, and suffixes, instead of using context information to produce high tokenization accuracy. YAMAMA (Khalifa et al., 2016) is a MADAMIRA-like (analysis/disambiguation) system that disambiguates using a maximum likelihood model inspired by FARASA.

CALIMA$_{Star}$ is primarily an out-of-context

	BAMA	SAMA	ALMOR	MAGEAD	ElixirFM	AlKhalil	CALIMA$_{Star}$
Functional Gender and Number	✗	✗	✗	partial	partial	partial	✓
Case and State Modeling	partial	partial	partial	✓	✓	partial	✓
Rationality	✗	✗	✗	✗	✗	✗	✓
Roots and Patterns	✗	✗	✗	✓	✓	✓	✓
Phonological Representation	✗	✗	✗	✓	✓	✗	✓
Number of Tokenization Schemes	1	1	1	1	1	1	4
Number of POS Tag Sets	1	1	2	1	1	1	4
Out-of-Context Probabilities	✗	✗	✗	✗	✗	✗	✓
Functionalities	Analysis	Analysis	Analysis	Analysis Generation	*Resolution* *Inflection* *Derivation* *Lookup*	Analysis	Analysis Generation Reinfletion

Table 1: A comparison of the different features in a number of morphological tools. A ✓ denotes a feature that is present in the system, while an ✗ denotes a feature that is not. ElixirFM uses different terminology to denote the analysis, generation, and reinflection functionalities.

analysis and generation system. Its database includes pre-compiled out-of-context probabilities for POS and lemmas. In Section 6, we report on how well these probability features do in terms of disambiguation. CALIMA$_{Star}$ also produces tokenizations in different schemes from features that are pre-compiled in its database.

3 Baseline Arabic Morphology: Algorithm and Database

In this section, we describe the basic database and algorithm used in a number of morphological analyzers, all following the work of Buckwalter (2002), including our system CALIMA$_{Star}$.

3.1 Database Structure

The database we use has six tables. Three are lexicon tables for prefixes, suffixes, and stems. Each lexicon table entry has three columns: a lookup form, a compatibility category to control behavior and agreement, and a list of feature-value pairs. The lookup form is an orthographically normalized surface form, which can appear with multiple categories and feature-value pairs. The other three tables are compatibility tables: prefix-suffix, prefix-stem, and stem-suffix. These tables are used to ensure that any analysis that is produced by the system contains a prefix, a stem, and a suffix that are compatible with each other. This means, for example, that a nominal prefix will not be combined with a verbal stem. The compatibility tables are based on the compatibility categories appearing in the lexicon tables.

The differences among the many analyzers based on this model are mostly in features. BAMA and SAMA have four features: the diacritized surface form, the lemma, the Buckwalter POS tag, and the English gloss (Buckwalter, 2002; Graff et al., 2009). ALMOR automatically extends these features with 17 others: MADA POS tag (Pasha et al., 2014), four possible proclitics, person, aspect, voice, mood, gender, number, state, case, enclitic, rationality, stem, and stem category. The ALMOR database also includes feature definitions specifying the values each feature can take, and default feature values for every POS tag. CALIMA$_{Star}$ extends on the ALMOR database feature list as will be discussed in Section 4.

3.2 Analysis Algorithm

The analyzer follows the algorithm description in Buckwalter (2002). The input of the algorithm is a word. The output is all the possible out-of-context analyses for this word. We match the input word using an orthographically normalized form of it that is consistent with the look up forms in the database. Orthographic normalization is necessary because, according to Habash (2010), the most common spelling errors in Arabic involve Hamzated Alifs and Alif-Maqsura/Ya confusion, affecting 11% of all words (or 4.5 errors per sentence) in the PATB. We normalize by removing diacritics, converting Hamzated Alif occurrences أ $\hat{A}$, إ $\check{A}$, آ $\bar{A}$,[3] to bare Alif ا A, Alif-Maqsura ى $\acute{y}$ to Ya ي y, and Ta-Marbuta ة $\hbar$ to Ha ه h.

The word is then segmented into all possible prefix-stem-suffix triplets. The segment validity is restricted by the minimum length of stem and the maximum lengths of the prefix and suffix which are inferred from the database, in addition to the

[3] Arabic transliteration is presented in the Habash-Soudi-Buckwalter scheme (Habash et al., 2007).

existence of all segments in their respective lexicon tables. Each segmentation triplet is tested for compatibility using the three-way compatibility tables in the database. For each valid triplet combination, the features from the prefix, stem and suffix are merged to produce a single feature-set for the word. The merging process involves four operations depending on the feature: (i) concatenation of the Buckwalter POS tag; (ii) concatenation and rewriting of the final diacritized form; (iii) value overwrite for the remaining features, where, first, the suffix features overwrite the stem features, and then the prefix features overwrite all features; and (iv) producing the *source* feature by the analyzer depending on whether the analysis is from the lexicon, a backoff, or a default analysis of punctuation, digits, or foreign words. The result of this process is a unique set of out-of-context analyses for the input word. The first two operations above are used in BAMA and SAMA; and all are used in ALMOR. When no valid analysis is found, a backoff solution suspends the requirement for having a valid stem from the stem table. However, the prefixes and suffixes must still be compatible. We discuss the CALIMA$_{Star}$ algorithm extensions beyond ALMOR in Section 5.

