Morphotactic Modeling in an Open-source Multi-dialectal Arabic Morphological Analyzer and Generator

Nizar Habash,[†] **Reham Marzouk,**[†,††] **Christian Khairallah,**[†] **Salam Khalifa**[†,‡]
Computational Approaches to Modeling Language (CAMeL) Lab
[†]New York University Abu Dhabi
[††]Alexandria University, [‡]Stony Brook University
nizar.habash@nyu.edu, igsr.r.marzouk@alexu.edu.eg
christian.khairallah@nyu.edu, salam.khalifa@stonybrook.edu

Abstract

Arabic is a morphologically rich and complex language, with numerous dialectal variants. Previous efforts on Arabic morphology modeling focused on specific variants and specific domains using a range of techniques with different degrees of linguistic modeling transparency. In this paper we propose a new approach to modeling Arabic morphology with an eye towards multi-dialectness, resource openness, and easy extensibility and use. We demonstrate our approach by modeling verbs from Standard Arabic and Egyptian Arabic, within a common framework, and with high coverage.

1 Introduction

There has been a lot of work on Arabic computational morphology in the last three decades (Beesley et al., 1989; Kiraz, 1994; Al-Sughaiyer and Al-Kharashi, 2004; Graff et al., 2009; Boudchiche et al., 2017; Taji et al., 2018). These efforts were motivated by Arabic's many challenges, namely, its morphological richness and complexity, its orthographic ambiguity and noise, and its numerous dialectal variants. The work on Arabic computational morphology has led to the development of many resources that directly model morphology (e.g., analyzers, generators) and also resources and tools that use them (Maamouri et al., 2004; Pasha et al., 2014). Morphological analyzers have consistently shown that they are still valuable components in the NLP toolbox, even as the latter increasingly shifts toward the neural modeling space, and especially in low-resource and dialectal settings (Zalmout and Habash, 2017; Baly et al., 2017; Inoue et al., 2022).

The range of techniques explored for morphological modeling has been quite large, from finite-state machines to procedural and functional programming languages, covering different degrees of depth in different linguistic representations, different variants, and different domains and genres. However, a common challenge among these approaches is the inconsistent coverage of different linguistic features. For example, the Standard Arabic Morphological Analyzer (SAMA, v3.1) (Graff et al., 2009), which was developed in conjunction with work on Modern Standard Arabic (MSA) newswire text in the Penn Arabic Treebank (PATB) (Maamouri et al., 2004), has only 65 imperative verb forms, while it has over 13 thousand perfective verb forms. SAMA also has only 15 instances of the interrogative proclitic أ *Âa*[1] which in principle can attach to any word. Another example is the Calima-ARZ system for Egyptian Arabic (EGY) (Habash et al., 2012), which used automatically generated stem classes making it very hard to linguistically generalize and extend. Many of the Arabic morphology resources are not freely available, easy to augment, or ready to plug-and-play in open-source public libraries.

The work presented in this paper is part of a larger effort on the CAMELMORPH Project.[2] CAMELMORPH's goal is to build large open-source morphological models for Arabic and its dialects across many genres and domains. The focus in this paper is on the core components that define lexical and morphological information and the tools to convert them into models that are readily usable within an existing Python open-source suite for Arabic NLP, Camel Tools (Obeid et al., 2020). We demonstrate the effectiveness of our approach by modeling MSA and EGY verbs using a shared representation, and showing improved coverage compared to publicly available analyzers. Our data and code are publicly available.[2]

Next we present some related work (§2), a discussion of Arabic linguistic background (§3), and our approach (§4). We then present the MSA and EGY verbal models (§5) and evaluate them (§6).

[1]HSB Arabic transliteration (Habash et al., 2007).
[2]http://morph.camel-lab.com

2 Related Work

There has been a considerable amount of work on Arabic morphological analysis (Al-Sughaiyer and Al-Kharashi, 2004; Habash, 2010). Altantawy et al. (2011) organized the various Arabic morphology processing efforts along a continuum of approaches that is characterized by two poles: on one end, very abstract and linguistically rich representations and rules are used to derive surface forms; while on the other end, simple and shallow techniques focus on efficient search in a space of pre-compiled (tabulated) solutions. The first type is typically but not strictly implemented using finite-state technology, and was one of the earliest efforts undertaken (Beesley et al., 1989; Kiraz, 1994; Beesley, 1996; Habash and Rambow, 2006; Smrž, 2007). These models can be rather complex and have many internal dependencies among the rules used for modeling sub-word structure and morphotactic and orthographic forms. The second type is typically not implemented in finite-state technology. Examples include the Buckwalter Arabic Morphological Analyzer (BAMA) (Buckwalter, 2004) and extensions of it (Graff et al., 2009; Taji et al., 2018). These systems do not represent the morphemic, phonological and orthographic rules directly, and instead compile their effect into the lexicon itself. Hulden and Samih (2012) demonstrated a method of mapping from the pre-compiled tabulated approaches to finite-state representation; and Altantawy et al. (2011) demonstrated the reverse process of going from finite-state to the tabulated representation.

In this paper we present an approach that is a middle ground between these two poles. In lieu of generative solutions employing rewrite rules to map from underlying forms (morphemes) to surface forms (allomorphs), we enumerate, in a limited pre-compiled manner, the various allomorphic forms, and indicate the different context conditions that select for their realization. Our morphological specifications also include information about how to order these different morphemes. Then, in an offline process, we convert our morphological specifications into a full pre-compiled tabulated format in the style of BAMA (Buckwalter, 2004) and CALIMA$_{Star}$ (Taji et al., 2018) databases (DBs) used in the open-source Python toolkit Camel Tools (Obeid et al., 2020). Camel Tools's morphological engines enable the use of the same morphological DB for analysis, generation, and reinflection.

