Chapter 2
Morphological Processing of Semitic Languages

Shuly Wintner

2.1 Introduction

This chapter addresses morphological processing of Semitic languages. In light of the complex morphology and problematic orthography of many of the Semitic languages, the chapter begins with a recapitulation of the challenges these phenomena pose on computational applications. It then discusses the approaches that were suggested to cope with these challenges in the past. The bulk of the chapter, then, discusses available solutions for morphological processing, including analysis, generation, and disambiguation, in a variety of Semitic languages. The concluding section discusses future research directions.

Semitic languages are characterized by complex, productive morphology, with a basic word-formation mechanism, root-and-pattern, that is unique to languages of this family. Morphological processing of Semitic languages therefore necessitates technology that can successfully cope with these complexities.[1] Several linguistic theories, and, consequently, computational linguistic approaches, are often developed with a narrow set of (mostly European) languages in mind. The adequacy of such approaches to other families of languages is sometimes sub-optimal. A related issue is the long tradition of scholarly work on some Semitic languages, notably Arabic [109] and Hebrew [117], which cannot always be easily consolidated with contemporary approaches.

Inconsistencies between modern, English-centric approaches and traditional ones are easily observed in matters of lexicography. In order to annotate corpora or produce tree-banks, an agreed-upon set of part-of-speech (POS) categories is

[1]Parts of this introduction are based on and adapted from [137].

S. Wintner (✉)
University of Haifa, Haifa, Israel
e-mail: shuly@cs.haifa.ac.il

I. Zitouni (ed.), *Natural Language Processing of Semitic Languages*,
Theory and Applications of Natural Language Processing,
DOI 10.1007/978-3-642-45358-8_2, © Springer-Verlag Berlin Heidelberg 2014

required. Since early approaches to POS tagging were limited to English, resources for other languages tend to use "tag sets", or inventories of categories, that are minor modifications of the standard English set. Such an adaptation is problematic for Semitic languages. As noted in the previous chapter, there are good reasons to view nouns, adjectives and numerals as sub-categories of a single category, *nominals*. Furthermore, the distinction between verbs and nominals is blurry. Netzer et al. [101] discuss a similar issue related to the correct tagging of modals in Hebrew. Even the correct citation form to use in dictionaries is a matter of some debate, as Arabic traditional dictionaries are root-based, rather than lemma-based [43].

These issues are complicated further when morphology is considered. The rich, non-concatenative morphology of Semitic languages frequently requires innovative solutions that standard approaches do not always provide. After a brief introduction of some basic notions (Sect. 2.2), we recapitulate some of these challenges in Sect. 2.3, and review the long line of proposed computational solutions in Sect. 2.4. Section 2.5 lists some available computational implementations for the morphology of various Semitic languages. Section 2.6 then discusses implementations of morphological disambiguation for several Semitic languages. We conclude the chapter with directions for future research.

2.2 Basic Notions

The word 'word' is one of the most loaded and ambiguous notions in linguistic theory [76]. Since most computational applications deal with written texts (as opposed to spoken language), the most useful notion is that of an *orthographic* word. This is a string of characters, from a well-defined alphabet of *letters*, delimited by spaces, or other delimiters, such as punctuation. A *text* typically consists of sequences of orthographic words, delimited by spaces or punctuation; orthographic words in a text are often referred to as *tokens*.

Orthographic words are frequently not atomic: they can be further divided to smaller units, called *morphemes*. Morphemes are the smallest meaning-bearing linguistic elements; they are elementary pairings of form and meaning. Morphemes can be either *free*, meaning that they can occur in isolation, as a single orthographic word; or *bound*, in which case they must combine with other morphemes in order to yield a word. For example, the word *two* consists of a single (free) morpheme, whereas *dogs* consists of two morphemes: the free morpheme *dog*, combined with the bound morpheme *-s*. The latter form indicates the fact that it must combine with other morphemes (hence the preceding dash); its function is, of course, denoting plurality. When a word consists of some free morpheme, potentially with combined bound morphemes, the free morpheme is called a *stem*, or sometimes *root*.