3.3 Generation Algorithm

The generation algorithm follows the description of the generation component in ALMOR (Habash, 2007). It minimally expects a lemma and POS as input. The other features are handled in one of two ways. All inflectional features, such as person, gender and number, are considered obligatory, and as such, all their values are generated if no value was specified. Clitics, on the other hand, are considered optional, and are only generated when specified.

For the input lemma and POS, we retrieve all stems in the database. For each stem, the stem categories are then used to retrieve all stem-compatible prefixes and suffixes. Only compatible prefixes and suffixes (as per the prefix-suffix compatibility tables) are used, with the stem, to generate inflected words and corresponding full analyses. The same merging process used in the analysis component is used here also. The input feature-list is used in filtering which prefixes and suffixes to consider. For clitics, this is done before merging, but for inflectional features, this is done after merging.

4 CALIMA$_{Star}$ Database

In this section we detail our specific CALIMA$_{Star}$ database extensions to the basic ALMOR database structure presented in Section 3.

4.1 Gender and Number Functional Features

Smrž (2007) and Alkuhlani and Habash (2011) pointed out the common discrepancy between the form of some Arabic words and their function. A prime example is the very common *broken plural*[4] — almost 55% of all plurals look like singular words, but are functionally plural (Alkuhlani and Habash, 2011). For example, شرائح *šrAŷH* ‘sections, slices, slivers’ is the broken plural of the feminine singular word شريحة *šryHħ* ‘section, slice, sliver’. ALMOR considers this (functionally feminine and plural) noun, masculine and singular because it has the form of a masculine singular noun. Alkuhlani and Habash (2011) modeled functional gender and number on a portion of the PATB, which is based on BAMA/SAMA. Smrž (2007) and Boudlal et al. (2010) modeled functional gender and number for part of their data.

Our contribution in CALIMA$_{Star}$ is that we extended all lexical databases, which are based on SAMA and ALMOR, with functional gender and number. We built on the work and guidelines by Alkuhlani and Habash (2011). While Alkuhlani and Habash (2011) only annotated the inflected words that appear in the PATB, we wanted to annotate our entire database. To do this properly, we inflected the 31,610 nominal lemmas[5] in the database for gender and number, making sure that each database stem is used at least once. Our dataset contained 77,023 inflected words, which were manually annotated by two annotators for functional gender and number. This effort took 130 hours to complete.

In our database, we renamed ALMOR’s *gen* and *num* features as *form_gen* and *form_num*, and used *gen* and *num* as the names for the functional gender and number features, respectively. To operationalize the use of the functional gender and number features in the database, we assigned them the value **-** when the corresponding functional feature matches the form feature for all inflected forms of the stem. This value is then overwritten by the

[4]The ‘broken’ part of the name refers to the change in the template associated with forming these plurals.

[5]Verbs are regular and have no discrepancy between form and function gender and number.

form feature value coming from the suffix when producing an analysis. If the functional features does not match the form feature, then the functional feature is assigned an explicit value in the database, which can be *masculine* (*m*), or *feminine* (*f*) for gender, and *singular* (*s*), *plural* (*p*), or *dual* (*d*) for number. These explicit values in the stem are not overwritten by the suffix corresponding feature values.

4.2 State and Case Features

We follow the functional classification of the Arabic *state* feature into definite, indefinite, or construct values as described and implemented by Smrž (2007). Our contribution here is applying this classification to the latest SAMA/ALMOR database which we use in CALIMA$_{Star}$.

The ALMOR handling of state has two problems. First, the definite state is only assigned by the definite article proclitic +ال *Al+*. This is linguistically incorrect, as there are nouns that can be definite without the definite article (e.g., directly addressed vocatives such as يا أولاد ادرسوا *yA ÂwlAd AdrswA* 'O children, study'); and the definite article can appear with construct state adjectives in what is called *false idafa* (e.g., الطويل القامة *AlTwyl AlqAmħ* 'The tall of stature'). Second, a number of nominal suffixes are ambiguous but that ambiguity is not represented. For example, the suffix *+u* indicating nominative case can appear with both definite and construct states, but ALMOR only assigns it the construct value. In CALIMA$_{Star}$ we address both of these issues: (i) we do not allow the definite article to assign a state value and relegate state specification to the suffixes completely; and (ii) we ensure all ambiguous suffix lexicon entries are duplicated and assigned explicit state analyses.

Case is properly handled for the most part in ALMOR, except for a few cases where it interacts with state, such as diptotes. Diptotes are nominals whose genitive case marker is the same as their accusative case marker when they are in an indefinite state (Habash, 2010). An example of a diptote is شرائح *šrAŷH* 'sections, slices, slivers'. When it is indefinite, the genitive diacritized form of this word is شَرَائِحَ *šarAŷiHa*. The state feature extension that produces a definite analysis for the word without a definite article morpheme required an extension of the case feature to produce the definite genitive diacritized form شَرَائِحِ *šarAŷiHi*.