Our approach is closest to Hockett (1954)'s Item-and-Arrangement approach, linguistically speaking; however, we do make use of post-processing transformations (a la Item-and-Process) in a limited way for phonological and orthographic phenomena that do not change the basic letter spelling of the Arabic word, but can change its diacritics. Also, to maximize the utility of our models, we use lemmas and features that allow us to relate our output to Word and Paradigm approaches (Bram, 2012).

Finally, while we do not explicitly rely on roots and patterns to derive our forms, as was done by Beesley (1996), and Habash and Rambow (2006), we plan, in future efforts, to abstract from existing entries templatic patterns that allow us to back off intelligently to unseen words if needed.

3 Arabic Linguistic Background

3.1 General Challenges

Arabic orthography, morphology, and dialectal variation pose a number of challenges for NLP.

Orthographic Ambiguity Arabic is typically written without the optional diacritical marks that are used for short vowels and consonantal gemination, leading to a high degree of ambiguity. MSA has upwards of 12 analyses per word (Pasha et al., 2014). A subtask of morphological analysis is producing the correct diacritization for each analysis.

Morphological Richness Arabic inflects for gender, number, person, aspect, mood, case, state and voice. In addition, Arabic orthography cliticizes a number of pronouns (direct object, possessive) and particles (conjunctions, prepositions, definite article, etc.). This results in thousands of forms for each verbal lemma. Because of orthographic ambiguity, words with analyses that differ in the presence of clitics are not uncommon, e.g., وحد *wHd* can be analyzed as *wa+Had∼a* 'and he limited' or *waH∼ada* 'he united', among other readings.

Morphological Complexity Arabic uses a combination of templatic morphemes (roots and patterns) and concatenative affixes and clitics. There are also many complex morphotactic rewriting operations that cause these morphemes to surface in different ways (allomorphs) in different contexts. We present a more detailed set of examples in Section 3.2 to motivate the approach in this paper.

Dialectal Variation In addition to MSA, the de facto official language in the Arab World, there is a number of different local dialects (e.g., Egyptian,

	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)
	Modern Standard Arabic (MSA)								
(1)	**Root + Pattern**	**Lemma**	**Suff.P3MS**	**Suff.P3FS**	**Suff.P3MP**	**Suff.P2MS**	**Suff.P2FS**	**Suff.P2MP**	**Pron.3MS**
(2)			**+a**	**+at**	**+uwA**	**+ta**	**+ti**	**+tum**	**+hu**
(3)	**k.t.b + 1a2a3**	**katab**	katab+a	katab+at	katab+uwA	katab+ta	katab+ti	katab+tum	
(4)		*write*	katab+a+hu	katab+at+hu	katab+**uw**+hu	katab+ta+hu	katab+ti+**hi**	katab+**tumuw**+hu	✓
(5)	**n.H.t + 1a2a3**	**naHat**	naHat+a	naHat+at	naHat+uwA	naHat+**~a**	naHat+**~i**	naHat+**~um**	
(6)		*sculpt*	naHat+a+hu	naHat+at+hu	naHat+**uw**+hu	naHat+**~a**+hu	naHat+**~i**+**hi**	naHat+**~umuw**+hu	✓
(7)	**r.n.n + 1a2a3**	**ran~**	**ran~**+a	**ran~**+at	**ran~**+uwA	ranan+ta	ranan+ti	ranan+tum	
(8)		*ring*	**ran~**+a+hu	**ran~**+at+hu	**ran~**+**uw**+hu	ranan+ta+hu	ranan+ti+**hi**	ranan+**tumuw**+hu	✓
(9)	**r.m.y + 1a2a3**	**ramaY**	**ramaY**	**ram**+at	**ram**+**awA**	ramay+ta	ramay+ti	ramay+tum	
(10)		*throw*	**ramA**+hu	**ram**+at+hu	**ram**+**aw**+hu	ramay+ta+hu	ramay+ti+**hi**	ramay+**tumuw**+hu	✓
(11)	**k.t.b + 1A2a3**	**kAtab**	kAtab+a	kAtab+at	kAtab+uwA	kAtab+ta	kAtab+ti	kAtab+tum	
(12)		*correspond with*	kAtab+a+hu	kAtab+at+hu	kAtab+**uw**+hu	kAtab+ta+hu	kAtab+ti+**hi**	kAtab+**tumuw**+hu	✓
	Egyptian Arabic (EGY)								
(13)	**Root + Pattern**	**Lemma**	**Suff.P3MS**	**Suff.P3FS**	**Suff.P3MP**	**Suff.P2MS**	**Suff.P2FS**	**Suff.P2MP**	**Pron.3MS**
(14)			**+**	**+it**	**+uwA**	**+t**	**+tiy**	**+tuwA**	**+uh**
(15)	**k.t.b + 1a2a3**	**katab**	katab	katab+it	katab+uwA	katab+t	katab+tiy	katab+tuwA	
(16)		*write*	katab+uh	katab+it+uh	katab+**uw**+**h**	katab+t+uh	katab+tiy+**h**	katab+**tuw**+**h**	✓
(17)	**n.H.t + 1a2a3**	**naHat**	naHat	naHat+it	naHat+uwA	naHat+**~**	naHat+**~iy**	naHat+**~uwA**	
(18)		*sculpt*	naHat+uh	naHat+it+uh	naHat+**uw**+**h**	naHat+**~**+uh	naHat+**~iy**+**h**	naHat+**~uw**+**h**	✓
(19)	**r.n.n + 1a2a3**	**ran~**	**ran~**	**ran~**+it	**ran~**+uwA	**ran~ay**+t	**ran~ay**+tiy	**ran~ay**+tuwA	
(20)		*ring*	**ran~**+uh	**ran~**+it+uh	**ran~**+**uw**+**h**	**ran~ay**+t+uh	**ran~ay**+tiy+**h**	**ran~ay**+**tuw**+**h**	✓
(21)	**r.m.y + 1a2a3**	**ramaY**	**ramaY**	**ram**+it	**ram**+uwA	ramay+t	ramay+tiy	ramay+tuwA	
(22)		*throw*	**ramA**+**h**	**ram**+it+uh	**ram**+**uw**+**h**	ramay+t+uh	ramay+tiy+**h**	ramay+**tuw**+**h**	✓
(23)	**k.t.b + 1A2i3**	**kAtib**	kAtib	**kAtb**+it	**kAtb**+uwA	kAtib+t	kAtib+tiy	kAtib+tuwA	
(24)		*correspond with*	**kAtb**+uh	**kAtb**+it+uh	**kAtb**+**uw**+**h**	kAtib+t+uh	kAtib+tiy+**h**	kAtib+**tuw**+**h**	✓

Table 1: Segments of the verbal paradigms of four verbs illustrating complex morphotactics in MSA and EGY.