Bound morphemes are typically *affixes*. Affixes come in many varieties: *prefixes* attach to a stem before the stem (e.g., *re-* in *reconsider*), *suffixes* attach after the stem (*-ing* in *dreaming*), *infixes* occur inside a stem (e.g., the *t* in Arabic *ijtahada*, from *jahada*), and *circumfixes* surround the stem they combine with (e.g., Hebrew *ti–u* in *tigdelu* "you will grow"). Some bound morphemes are likely *clitics* [140], but as in the previous chapter, we blur the distinction between clitics and affixes here.

Morphological processes define the shape of words. They are usually classified to two types of processes. *Derivational* morphology deals with word formation; such processes can create new words from existing ones, potentially changing the category of the original word. For example, the processes that create *faithfulness* from *faithful*, and *faithful* from *faith*, are derivational. Such processes are typically not highly productive; for example, one cannot derive **loveful* from *love*. In contrast, *inflectional* morphology yields *inflected forms*, variants of some *base*, or *citation* form, of words; these forms are constructed to adhere to some syntactic constraints, but they do not change the basic meaning of the base form. Inflectional processes are usually highly productive, applying to most members of a particular word class. For example, English nouns inflect for *number*, so most nouns occur in two forms, the singular (which is considered the citation form) and the plural, regularly obtained by adding the suffix *-s* to the base form.

Word formation in Semitic languages is based on a unique mechanism, known as *root-and-pattern*. Words in this language family are often created by the combination of two bound morphemes, a *root* and a *pattern*. The root is a sequence of consonants only, typically three; and the pattern is a sequence of vowels and consonants with open slots in it. The root combines with the pattern through a process called *interdigitation*: each letter of the root (*radical*) fills a slot in the pattern. For example, the Hebrew root *p.t.x*, denoting a notion of opening, combines with the pattern *maCCeC* (the slots are denoted by *C*), typically denoting tools and instruments, to yield *maptex* "key".

In addition to the unique root-and-pattern morphology, Semitic languages are characterized by a productive system of more standard affixation processes. These include prefixes, suffixes, infixes and circumfixes, which are involved in both inflectional and derivational processes (see the previous linguistic-introduction chapter). Consider the Arabic word *wasayaktubuwnaha* "and they will write it". A possible analysis of this complex word defines the stem as *aktub* "write", with an inflectional circumfix, *y—uwna*, denoting third person masculine plural, an inflectional suffix, *-ha* "it", and two prefixes, *sa-* "will" and *wa-* "and". For more information on Arabic morphology from a computational perspective, see [127]; [63, Chap. 4]. For a good introduction to computational morphology in general, consult [112, 128].

2.3 The Challenges of Morphological Processing

Morphological processing is a crucial component of many natural language processing (NLP) applications. Whether the goal is information retrieval, question answering, text summarization or machine translation, NLP systems must be aware of word structure. For some languages and for some applications, simply stipulating a list of surface forms is a viable option; this is not the case for languages with complex morphology, in particular Semitic languages, both because of the huge number of potential forms and because of the difficulty of such an approach to handle out-of-lexicon items (in particular, proper names), which may combine with prefix or suffix particles. For example, the Hebrew prefix *l*- "to" can combine with any proper name denoting a location, an organization or a person.

An alternative solution would be a dedicated morphological analyzer, implementing the morphological and orthographic rules of the language. Ideally, a morphological analyzer for any language should reflect the rules underlying derivational and inflectional processes in that language. Of course, the more complex the rules, the harder it is to construct such an analyzer. The main challenge of morphological analysis of Semitic languages stems from the need to faithfully implement a complex set of interacting rules, some of which are non-concatenative.