4.3 Rationality

Rationality (or +Human) is a lexical feature in Arabic that affects noun-adjective and subject-verb agreement. Nouns that are exclusive to humans, such as موظف *mwĎf* 'employee' are considered rational, while others, such as شريحة *šryHħ* 'slice' are considered irrational. Rational nouns take adjectives and verbs that agree with them in gender and number. Irrational nouns, on the other hand, agree with their verbs and adjectives when they are singular, however they take the singular feminine form of a verb or adjective when they are plural, regardless of what the gender of the noun is. The plural word شرائح *šrAŷH* 'slices' is irrational and as such would take the adjective رفيعة *rfyςħ* 'thin [feminine singular]' instead of رفيعات *rfyςAt* 'thin [feminine plural]'.

Alkuhlani and Habash (2011) manually annotated a part of PATB for rationality. ALMOR and MAGEAD include a rationality feature, but it was automatically populated and not fully checked. Our contribution in CALIMA$_{Star}$ is fully annotating the SAMA/ALMOR database for rationality. We build on the work of Alkuhlani and Habash (2011), and use their manual annotation guidelines. We manually annotated all the noun and proper noun lemmas in the database (18,120 entries). This effort took approximately 60 hours.

4.4 Roots and Patterns

Arabic templatic morphology, or the use of roots and patterns, has been modeled successfully in a number of systems (Beesley et al., 1989; Smrž, 2007). MAGEAD (Habash and Rambow, 2006; Altantawy et al., 2010) made extensive use of the ElixirFM database. However, roots and patterns are not part of the SAMA/ALMOR database that we base our work on.

Our contribution is to fully specify the roots and patterns in the CALIMA$_{Star}$ database. This manual effort started from the MAGEAD and ElixirFM lemma root information when available. The roots are linked to the lemmas directly and specified in the lexicon stem entries. The patterns we include are concrete patterns (Habash and Rambow, 2006) that are generated automatically by subtracting the root from the diacritized stem.

4.5 Phonological Representation

The phonological form has been shown to be particularly useful for NLP applications, such as Arabic speech recognition (Biadsy et al., 2009). In MSA, most Arabic letters have a one-to-one mapping to phonemes. However, there are some exceptions. The Arabic definite article +ال *Al+* assimilates to the first consonant in the nominal it modifies if this consonant is one of the so-called *Sun Letters* (Habash, 2010). For instance, the phonology of the noun الشَّرَائِح *AlšarAŷiHa* is / a sh sh a r aa 2 i 7 a /,[6] as opposed to */ a l sh a r aa 2 i 7 a /, because the letter ش *š* is a Sun Letter. Another exception is the silent Alif in the suffix +وا *+uwA*, which indicates a masculine plural conjugation in verbs. The phonology of the masculine plural verb كَتَبُوا *katabuwA* 'they wrote' is / k a t a b uu /, as opposed to */ k a t a b uu aa /.

This feature does not exist in SAMA or ALMOR. We extended our database entries with their respective phonological representation through an automatic process comparable to Biadsy et al. (2009). The definite article assimilation is handled through rewrite rules because it involves an interaction between a stem and a prefix.

4.6 Tokenization Schemes

Tokenization is important for NLP tasks such as machine translation because it reduces word sparsity (Habash and Sadat, 2006; Zalmout and Habash, 2017b). Many tokenization schemes with different granularities and normalization rules exist, and the selection of tokenization scheme depends on the task on hand.

In MADAMIRA, tokenization happens after analysis and disambiguation through an expensive regeneration process. Our contribution is the insight that since a particular tokenization is completely dependent on the analysis, we can specify the tokenization details in the database entry. This is a tradeoff of a bigger database (space) with a faster tokenization (time). We specifically add four tokenization schemes D1, D2, D3 and ATB (Habash, 2010), in both normalized and unnormalized forms. We refer to the normalized schemes as tokenization, and to the unnormalized schemes as segmentation. A normalized ATB tokenization for the word كتبوها *ktbwhA* 'they wrote it' is كتبوا+ها *ktbwA+hA* 'they_wrote +it', whereas an unnormalized ATB tokenization (segmentation) for the same word would be كتبو+ها *ktbw+hA*. In the lexicon, the entry for the suffix in this analysis has a different feature for each of these schemes.

We extended our database with the tokenization features in a semi-automated process. For each scheme, we manually determined the list of affixes that would be detached. For each of these affixes, we generated the tokenized form. Stems do not have any parts to detach, but some stems have to be normalized for some tokenization schemes. We used our CALIMA$_{Star}$ generator to automatically get the normalized forms of these stems. The resulting tokenizations of a word are the concatenation of the tokenizations of the prefix, stem, and suffix of the word's analysis.