Levantine, and Gulf) that are commonly used on a daily basis. These dialects differ significantly from each other and from MSA in terms of phonology, morphology and lexicon although they share many similar aspects that support joint modeling. In Section 3.2, we present a more detailed example for MSA and EGY and compare them with each other.

Orthographic Inconsistency There is a high degree of orthographic inconsistency and variety in both MSA and dialectal Arabic (Zaghouani et al., 2014; Habash et al., 2018). For MSA there are standard guidelines with some minor regional differences; but dialectal Arabic has no official spelling rules. Habash et al. (2018) put forth a system for conventional orthography for dialectal Arabic (CODA), which has been used in some Arabic NLP resources. We consider CODA for EGY as our 'reference spelling,' but recognize its limitations.[3] We do not target modeling spelling variations in this work; and follow the philosophy that spelling errors need to be handled in components outside of the morphological analyzer. This is an important future research direction we plan to pursue.

[3]To allow comparing with previous work on Egyptian Arabic, we include a limited number of non-CODA-compliant phenomena, namely the negation and indirect object clitics, which CODA separates. This is simply a modeling decision that is independent of the framework.

3.2 Motivating Linguistic Phenomena

We describe in this section the linguistic facts relevant to this paper and approach. Arabic morphology includes a combination of templatic and concatenative morphemes, both with many allomoprhic variants.[4] Table 1 (MSA) contrasts parts of the verbal paradigm for five verbs, all of which have triliteral roots, but four are in Form I (1a2a3); and one is in Form III (1A2a3 in MSA, 1A2i3 in EGY). We consider a few subject suffixes and one pronominal clitic; and we indicate the verbal citation form (or Lemma).[5] The table marks all *default* morpheme realizations in gray, and indicates in underlined black font allomorph changes. For example the word كَتَبَتْهُ *katab+at+hu* (cell Table 1.(4d)) simply composes the morphemic forms of the templatic root *k.t.b.* and pattern *1a2a3* with the suffix *+at* (P3FS, perfective 3rd person feminine singular) and the enclitic pronoun *+hu* (direct object 3rd person masculine singular). However, only in 29 out of 60 cells in the MSA examples, and 38 out of 60 in the EGY examples, is an allomorph

[4]We limit our discussion in this paper to the fully diacritized orthographic forms of the allomorphs in Arabic. We do not model phonological representations and only discuss them where necessary.

[5]Arabic Lemmas are based on the perfective 3rd person masculine singular form without the final diacritic vowel.

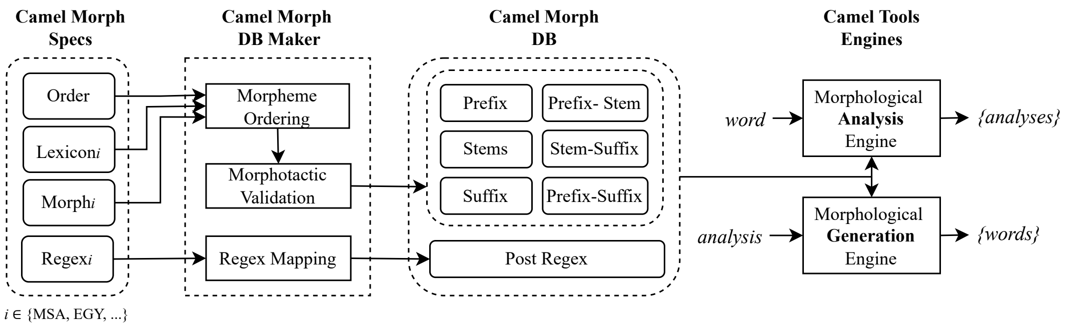

Figure 1: A high-level diagram of the CAMELMORPH approach.

of the root, pattern, suffix, or enclitic realized. It should be noted that although the five verbs happen to exist in both MSA and EGY, only 4 out of 60 forms in this example table match exactly. That said, the differences are regular and consistent, involving different suffix forms and different morphotactics. If we ignore the diacritics, 38 out of 60 forms match, an order of magnitude increase.

The following set of linguistic morphotactics can be observed in the examples in Table 1.

Geminate Verbs Verbs with geminate roots (equal second and third radicals) have an allomorph stem with an elided vowel in the context of vowel-initial suffixes (**v-suff**) in MSA, e.g. Table 1.(7c-8h) shows two variants: *ranan* (morpheme) and *ran* . The same phenomenon happens in EGY, but stems before consonant-initial suffixes (**c-suff**) also have a form different from the default interdigitation of root and pattern: a stem buffer vowel is inserted before the suffix, see Table 1.(19c-20h).

Defective Verbs Verbs with defective roots (third radical is *w* or *y*) have three allomorph stems that depend on the nature of the suffix in both MSA and EGY: vowel-initial, orthographically represented with a diacritic only (zero-letter suffix or **z-suff**), or being followed by an enclitic, e.g. Table 1.(9c-10h;21c-22h) shows four variants: *ramay* (morpheme), *ramaY*, *ramA*, and *ram*.

***t*-ending Verbs** Suffixes starting with the letter *t* in both MSA and EGY have orthographic allomorphs that replace the initial *t* with a letter gemination diacritic, Shadda ∼, when following verbs ending with the letter *t* (#t), e.g. Table 1.(5f-6h).