Once such a grammar is available, it typically produces more than one analysis for any given surface form; in other words, Semitic languages exhibit a high degree of *morphological ambiguity*, which has to be resolved in a typical computational application. The level of morphological ambiguity is higher in many Semitic languages than it is in English, due to the rich morphology and deficient orthography. This calls for sophisticated methods for disambiguation. While in English (and other European languages) morphological disambiguation may amount to POS tagging, Semitic languages require more effort, since determining the correct POS of a given token is intertwined with the problem of segmenting the token to morphemes, the set of morphological features (and their values) is larger, and consequently the number of classes is too large for standard classification techniques. Several models were proposed to address these issues.

Contemporary approaches to part-of-speech tagging are all based on machine learning: a large corpus of text is manually annotated with the correct POS for each word; then, a classifier is trained on the annotated corpus, resulting in a system that can predict POS tags for unseen texts with high accuracy. The state of the art in POS tagging for English is extremely good, with accuracies that are indistinguishable from human level performance. Various classifiers were built for this task, implementing a variety of classification techniques, such as Hidden Markov Models (HMM) [26], Average Perceptron [37], Maximum Entropy [111, 130, 131, 133, 134], Support Vector Machines (SVM) [58], and others.

For languages with complex morphology, and Semitic languages in particular, however, these standard techniques do not perform as well, for several reasons:

1. Due to issues of orthography, a single token in several Semitic languages can actually be a sequence of more than one lexical item, and hence be associated with a sequence of tags. For example, the Hebrew form *šbth* can be interpreted as *š+b+h+th* "that+in+the+tea", corresponding to a tag sequence consisting of a subordinating conjunction, followed by a preposition, a determiner and a noun.
2. The rich morphology implies a much larger tagset, since tags reflect the wealth of morphological information which words exhibit. The richer tagset immediately implies problems of data sparseness, since most of the tags occur only rarely, if at all, in a given corpus. For example, the MILA Hebrew morphological analyzer [80] produces 22 different parts of speech, some with subcategories; 6 values for the number feature (including disjunctions of values), 4 for gender, 5 for person,

7 for tense and 3 for nominal state. Possessive pronominal suffixes can have 15 different values, and prefix particle sequences can theoretically have hundreds of different forms. While not all the combinations of these values are possible, the number of possible analyses (i.e., the size of the tagset) is in the thousands.

3. As a result of both orthographic deficiencies and morphological wealth, word forms in Semitic languages tend to be ambiguous. Itai and Wintner [80] report an average of 2.6 analysis per word in their corpora. In some cases, different analyses are identical in all their features, except the lexical item, a phenomenon that makes morphological disambiguation closer to the problem of word sense disambiguation than to standard POS tagging.
4. Word order in Semitic is relatively free, and in any case freer than in English.

An additional challenge of morphological processing of Semitic languages, with an emphasis on Arabic, stems from the *form–function* discrepancy. The *form* of words in these languages typically provides good hints for some of the morphological features of the word, or its *function*; in many cases, however, the form and the function are in disagreement. A concrete example is the phenomenon of *broken plural* forms in Arabic. For a significant number of nouns, the plural form is not obtained by the concatenation of a plural suffix, but rather by an internal change (not unlike umlauting) that renders the surface form, which is plural in function, singular in form. A related phenomenon involves gender agreement in Arabic: while adjectives must agree with their head nouns in gender, when the noun is plural and irrational (non-human), the adjective must be feminine singular (see a detailed discussion in [3]).

Somewhat similarly, Hebrew nouns are marked for gender by a small number of suffixes; but several masculine-appearing nouns are actually feminine, and vice versa. Furthermore, Hebrew has two plural suffixes, -*im* for plural nouns and -*wt* for feminine nouns, but a non-negligible number of feminine nouns take the masculine suffix and vice versa.

Finally, it is important to note that morphological processing of Semitic languages is often handicapped by subtle orthographic issues [28]. Hebrew, for example, has a writing system that encodes vocalic information using a large set of diacritics; this system, however, is rarely in use, and most contemporary texts are written without the diacritics. Unfortunately, while a standard for non-vocalized Hebrew exists [53], it is not adhered to by most authors, and consequently the same word may be spelled in different ways, sometimes even within the same document. Arabic suffers from related problems, especially when the various dialects are considered, in which standardized forms do not exist [71].