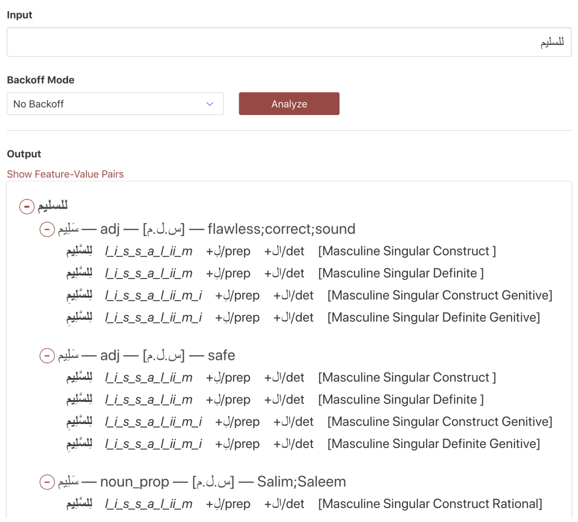
4.7 Multiple POS Tag Sets

There are many Arabic POS tag sets used by different researchers and in different tools, e.g., Buckwalter (Buckwalter, 2002), MADA (Pasha et al., 2014), Columbia Arabic Treebank (CATiB) (Habash and Roth, 2009), CATiBex (Marton et al., 2013), Universal Dependencies (UD) (Nivre et al., 2016), and Kulick (Kulick et al., 2006). It is desirable to link these POS tag sets to each other. CALIMA$_{Star}$ currently supports four POS tag sets: the Buckwalter POS tag set, and the MADA POS tag set, both of which are part of the ALMOR database, as well as the CATiB and UD POS tag sets. We chose to start our extension with the CATiB and UD POS tag sets for their importance to the work on Arabic dependency parsing. This goal steered us to output the CATiB and UD POS tags following the ATB tokenization, which is the commonly used tokenization format in treebanking and parsing. This extension was an automatic mapping from both the Buckwalter and MADA POS tag sets. We plan to add more POS tag sets to our database in the future.

4.8 Lemma and POS Probability

Inspired by Khalifa et al. (2016) who used lemma and POS tag probabilities for out-of-context disambiguation, we added three different probability scores for lemma, POS (MADA POS) and joint lemma-POS, for each stem entry in the database. The scores were generated from the *train* set (Diab et al., 2013) of the PATB. We used the SRILM toolkit (Stolcke, 2002) to generate the scores with no smoothing. In Section 6, we show that these scores can be used to select the correct POS tag and lemma out of context with a high accuracy.

[6]The phonological representation we use follows the CAMEL Arabic Phonetic Inventory (CAPHI) (Habash et al., 2018), which is inspired by Arpabet (Shoup, 1980).



Input: للسليم

Backoff Mode: No Backoff

Analyze

Output

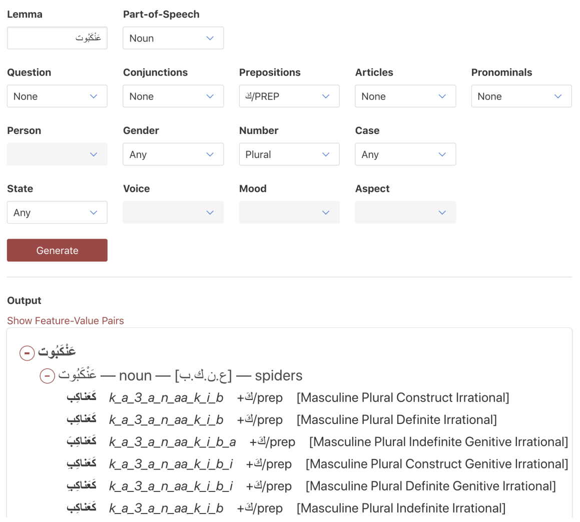
Show Feature-Value Pairs

```
للسليم
  سَلِيم — adj — [س.ل.م] — flawless;correct;sound
    لِلسَّلِيم   l_i_s_s_a_l_ii_m     +لِ/prep   +ال/det   [Masculine Singular Construct ]
    لِلسَّلِيم   l_i_s_s_a_l_ii_m     +لِ/prep   +ال/det   [Masculine Singular Definite ]
    لِلسَّلِيم   l_i_s_s_a_l_ii_m_i   +لِ/prep   +ال/det   [Masculine Singular Construct Genitive]
    لِلسَّلِيم   l_i_s_s_a_l_ii_m_i   +لِ/prep   +ال/det   [Masculine Singular Definite Genitive]

  سَلِيم — adj — [س.ل.م] — safe
    لِلسَّلِيم   l_i_s_s_a_l_ii_m     +لِ/prep   +ال/det   [Masculine Singular Construct ]
    لِلسَّلِيم   l_i_s_s_a_l_ii_m     +لِ/prep   +ال/det   [Masculine Singular Definite ]
    لِلسَّلِيم   l_i_s_s_a_l_ii_m_i   +لِ/prep   +ال/det   [Masculine Singular Construct Genitive]
    لِلسَّلِيم   l_i_s_s_a_l_ii_m_i   +لِ/prep   +ال/det   [Masculine Singular Definite Genitive]

  سَلِيم — noun_prop — [س.ل.م] — Salim;Saleem
    لِلسَّلِيم   l_i_s_s_a_l_ii_m     +لِ/prep   +ال/det   [Masculine Singular Construct Rational]
```

(a) The analysis output of the word للسليم *llslym* grouped by lemma and POS.