Masculine Plural Suffixes The masculine plural suffixes (+*uwA*, +*tum* and +*tuwA* in Table 1.(e;h) also have multiple forms that depend on the presence of enclitics and the verbal stem ending.

***hu* Enclitic** The MSA clitic +*hu* has an allomorphic variant +*hi* that harmonizes with suffixes ending with the high front vowel *i*, e.g. Table 1.(g4).

Short Vowel Elision The short vowel in EGY verb stem *kAt[i]b* (in brackets) is elided when the stem is followed by a vowel, whether from a suffix or an enclitic, e.g. Table 1.(23;24). Similarly, the vowel of the EGY enclitic +*[u]h* is elided after vowel-ending base words (stem+suffix), e.g. Table 1.(22c;22e;22g-h). Such transformations which only change word diacritics are ideally modeled as orthographic rewrites (reflecting phonological and morpho-phonological adjustments).

It should be noted that words can be composed completely of allomorphs of the underlying morphemes, e.g. EGY word *ran∼ay+tuw+h* in Table 1.(20h). These phenomena are only part of the complete list of phenomena we model, but are typical in terms of complexity. In the next section we will refer specifically to all of these phenomena and how we model them and their interactions.

4 The CAMELMORPH Approach

Figure 1 presents the overall approach we take in the CAMELMORPH project. The leftmost three boxes (CAMELMORPH **Specs**, **DB Maker** and DB) represent the offline process to create a Camel Tools-compatible morphological database (CAMELMORPH DB) from CAMELMORPH Specifications (**Specs**). The rightmost part of the figure represents the online process of using the DB in the Camel Tools morphological analysis and generation engines, where an input word results in a number of possible analyses, and an analysis can result in one or more words.

While we focus in this paper on the process of creating Camel Tools-compatible DBs, the overall approach can be used to generate other repre-

MO	Morph Order	DBPrefix	DBStem	DBSuffix
		[CONJ]	[PVStem] [PVBuff]	[PVSuff] [Pron]

		Class	Lemma/ Morpheme	Form	Gloss	Set Conds	Required Conds	katab+ti+hi	naHat+~umuw+hu	ran~+uw+hu	ranan+ta	ram+A+hu
Lexicon	L1	[PVStem]	katab	katab	write	trans		✓				
	L2	[PVStem]	naHat	naHat	sculpt	#t trans			✓			
	L3a	[PVStem]	ran~	ran~	ring	trans	v-suff			✓		
	L3b	[PVStem]	ran~	ranan	ring	trans	c-suff				✓	
	L4	[PVStem]	ramaY	ram	throw	#-ay trans						✓
Buffers	B1	[PVBuff]					else	✓			✓	
	B2a	[PVBuff]		aY			#-ay z-suff else					
	B2b	[PVBuff]		A			#-ay z-suff obj					✓
	B2c	[PVBuff]		ay			#-ay c-suff					
	B2d	[PVBuff]					#-ay v-suff					
Affixes	A1a	[PVSuff]	Suff.P3MS	a	he	v-suff	else					
	A1b	[PVSuff]	Suff.P3MS		he	z-suff	#-ay					✓
	A2	[PVSuff]	Suff.P3FS	at	she	v-suff						
	A3a	[PVSuff]	Suff.P3MP	uwA	they [mp]	v-suff	else else					
	A3b	[PVSuff]	Suff.P3MP	uw	they [mp]	v-suff	else obj			✓		
	A3c	[PVSuff]	Suff.P3MP	awA	they [mp]	v-suff	#-ay else					
	A3d	[PVSuff]	Suff.P3MP	aw	they [mp]	v-suff	#-ay obj					
	A4a	[PVSuff]	Suff.P2MS	ta	you [ms]	c-suff	else				✓	
	A4b	[PVSuff]	Suff.P2MS	~a	you [ms]	c-suff	#t					
	A5a	[PVSuff]	Suff.P2FS	ti	you [fs]	c-suff suff-i	else	✓				
	A5b	[PVSuff]	Suff.P2FS	~i	you [fs]	c-suff suff-i	#t					
	A6a	[PVSuff]	Suff.P2MP	tum	you [mp]	c-suff	else else					
	A6b	[PVSuff]	Suff.P2MP	tumuw	you [mp]	c-suff	else obj					
	A6c	[PVSuff]	Suff.P2MP	~um	you [mp]	c-suff	#t else					
	A6d	[PVSuff]	Suff.P2MP	~umuw	you [mp]	c-suff	#t obj		✓			
Clitics	C1	[Pron]									✓	
	C2a	[Pron]	Pron.3MS	hu	him	obj	trans else		✓	✓		✓
	C2b	[Pron]	Pron.3MS	hi	him	obj	trans suff-i	✓				

Figure 2: Sample Morphological Specifications for MSA perfective verbs, with examples.

sentations, e.g., finite-state machinery (directly or indirectly as Hulden and Samih (2012) has previously demonstrated). We chose to work with Camel Tools because it is a Python toolkit with growing popularity, and its morphological engine is relatively efficient.

Next, we describe the various components of the CAMELMORPH DB making process.

4.1 The CAMELMORPH Specifications

The morphological specifications (**Specs**) are the core of the CAMELMORPH project. There are four types of **Specs**: **Order**, **Lexicon**, various morphological units (**Morph**) – **Affixes**, **Clitics**, and **Buffers**, and Regular Expression Substitutions (**Regex**). An example of the set of **Order**, **Lexicon** and **Morph Specs** needed to model the MSA verbs in Table 1 is presented in Figure 2. We also present a **Regex** example to handle EGY verbs in Figure 2.

Morph Order The **Morph Order** specifies the arrangement of all the morphemes that can appear in a word. It only indicates the order of the morphemes, but not their morphotactic interactions. In Figure 2.(MO) (at the top of the figure), a minimal order is specified to form a perfective verb stem with a stem buffer, suffix, and pronominal clitic. The Prefix conjunction is allowed, but is not included in this example. Inside the **Morph Order**, the morphemes are specified by their class, e.g., **[PVStem]** refers to all the perfective verb stems.

The **Morph Order** also specifies which morpheme classes fall together to make the CAMELMORPH DB stem, and DB complex prefix and complex suffix sequences (sets of prefixes or suffixes that precede or follow the stem, respectively). In this example, a complex suffix sequence would include the resulting concatenation and rewriting of the perfective verb suffix and enclitic. The DB stem is created by concatenating the perfective verb stem and its buffer.

Different **Morph Order** lines are needed for the specifications of imperfect and command verb aspects. The number of specific **Morph Order** lines can vary depending on the choices of the linguist designing it.

Finally, since Arabic dialects and MSA all share the same morpheme order (with minor exceptions), we can use a common **Morph Order** for them all. This paves the way toward models of intra-word code-switching, which we leave for future work.

Lexicon, Buffers, Affixes, and Clitics All the morphemes used in the model are specified in a common style regardless of their type as lexical stem, inflectional affix, or attached clitic (syntactically independent, by phonological or orthographically dependent morpheme). The specification of any morphemes includes six elements.

(1) Class specifies the set of morphemes that the morpheme in question belongs to. The **Class** is the link between the Morph Order and the specific morphemes. It determines the position of the morpheme in the word.

(2) Lemma (in **Lexicon**) or **Morpheme** (in **Affixes and Clitics**) specifies the morpheme. For the **Lexicon**, the lemma is an abstraction over all the inflectional forms of a word's morphological inflection family. For the affixes and clitics, we use a functional specification. For example, *Suff.P3MP* refers to the perfective 3^{rd} person masculine plural suffix. **Stem Buffers**, as in the class **[PVBuffer]**, are not morphemes per se, but rather fragments of stems that vary highly in different contexts. As such **Stem Buffers** have no proper *morpheme* form defined; but their class specifies their position in the word. This concept is an innovation that allows us to refer to specific parts of the word form where complex morphotactic interactions happen and isolate it from the rest of the verbal patterns. There are two advantages to this approach. First, it reduces the total number of stems needed to be specified. So, for the defective verb in Table 1.(9-10;21-22), instead of listing four stems, *ramaY, ramA, ram,* and *ramay*, we only specify *ram* with a condition term (see below) marking its class as **#-ay**, i.e. defective. A second advantage of the buffer concept is that it allows us to relate dialect and MSA stems to each other, e.g. by treating the stems of geminate EGY verbs *ran∼* and *ran∼ay* as the same (*ran∼*) with different conditioned buffer values. Obviously, more complex non-suffixing or prefixing stem changes cannot be handled meaningfully using the buffer concept. Nevertheless, the current method is able to handle all Arabic-related concatenative phenomena perfectly.

(3) Form specifies the actual realized form of the morpheme. Each of the allomorphs of a morpheme gets a different **Form** line. For example, the two forms of the clitic Pron.3MS (*hu* and *hi*) share the same **Morpheme** and **Class** but have different forms. When multiple forms appear for the same morpheme, they need to be distinguished through different **Required** Conditions (**Conds**), which specify their complementary distribution.

(4) Gloss specifies the English meaning of the morpheme. It is not an essential feature of the model, but still useful to distinguish and explain any semantic differences.

(5) Set Conds and **(6) Required Conds** are a collection of terms that allow us to specify which allomorphs are compatible. Each form (allomorph) both *sets* and *requires* zero or more conditions to be true to be validated for use. Effectively, these conditions define the various contexts of co-occurrence and control the complementary distribution of the allomoprhs. In the example in Figure 2, there are nine condition terms:

(1-2) **trans** (transitive) and **obj** (object pronoun) license the use of pronominal clitics with transitive verbs. The **obj** condition also interacts with some suffix and verb buffer forms, e.g., Figure 2.(B2b;A6b;A6d).

(3-5) **v-suff**, **c-suff**, and **z-suff** specify the form of the suffixes: vowel-initial, consonant-initial or zero letter suffixes. They interact with verb stems and buffer forms.

(6-7) **#-ay** and **#t** specify the type of the verb as defective or t-ending, respectively.

(8) **suff-i** specifies the context of a suffixes ending with a *i*, e.g. Figure 2.(A5a;A5b).

(9) Finally, the term **else** is not a condition in itself, but it allows specifying the negation of a condition or set of conditions to model complementary distributions. The scope of an **else** is the column it appears in the **Required Conds** field. For example, in Figure 2.(C2a), the **else** indicates the negation of the **suff-i** in Figure 2.(C2b).

The right-hand side of Figure 2 presents five word examples from Table 1 and highlights the specific allomorphs which are selected to form them. Here, the order of the allomorphs is determined by

(Input)	(R1)	(R2)	(Clean up)
	V! → ø / V C _ C V	**V! → ø / V: _**	**! → ø**
kAti!b	kAti!b	kAti!b	**kAtib**
kAti!b+it	**kAtb**+it	kAtb+it	kAtb+it
kAti!b+uwA	**kAtb**+uwA	kAtb+uwA	kAtb+uwA
kAti!b+t	kAti!b+t	kAti!b+t	**kAtib**+t
kAti!b+tiy	kAti!b+tiy	kAti!b+tiy	**kAtib**+tiy
kAti!b+tuwA	kAti!b+tuwA	kAti!b+tuwA	**kAtib**+tuwA
kAti!b+u!h	**kAtb**+u!h	kAtb+u!h	kAtb+**uh**
kAti!b+it+u!h	**kAtb**+it+u!h	kAtb+it+u!h	kAtb+it+**uh**
kAti!b+uw+u!h	**kAtb**+uw+u!h	kAtb+uw+**h**	kAtb+uw+h
kAti!b+t+u!h	kAti!b+t+u!h	kAti!b+t+u!h	**kAtib**+t+**uh**
kAti!b+tiy+u!h	kAti!b+tiy+u!h	kAti!b+tiy+**h**	**kAtib**+tiy+h
kAti!b+tuw+u!h	kAti!b+tuw+u!h	kAti!b+tuw+**h**	**kAtib**+tuw+h

Table 2: Example of the application of rewrite rules to model the EGY verbs in Table 1.(23;24).

the **Morph Order**, and their compatibility through the set and required condition terms. For instance, in the second example, *naHat+∼umuw+hu*, the selected stem sets the conditions **#t** and **trans**. The *Suff.P2MP* has four allomorphs, and the *Pron.3MS* enclitic has two. Two of the *Suff.P2MP* allomophs are compatible with the stem's **#t**; and only one of these two (requiring **#t** and **obj**) is compatible with one of the *Pron.3MS* allomorphs (setting **obj** and requiring **trans** and *not* **suff-i**).