Lemma: عَنْكَبُوت

Part-of-Speech: Noun

Question	Conjunctions	Prepositions	Articles	Pronominals
None	None	كَ/PREP	None	None

Person	Gender	Number	Case
	Any	Plural	Any

State	Voice	Mood	Aspect
Any			

Generate

Output

Show Feature-Value Pairs

```
عَنْكَبُوت
  عَنْكَبُوت — noun — [ع.ن.ك.ب] — spiders
    كَعَناكِب   k_a_3_a_n_aa_k_i_b     +كَ/prep   [Masculine Plural Construct Irrational]
    كَعَناكِب   k_a_3_a_n_aa_k_i_b     +كَ/prep   [Masculine Plural Definite Irrational]
    كَعَناكِبَ   k_a_3_a_n_aa_k_i_b_a   +كَ/prep   [Masculine Plural Indefinite Genitive Irrational]
    كَعَناكِبِ   k_a_3_a_n_aa_k_i_b_i   +كَ/prep   [Masculine Plural Construct Genitive Irrational]
    كَعَناكِبِ   k_a_3_a_n_aa_k_i_b_i   +كَ/prep   [Masculine Plural Definite Genitive Irrational]
    كَعَناكِب   k_a_3_a_n_aa_k_i_b     +كَ/prep   [Masculine Plural Indefinite Irrational]
```

(b) The generation output for the nominal lemma عَنكَبُوت *ςanokabuwt* 'spider' as plural with the preposition كَ *ka* 'as'.

Figure 1: A screen capture of the CALIMA$_{Star}$ analyzer and generator interfaces.

5 CALIMA$_{Star}$ Engine

The CALIMA$_{Star}$ engine is a new implementation of the analysis and generation algorithms described in Section 3 with some extensions. It uses the database described in Section 4.

CALIMA$_{Star}$ API The CALIMA$_{Star}$ engine is implemented in Python. We provide a command-line tool interface as well as an API. CALIMA$_{Star}$ is a part of a collection of Arabic NLP tools we plan to release.

Analysis and Generation Extensions All of the algorithmic extensions in the CALIMA$_{Star}$ engine are minor, and intended to accommodate the additional features in the database. Examples include the special handling of functional gender and number as discussed in Section 4.1, the concatenation of prefix, stem and suffix features for added POS tags and tokenization schemes. The concatenation of the CAPHI string and the pattern requires rewrite rules because of prefix-stem interactions.

Reinflection Inspired by the SIGMORPHON 2016 Shared Task on morphological reinflection (Cotterell et al., 2016), we provide a reinflection functionality in the CALIMA$_{Star}$ API, which makes use of the existing analysis and generation components. The input to the CALIMA$_{Star}$ reinflector is an already inflected word and a desired set of feature-value changes. The system analyzes the word, adjusts its features given the input feature-value pairs, and generates the reinflected form(s). The reinflector is also used as a backoff mode for the generator when the input lemma is not recognized.

Web Interface We created a web interface for the CALIMA$_{Star}$ analyzer and generator – see Figures 1 (a) and (b), respectively. The analyzer interface expects an input word and a selected backoff mode. Options for backoff include no backoff, proper-noun backoff, or any POS backoff. The generator interface minimally expects a lemma and POS for input; feature values can be specified as needed. The generator interface changes which features are allowed to select values for depending on the input POS. For example, in Figure 1 (b), the verbal features, person, voice, mood and aspect, are disabled because the input POS is a noun. There are two output modes for both analyzer and generator interfaces. The first output mode is a user-friendly display of words grouped by lemma, POS tag, root and English gloss. For each inflected word, the interface shows its diacritization, phonology, clitics, and inflectional features, in a human-readable form. This output mode is what Figures 1 (a) and (b) show. The second mode presents the output in a feature-value pair format more suitable for debugging and programming interfaces.

The engine and web interface are linked from http://resources.camel-lab.com/.

6 Evaluation

In this section we validate our system and evaluate it against other systems in terms of speed and coverage.

6.1 Internal Validation

We ran a number of tests to validate our database extensions. All these tests were run on the PATB *dev* set (Diab et al., 2013) from PATB parts 1, 2 and 3, except where indicated.

Analysis This test aimed to validate that we are producing all the analyses that are produced by ALMOR. We ran the tokens in the dataset through ALMOR and CALIMA$_{Star}$, and verified that every analysis produced by ALMOR is also produced by CALIMA$_{Star}$. CALIMA$_{Star}$ produced more analyses on average than ALMOR. And naturally, the CALIMA$_{Star}$ analyses were richer than the ALMOR analyses.

Generation Since we do not have a manually annotated version of this data set with all of our extensions, we relied on automatic matching of CALIMA$_{Star}$ analyses against the ALMOR analyses used to train and evaluate MADAMIRA (Pasha et al., 2014).[7] This matching allowed us to extend the ALMOR analyses with functional gender and number features. We used the lemma and extended features as input to the CALIMA$_{Star}$ generation component. CALIMA$_{Star}$ produced the full diacritized word in all cases.

Reinflection Validating the reinflection component required us to have a source inflected word, and a target inflected word. We grouped the extended ALMOR analyses from the Generation test above by lemma and POS tag, and generated all possible pairs of words that share a lemma and POS tag. For each pair, we used the features of the first word to reinflect the second word, and vice versa. Our system produced the correct diacritized word in all cases.

Functional Gender and Number Analysis This test aimed to validate that the database extension for functional gender and number was consistent with the manual annotation. For this test, we analyzed every word that was manually annotated for functional gender and number, and confirmed that the analyzer is producing an analysis with the expected functional feature values. Our system produced the expected values in all cases.