Regex Substitution Rules The last component of the CAMELMORPH **Specs** is the regex substitution rules. These rules can be used to model orthographic and phonological rewrite phenomena that involve morpheme diacritics. Table 2 illustrates how three rules can be used to model the vowel elision phenomena in EGY verbs in Table 1.(23;24). While the rules are implemented in the system with regex substitutions over orthographic forms, we represent them in the headers of Table 2 in SPE-type rule form (Chomsky and Halle, 1968) for readability.[6] To control the scope of the rules, we also extend the EGY verb stem and enclitic entries by marking elision candidates in the morphemes directly using a ! character. Only marked vowel diacritics in elision contexts are deleted. In Table 2, we use two rules (R1) and (R2), applied in sequence, followed by a final cleanup step to remove the ! marker for vowels that were not deleted. The morpheme boundary (+) is maintained for illustrative purposes. The grayed out cells indicate where a rule is applied, and the bolding indicates the affected morpheme.

[6]V represents any vowel, which corresponds to short vowels (diacritical marks [aiu]) and long vowels represented as (aA|iy|uw). The symbol V: represents long vowels only.

To allow us to use regex substitution rules within Camel Tools, we needed to make some extensions, which we plan on releasing in future Camel Tools releases.

4.2 The CAMELMORPH DB

We describe next the format of the CAMELMORPH DB, which we want to generate from the CAMELMORPH **Specs**. The CAMELMORPH DB has the same basic structure as the Buckwalter Arabic Morphological Analyzer (BAMA) (Buckwalter, 2004): it consists of (a) three lexical tables for complex prefixes (sequences of all possible co-occurring proclitics and prefixes), complex suffixes (sequences of all possible co-occurring suffixes and enclitics), and stems, and (b) three compatibility tables that specify allowed co-occurrences of complex prefixes with stems, stems with complex suffixes, and complex prefixes with complex suffixes (see Figure 1). During the analysis of a word, all combinations of allowable prefixes, stems, and suffixes matching the input in undiacritized space are considered and checked for existence in the lexical tables, and if so, their lexical categories are checked for compatibility in compatibility tables. Only valid and compatible combinations are output. This representation, which was pioneered by Buckwalter (2002) has been used by many other systems since then (Habash, 2004; Taji et al., 2018; Obeid et al., 2020) with numerous extensions. Habash (2004) demonstrated how to extend the algorithm with the same DB to perform generation. And Taji et al. (2018) demonstrated its use for reinflection and more complex gender/number modeling.

In our work, we extend Obeid et al. (2020)'s version by factoring out some hard-coded components to handle regex-based post-processing, and include them in the DB files. Our extensions will be integrated in Camel Tools once the full morphological models are finalized for all parts-of-speech.

4.3 The CAMELMORPH DB Maker

The CAMELMORPH **DB Maker** takes the CAMELMORPH **Morph Specs** as input and generates a BAMA-like CAMELMORPH DB. The basic algorithm behind this conversion is to identify all the unique condition terms set and required from all the instances of the classes ordered in the **Morph Order**. Each such combination is checked for compatibility (i.e., morphotactic validation) and incompatible combinations are discarded. Surface strings and features associated with compatible combina-

		MSA			EGY		
		CamelMorph **Specs**	CamelMorph **DB**	**Calima MSA**	CamelMorph **Specs**	CamelMorph **DB**	**Calima EGY**
(a)	**Lemmas**	9,331	9,331	9,112	8,404	8,404	10,661
(b)	**Prefix & Proclitic Morphemes (Allomorphs)**	34 (35)			23 (24)		
	Suffix & Enclitic Morphemes (Allomorphs)	119 (231)			39 (56)		
	Stem Buffers (Pre-stem/Post-stem)	6 / 71			4 / 47		
	Unique Condition Terms	35			30		
	Morph Order	69			11		
(c)	**Compatibility Tables**		48,798	2,733		13,734	6,649
	Complex Prefix Sequence		2,440	1,127		896	5,499
	Complex Suffix Sequence		12,902	574		2,619	1,237
(d)	**PV-Active Stems**	10,514	13,329	13,299	8,718	11,421	10,487
	PV-Passive Stems	10,509	11,483	303	n/a	n/a	3,558
	IV-Active Stems	10,486	14,305	13,382	8,406	18,052	4,264
	IV-Passive Stems	10,486	14,246	2,825	n/a	n/a	707
	CV Stems	10,486	12,785	66	8,406	9,402	6,054
(e)	**All Unique Diacritized Forms**		93,212,172	37,017,732		192,427,668	9,795,021
	All Unique Full Analyses		254,312,696	87,968,972		515,194,392	95,795,018
	All Unique Full Analyses without Clitics		1,602,403	321,323		159,697	52,190

Table 3: Statistics of the MSA and EGY verbal morphology models in CamelMorph.

tions are split into complex prefix, complex suffix and stem sequences and added into the lexical tables. Also, compatibility categories are created for the complex morpheme sequences, and are added to the compatibility tables. Memoization is used to speed up this process and make it efficient. As for the **Regex** substitution rules, they are simply copied into the DB with minimal processing.

5 Modeling Arabic Verbs

We developed two morphological models for MSA and EGY verbs. This effort made use of publicly available resources and tools, together with extensive reformulation, quality assessment, and reference cross-checking by a team of linguists and computer scientists.

For MSA in particular, we filled many known gaps in previous models, namely, adding passive and imperative forms, and the interrogative proclitic. We also added some admittedly archaic forms from Classical Arabic: energetic and extra energetic moods and indirect object pronominal clitics used with ditransitive verbs. For EGY, we paid special attention to completing verbal paradigms and modeling phono-orthographic phenomena.

Table 3 presents some of the statistics about these two models. For each variant (MSA and EGY), we present three sets of contrasting numbers: The CamelMorph **Specs**, the CamelMorph DB, and two pre-existing Camel Tools MSA and EGY databases for reference: **Calima MSA** and **Calima EGY**, respectively.[7]

The total number of lemmas in CamelMorph, and in **Calima MSA** and **Calima EGY** is generally comparable, although **Calima EGY** has more lemmas, presumably because automatic methods of lexicographic population were used in that effort. However, the number of lemmas does not indicate the modeling of their full paradigm.

The total number of morphological specifications outside the lexicon (Table 3.(b)) is two orders of magnitude smaller than the forms compiled into CamelMorph DB (Table 3.(c)). MSA **Specs** are 2.6 times the number of those in EGY (Table 3.(b)), which is expected given MSA's richer inflectional features space.

Looking at the stem counts in both MSA and EGY (Table 3.(d)), we notice that the number of forms in CamelMorph DB is higher than those in **Specs** by 26% and 52% for MSA and EGY, respectively. This increase is because of the pre- and post-buffer merging with the stems. Additionally, MSA Passive and CV (Command) forms were enriched to match the size of other verb forms. This is a major coverage increase resulting in more complete verbal paradigms. EGY on the other hand has no passive stems in CamelMorph, as by design, we consider them to be unaccusative derivational forms and not inflectional passives. This is a de-

[7]For MSA, we compared with the `calima-msa-s31_0.4.2.utf8.db` version (Taji et al., 2018) based on SAMA (Graff et al., 2009). For EGY we only compared to the `calima-egy-c044_0.2.0.utf8.db` entries (no MSA extensions) based on Habash et al. (2012).

sign choice of our **Specs** and not a limitation of the framework. We also note the large increase in EGY IV stems which is due to pre-stem buffers that interact with some of the person and number prefixes. One advantage of the CAMELMORPH framework is the ease of configuring the specifications of the DB being generated while considering tradeoffs in efficiency.

In terms of the total number of analyses (Table 3.(e)), CAMELMORPH has 2.9 times and 5.4 times the number of analyses in **Calima MSA** and **Calima EGY**, respectively. The total number of unique CAMELMORPH EGY full analyses is remarkably twice that of MSA, while the respective number of analyses without clitics is one-tenth. This is consistent with MSA having a richer inflection space; while EGY has a richer enclitic space, which includes negation clitics and indirect and benefactive object pronouns.

6 Evaluation

We present two recall-based evaluations to measure the quality of the new verb morphological models we developed.

MSA Recall Evaluation and Error Analysis To evaluate the quality of our CAMELMORPH MSA verb model in terms of recall of correct morphological analyses, we used manually annotated verbal entries in the training portion of the PATB (latest versions of parts 1,2,3) (Maamouri et al., 2004) as defined by Diab et al. (2013). There are 47,691 verb tokens (14,786 unique analyses). Out of all verb tokens, 98.4% of their full analyses were recalled successfully, and 0.3% were out-of-vocabulary. Of the remainder 1.4% with no perfect matches, we randomly selected 100 unique verb analysis examples and manually analyzed the results. In 93% of the cases, the PATB annotation was suboptimal or incorrect: 64% (absolute) of the cases come from the use of a li/PREP clitic with verbs instead of li/CONJ_SUB, which seems like a consistent annotation choice, albeit odd for verbs. In 29% of the cases, the PATB annotation did not specify a lemma or diacritization (13%), or had an incorrect lemma or diacritization (16%). In 6% (absolute) of the latter, the lemma was incorrectly specified in the passive voice. Our CAMELMORPH MSA system failed to produce matches in 7% of the sample. Most of the cases were missing lexical entries or alternative spellings of some clitic combinations, e.g., *fa+li* as *fa+l*.

EGY Recall Evaluation and Error Analysis Similar to our MSA recall evaluation, we conducted a recall evaluation for EGY using the verbal entries in the training portion of the LDC's ARZATB (Maamouri et al., 2012) as defined by Diab et al. (2013). Given the inconsistencies in some of the ARZATB entries, we used a version of ARZATB that was automatically synchronized with a combination of EGY and MSA analyzers as our reference. This version was reported on in previous publications (Pasha et al., 2014; Zalmout and Habash, 2019; Inoue et al., 2022). For recall evaluation, we also use CAMELMORPH EGY and MSA together in a similar manner, with preference towards EGY if an imperfect (i.e., not all analysis features match) tie is reached. To deal with the common spelling variations in the input words, we use the a dediacritized version of the correct answer, which is intended to mimic a more CODA-compliant spelling. Of the original token count of 20,339 verbs, 69.9% of the full analyses are recalled successfully. In 1.4%, no analysis is generated, and in 24.2%, no single analysis matches the reference analysis perfectly. 4.5% of the reference analyses were not usable due to synchronization issues. We took a sample of 100 unique verb analyses from the set with no matches, and analyzed them manually. Almost half of the sample (47%) was due to reference errors. Another third (37%) involved valid alternative diacritizations reflecting different pronunciations (e.g. مسك *misik* vs *masak* 'to hold'). 10% were due to missing entries; and 6% were due to diacritization errors that can be fixed with regular expressions.

The difference in recall between MSA and EGY is striking but completely understandable given the differences in standardization traditions and the maturity of existing resources.