Tokenization We tested our D3 tokenization extensions, as it is the most complex tokenization scheme in our database. We ran our test set in MADAMIRA to produce the D3 tokenization. We then compared the tokenization produced by MADAMIRA to the tokenization produced by CALIMA$_{Star}$ for the analysis that is equivalent to MADAMIRA's top analysis (modulo our extensions). We matched MADAMIRA's diacritized tokenization in 99.5% of the cases, and we matched the undiacritized tokenization in 99.9% of the cases. The only undiacritized mismatches are the result of MADAMIRA tokenization errors in words such as بحيث *bHyθ* 'with/by + where; whereby', which MADAMIRA tokenizes as بحيث +ب *b+ bHyθ* instead of حيث +ب *b+ Hyθ*. This validation test also brought to light some minor cases which we intend to fix in future releases of the database.

Lemma and POS Probability We carried out preliminary experiments on the use of the different probability scores (Section 4.8) for out-of-context POS and lemma selection. We found that the top-one choice among the CALIMA$_{Star}$ analyses ranked using the joint lemma-POS score preformed the highest with 92%, 90% and 88% accuracy in terms of POS, lemma, and POS+lemma, respectively. The results are comparable to the maximum likelihood disambiguation baseline reported by Zalmout and Habash (2017a), which supports the use of morphological analyzers as a backbone for the different NLP task.

6.2 Comparison with other Systems

We compare CALIMA$_{Star}$ to four other morphological analyzers: AraMorph,[8] SAMA 3.1 (Graff et al., 2009), ALMOR and MADAMIRA,[9] in terms of coverage (out-of-vocabulary (OOV) rate and analyses per word) and speed performance (words, analyses, and features per second). We ran the experiments on one million words from the Arabic Gigaword corpus (Parker et al., 2011). The results of the comparison are in Table 2.

[7]The ALMOR analyses were themselves automatically matched against the gold PATB annotations in a similar manner to Habash and Rambow (2005).

[8]We used the AraMorph 1.2.1 version, and implementation of BAMA 1.2 that was optimized by Jon Dehdari from SourceForge `https://sourceforge.net/projects/aramorph/files/aramorph/1.2.1/`.

[9]We ran MADAMIRA in 'analyze only' mode.

System			Coverage		Speed		
Engine	**Database**	**Features #**	**OOV%**	$\frac{Analysis}{Word}$	$\frac{Word}{Second}$	$\frac{Analysis}{Second}$	$\frac{Feature}{Second}$
AraMorph	AraMorph	4	1.7	2.1	43.5K	93K	372K
SAMA	SAMA	4	1.4	10.2	2.3K	24K	96K
ALMOR	ALMOR	22	1.4	10.7	1.1K	12K	269K
MADAMIRA	ALMOR	22	1.6	10.7	6.8K	73K	1,742K
CALIMA$_{Star}$	ALMOR	22	1.3	10.7	8.2K	88K	1,938K
CALIMA$_{Star}$	CALIMA$_{Star}$	40	1.3	18.9	5.4K	102K	4,094K

Table 2: A comparison of five systems, AraMorph, SAMA, ALMOR, MADAMIRA and CALIMA$_{Star}$, in terms of coverage (OOV and analyses per word) and speed (word, analyses, features per second).

The first three columns of Table 2 specify the systems, the databases, and the number of features in the databases. AraMorph was run on the AraMorph database, and SAMA was run on the SAMA database. Both of these contain four features: the diacritization, Buckwalter POS tag, lemma, and English gloss. ALMOR and MADAMIRA both use the ALMOR database, which produces 22 features: the same four features of AraMorph/SAMA in addition to the MADA POS tag, four proclitic features, person, aspect, voice, mood, gender, number, state, case, enclitic, rationality, source, stem, and stem category. CALIMA$_{Star}$ was run on the CALIMA$_{Star}$ database described in Section 4, which has 40 features: the 22 ALMOR features in addition to functional gender and number, CAPHI, root, pattern, tokenization and segmentation in four schemes (D1, D2, D3, and ATB), CATiB and UD POS tags, and POS, lemma, and joint lemma-POS probability scores. We also ran the CALIMA$_{Star}$ engine with the ALMOR database as a comparison point.

Coverage The OOV rates of the different analyzers are generally close to each other. The differences stem from two sources. First, AraMorph's database is an older version of SAMA/ALMOR/CALIMA$_{Star}$ with less lemmas ($\sim$38K vs $\sim$40K lemmas). And secondly, different engines handle numbers, foreign words, and digits in different ways, with the CALIMA$_{Star}$ engine outperforming all others.

The number of analyses per word is strongly connected to the database being used. AraMorph has less coverage, in terms of affixes and stems, and that is why it produces a smaller number of analyses per word. ALMOR and MADAMIRA both use the same database, thus producing the same number of analyses per word. CALIMA$_{Star}$ extends the suffix lexicon to fully cover the state and case analyses resulting in the largest number of analyses per word.

Speed The last three columns of Table 2 compare the speed of the systems, in terms of words per second, analyses per second, and features per second. In terms of words per second, AraMorph is the fastest system. Comparing the ALMOR, MADAMIRA and CALIMA$_{Star}$ engines using the same database (ALMOR), CALIMA$_{Star}$ is the fastest of the three. However, using the larger CALIMA$_{Star}$ database slows the CALIMA$_{Star}$ engine down to second place in this three-way comparison. That said, CALIMA$_{Star}$ produces more analyses per seconds and features per second than all the other systems.

7 Conclusion and Future Work

CALIMA$_{Star}$ is an Arabic analyzer and generator that supports a large number of morphological features. It has as an API and a web interface, and will be released publicly. Compared with four commonly used Arabic analyzers, CALIMA$_{Star}$ has better coverage and is very competitive speedwise.

In the future, we will extend our database in terms of lexical entries, features, and dialects. We will also integrate our system into existing disambiguators and parsers (Pasha et al., 2014; Shahrour et al., 2016; Zalmout and Habash, 2017a). Finally, we plan on conducting task-based evaluations where we can assess the added value of some of the new features.

Acknowledgments We thank the annotators who helped with gender, number and rationality features: Linda Alamir-Salloum and Sara Hassan.

References

Ahmed Abdelali, Kareem Darwish, Nadir Durrani, and Hamdy Mubarak. 2016. Farasa: A fast and furious segmenter for Arabic. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 11–16, San Diego, California.

Imad A. Al-Sughaiyer and Ibrahim A. Al-Kharashi. 2004. Arabic morphological analysis techniques: A comprehensive survey. *Journal of the American Society for Information Science and Technology*, 55(3):189–213.

Sarah Alkuhlani and Nizar Habash. 2011. A Corpus for Modeling Morpho-Syntactic Agreement in Arabic: Gender, Number and Rationality. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL'11)*, Portland, Oregon, USA.

Mohamed Altantawy, Nizar Habash, and Owen Rambow. 2011. Fast Yet Rich Morphological Analysis. In *Proceedings of the 9th International Workshop on Finite-State Methods and Natural Language Processing (FSMNLP 2011)*, Blois, France.

Mohamed Altantawy, Nizar Habash, Owen Rambow, and Ibrahim Saleh. 2010. Morphological Analysis and Generation of Arabic Nouns: A Morphemic Functional Approach. In *Proceedings of the seventh International Conference on Language Resources and Evaluation (LREC)*, Valletta, Malta.

Kenneth Beesley, Tim Buckwalter, and Stuart Newton. 1989. Two-Level Finite-State Analysis of Arabic Morphology. In *Proceedings of the Seminar on Bilingual Computing in Arabic and English*.

Kenneth R. Beesley. 1996. Arabic finite-state morphological analysis and generation. In *Proceedings of COLING-96, the 16th International Conference on Computational Linguistics*, Copenhagen.

Fadi Biadsy, Nizar Habash, and Julia Hirschberg. 2009. Improving the Arabic Pronunciation Dictionary for Phone and Word Recognition with Linguistically-Based Pronunciation Rules. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 397–405, Boulder, Colorado.

Mohamed Boudchiche, Azzeddine Mazroui, Mohamed Ould Abdallahi Ould Bebah, Abdelhak Lakhouaja, and Abderrahim Boudlal. 2017. AlKhalil Morpho Sys 2: A robust Arabic morpho-syntactic analyzer. *Journal of King Saud University-Computer and Information Sciences*, 29(2):141–146.

Abderrahim Boudlal, Abdelhak Lakhouaja, Azzeddine Mazroui, Abdelouafi Meziane, MOAO Bebah, and M Shoul. 2010. Alkhalil Morpho Sys1: A morphosyntactic analysis system for Arabic texts. In *International Arab conference on information technology*, pages 1–6. Benghazi Libya.

Tim Buckwalter. 2002. Buckwalter Arabic Morphological Analyzer Version 1.0. Linguistic Data Consortium, University of Pennsylvania, 2002. LDC Catalog No.: LDC2002L49.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, David Yarowsky, Jason Eisner, and Mans Hulden. 2016. The SIGMORPHON 2016 shared task on morphological reinflection. In *Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 10–22.

Mona Diab, Nizar Habash, Owen Rambow, and Ryan Roth. 2013. LDC Arabic treebanks and associated corpora: Data divisions manual. *arXiv preprint arXiv:1309.5652*.

Mona Diab, Kadri Hacioglu, and Daniel Jurafsky. 2004. Automatic Tagging of Arabic Text: From Raw Text to Base Phrase Chunks. In *Proceedings of the 5th Meeting of the North American Chapter of the Association for Computational Linguistics/Human Language Technologies Conference (HLT-NAACL04)*, pages 149–152, Boston, MA.

David Graff, Mohamed Maamouri, Basma Bouziri, Sondos Krouna, Seth Kulick, and Tim Buckwalter. 2009. Standard Arabic Morphological Analyzer (SAMA) Version 3.1. Linguistic Data Consortium LDC2009E73.

Nizar Habash. 2007. Arabic Morphological Representations for Machine Translation. In A. van den Bosch and A. Soudi, editors, *Arabic Computational Morphology: Knowledge-based and Empirical Methods*. Springer.

Nizar Habash. 2010. *Introduction to Arabic Natural Language Processing*. Morgan & Claypool Publishers.