7 Conclusion and Future Work

We presented a new approach to modeling Arabic morphotactics and demonstrated its usefulness by creating a large-scale verbal analyzer for MSA and EGY using a common framework. All of our models and code will be publicly available. In the future, we plan to extend our work to all other POS classes in MSA and EGY, as well as target other dialects of Arabic. Some of the interesting challenges we want to address are noisy spelling, dialect-MSA intra-word code switching, and template-based backoff modeling.

References

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.

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

Ramy Baly, Hazem Hajj, Nizar Habash, Khaled Bashir Shaban, and Wassim El-Hajj. 2017. A sentiment treebank and morphologically enriched recursive deep models for effective sentiment analysis in Arabic. *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)*, 16(4):23.

Kenneth Beesley. 1996. Arabic Finite-State Morphological Analysis and Generation. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, pages 89–94, Copenhagen, Denmark.

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*.

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.

Barli Bram. 2012. Three models of English morphology. *LLT Journal: A Journal on Language and Language Teaching*, 15(1):179–185.

Tim Buckwalter. 2002. Buckwalter Arabic morphological analyzer version 1.0. Linguistic Data Consortium (LDC) catalog number LDC2002L49, ISBN 1-58563-257-0.

Tim Buckwalter. 2004. Buckwalter Arabic Morphological Analyzer Version 2.0. LDC catalog number LDC2004L02, ISBN 1-58563-324-0.

Noam Chomsky and Morris Halle. 1968. *The sound pattern of English*. New York: Harper and Row.

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

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. 2004. Large Scale Lexeme Based Arabic Morphological Generation. In *Proceedings of Traitement Automatique des Langues Naturelles (TALN)*, pages 271–276, Fez, Morocco.

Nizar Habash, Fadhl Eryani, Salam Khalifa, Owen Rambow, Dana Abdulrahim, Alexander Erdmann, Reem Faraj, Wajdi Zaghouani, Houda Bouamor, Nasser Zalmout, Sara Hassan, Faisal Al shargi, Sakhar Alkhereyf, Basma Abdulkareem, Ramy Eskander, Mohammad Salameh, and Hind Saddiki. 2018. Unified guidelines and resources for Arabic dialect orthography. In *Proceedings of the Language Resources and Evaluation Conference (LREC)*, Miyazaki, Japan.

Nizar Habash, Ramy Eskander, and Abdelati Hawwari. 2012. A Morphological Analyzer for Egyptian Arabic. In *Proceedings of the Workshop of the Special Interest Group on Computational Morphology and Phonology (SIGMORPHON)*, pages 1–9, Montréal, Canada.

Nizar Habash and Owen Rambow. 2006. MAGEAD: A morphological analyzer and generator for the Arabic dialects. In *Proceedings of the International Conference on Computational Linguistics and the Conference of the Association for Computational Linguistics (COLING-ACL)*, pages 681–688, Sydney, Australia.

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*, pages 15–22. Springer, Netherlands.

Nizar Y Habash. 2010. *Introduction to Arabic natural language processing*. Morgan & Claypool Publishers.

Charles F Hockett. 1954. Two models of grammatical description. *Word*, 10(2-3):210–234.

Mans Hulden and Younes Samih. 2012. Conversion of procedural morphologies to finite-state morphologies: a case study of Arabic. In *Proceedings of the 10th International Workshop on Finite State Methods and Natural Language Processing*, pages 70–74.

Go Inoue, Salam Khalifa, and Nizar Habash. 2022. Morphosyntactic tagging with pre-trained language models for Arabic and its dialects. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1708–1719, Dublin, Ireland. Association for Computational Linguistics.

George Kiraz. 1994. Multi-tape Two-level Morphology: A Case study in Semitic Non-Linear Morphology. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, pages 180–186, Kyoto, Japan.

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

Mohamed Maamouri, Ann Bies, Seth Kulick, Dalila Tabessi, and Sondos Krouna. 2012. Egyptian Arabic Treebank DF Parts 1-8 V2.0 - LDC catalog numbers LDC2012E93, LDC2012E98, LDC2012E89, LDC2012E99, LDC2012E107, LDC2012E125, LDC2013E12, LDC2013E21.

Ossama Obeid, Nasser Zalmout, Salam Khalifa, Dima Taji, Mai Oudah, Bashar Alhafni, Go Inoue, Fadhl Eryani, Alexander Erdmann, and Nizar Habash. 2020. CAMeL tools: An open source python toolkit for Arabic natural language processing. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 7022–7032, Marseille, France. European Language Resources Association.

Arfath Pasha, Mohamed Al-Badrashiny, Mona Diab, Ahmed El Kholy, Ramy Eskander, Nizar Habash, Manoj Pooleery, Owen Rambow, and Ryan Roth. 2014. MADAMIRA: A fast, comprehensive tool for morphological analysis and disambiguation of Arabic. In *Proceedings of the Language Resources and Evaluation Conference (LREC)*, pages 1094–1101, Reykjavik, Iceland.

Otakar Smrž. 2007. *Functional Arabic Morphology. Formal System and Implementation*. Ph.D. thesis, Charles University in Prague, Prague, Czech Republic.

Dima Taji, Salam Khalifa, Ossama Obeid, Fadhl Eryani, and Nizar Habash. 2018. An Arabic Morphological Analyzer and Generator with Copious Features. In *Proceedings of the Fifteenth Workshop on Computational Research in Phonetics, Phonology, and Morphology (SIGMORPHON)*, pages 140–150.

Wajdi Zaghouani, Behrang Mohit, Nizar Habash, Ossama Obeid, Nadi Tomeh, Alla Rozovskaya, Noura Farra, Sarah Alkuhlani, and Kemal Oflazer. 2014. Large Scale Arabic Error Annotation: Guidelines and Framework. In *Proceedings of the Language Resources and Evaluation Conference (LREC)*, Reykjavik, Iceland.

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

Nasser Zalmout and Nizar Habash. 2019. Joint diacritization, lemmatization, normalization, and fine-grained morphological tagging. *arXiv preprint arXiv:1910.02267*.