Nizar Habash, Fadhl Eryani, Salam Khalifa, Owen Rambow, Dana Abdulrahim, Alexander Erdmann, Reem Faraj, Wajdi Zaghouani, Houda Bouamor, Nasser Zalmout, et al. 2018. Unified guidelines and resources for Arabic dialect orthography. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*.

Nizar Habash, Ramy Eskander, and Adbelati Hawwari. 2012. A Morphological Analyzer for Egyptian Arabic. In *NAACL-HLT 2012 Workshop on Computational Morphology and Phonology (SIGMORPHON2012)*, pages 1–9.

Nizar Habash and Owen Rambow. 2005. Arabic tokenization, part-of-speech tagging and morphological disambiguation in one fell swoop. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 573–580, Ann Arbor, Michigan. Association for Computational Linguistics.

Nizar Habash and Owen Rambow. 2006. MAGEAD: A Morphological Analyzer and Generator for the Arabic Dialects. In *Proceedings of ACL*, pages 681–688, Sydney, Australia. Association for Computational Linguistics.

Nizar Habash, Owen Rambow, and Ryan Roth. 2009. MADA+TOKAN: A toolkit for Arabic tokenization, diacritization, morphological disambiguation, POS tagging, stemming and lemmatization. In *Proceedings of the Second International Conference on Arabic Language Resources and Tools*. The MEDAR Consortium.

Nizar Habash and Ryan M Roth. 2009. CATiB: The Columbia Arabic Treebank. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 221–224.

Nizar Habash and Fatiha Sadat. 2006. Arabic Preprocessing Schemes for Statistical Machine Translation. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 49–52, New York City, USA.

Nizar Habash, Abdelhadi Soudi, and Tim Buckwalter. 2007. On Arabic Transliteration. In A. van den Bosch and A. Soudi, editors, *Arabic Computational Morphology: Knowledge-based and Empirical Methods*. Springer.

Salam Khalifa, Sara Hassan, and Nizar Habash. 2017. A morphological analyzer for Gulf Arabic verbs. In *Proceedings of the Workshop for Arabic Natural Language Processing 2017 (co-located with EACL 2017)*, Valencia, Spain.

Salam Khalifa, Nasser Zalmout, and Nizar Habash. 2016. YAMAMA: Yet Another Multi-Dialect Arabic Morphological Analyzer. In *Proceedings of the International Conference on Computational Linguistics (COLING): System Demonstrations*, pages 223–227.

Seth Kulick, Ryan Gabbard, and Mitch Marcus. 2006. Parsing the Arabic Treebank: Analysis and Improvements. In *Proceedings of the 5th Conference on Treebanks and Linguistics Theories*, pages 31–32.

Mohamed Maamouri, Ann Bies, Tim Buckwalter, and Wigdan Mekki. 2004. The Penn Arabic Treebank: Building a Large-Scale Annotated Arabic Corpus. In *NEMLAR Conference on Arabic Language Resources and Tools*, pages 102–109, Cairo, Egypt.

Yuval Marton, Nizar Habash, and Owen Rambow. 2013. Dependency parsing of Modern Standard Arabic with lexical and inflectional features. *Computational Linguistics*, 39(1):161–194.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Portorož, Slovenia.

Robert Parker, David Graff, Ke Chen, Junbo Kong, and Kazuaki Maeda. 2011. Arabic Gigaword Fifth Edition. LDC catalog number No. LDC2011T11, ISBN 1-58563-595-2.

Arfath Pasha, Mohamed Al-Badrashiny, Mona Diab, Ahmed El Kholy, Ramy Eskander, Nizar Habash, Manoj Pooleery, Owen Rambow, and Ryan M Roth. 2014. MADAMIRA: A Fast, Comprehensive Tool for Morphological Analysis and Disambiguation of Arabic. In *Proceedings of the Language Resources and Evaluation Conference (LREC), Reykjavik, Iceland*.

Anas Shahrour, Salam Khalifa, Dima Taji, and Nizar Habash. 2016. CamelParser: A system for Arabic syntactic analysis and morphological disambiguation. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations*, pages 228–232.

June E Shoup. 1980. Phonological aspects of speech recognition. *Trends in speech recognition*, pages 125–138.

Otakar Smrž. 2007. ElixirFM — Implementation of Functional Arabic Morphology. In *Proceedings of the 2007 Workshop on Computational Approaches to Semitic Languages: Common Issues and Resources*, pages 1–8, Prague, Czech Republic.

Andreas Stolcke. 2002. SRILM an Extensible Language Modeling Toolkit. In *Proceedings of the International Conference on Spoken Language Processing*.

Michelle Vanni and Rémi Zajac. 1996. The Temple Translator's Workstation Project. In *Proceedings of a workshop on held at Vienna, Virginia: May 6-8, 1996*, pages 101–106.

Nasser Zalmout and Nizar Habash. 2017a. Don't throw those morphological analyzers away just yet: Neural morphological disambiguation for Arabic. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 704–713, Copenhagen, Denmark.

Nasser Zalmout and Nizar Habash. 2017b. Optimizing tokenization choice for machine translation across multiple target languages. *The Prague Bulletin of Mathematical Linguistics*, 108(1):257–269.