2.4 Computational Approaches to Morphology

No single method exists that provides an adequate solution for the challenges involved in morphological processing of Semitic languages. The most common approach to morphological processing of natural language is *finite-state technology*

[22, 81, 83, 89, 113]. The adequacy of this technology for Semitic languages has frequently been challenged, but clearly, with some sophisticated developments, such as flag diacritics [19], multi-tape automata [88] or registered automata [36], finite-state technology has been effectively used for describing the morphological structure of several Semitic languages [8, 16, 17, 68, 85, 88, 138]. We survey this technology in the present section.

2.4.1 Two-Level Morphology

Two-level morphology was "the first general model in the history of computational linguistics for the analysis and generation of morphologically complex languages" [84]. Developed by Koskenniemi [89], this technology facilitates the specification of rules that relate pairs of surface strings through systematic rules. Such rules, however, do not specify how one string is to be derived from another; rather, they specify mutual constraints on those strings. Furthermore, rules do not apply sequentially. Instead, a set of rules, each of which constrains a particular string-pair correspondence, is applied in parallel, such that all the constraints must hold simultaneously. In practice, one of the strings in a pair would be a surface realization, while the other would be an underlying form. Thus, for example, the Hebrew surface form *[xicim]* "arrows" can correspond to the underlying form *xec+im* through the mapping:

```
x i c 0 i m
x e c + i m
```

where '0' is the empty string. The example reflects a rule that maps *[i]* to *e* in the context of the plural suffix *im*; the upper string is the surface realization, and the lower is its underlying form. The underlying forms are further constrained by consulting a *lexicon*.

One of the greatest advantages of two-level morphology is that rules are entirely declarative: indeed, the original formulation of [89] allows for both analysis and generation within the same grammar. The formalism was later implemented as part of the Xerox tools (Sect. 2.4.3); two-level rules are compiled to finite-state transducers, which indeed allow for both analysis and generation.

2.4.2 Multi-tape Automata

Two-level morphology was applied to one Semitic language, Akkadian, by Kataja and Koskenniemi [85]. However, the applicability of the technology to Semitic languages in general was challenged by Lavie et al. [91], who describe some difficulties of this technology in accounting for Hebrew verb inflections. Lavie et al. [91] conclude: "The Two Level rules are not the natural way to describe… verb inflection process. The only alternative choice… is to keep all bases… it seems wasteful to save all the secondary bases of verbs of the same pattern."

Addressing such issues, [88] expands the traditional two-level model to n-tape automata, following insight originally suggested by Kay [86] and Kataja and Koskenniemi [85]. The two levels of expression are expanded: one of them is retained for the surface form, but the lexical string can now be spread across multiple representations (e.g., one for the root and one for the pattern). Thus, elements that are separated on the surface (such as the root's consonants) can be adjacent on a particular lexical level.

Using multi-tape automata, [88] provides elegant solutions for derivational and inflectional morphology of two Semitic languages, Syriac and Arabic. The same approach is then extended by Habash et al. [68], who define a multi-tape automaton consisting of *five* tapes: one for the pattern and affixational morphemes, one for the root, one for the vocalism, one for phonological information and one for the orthography. This model is then successfully applied to both MSA and dialectal Arabic [6, 65, 68] The model is detailed in [7].

Hulden [79], however, notes that no other systems were built using this technology, and conjectures that the reason may be that "when the number of tapes grows, the required joint symbol alphabet grows with exponential rapidity unless special mechanisms are devised to curtail this growth. This explosion in the number of transitions in an n-tape automaton can in many cases be more severe than the growth in the number of states of a complex grammar." To alleviate the problem, [79] describes an algorithm that simulates an n-tape automaton with a simple single-tape finite-state machine. Consequently, the elegant representation of multi-tape automata can be retained, while the conversion algorithm facilitates an implementation using standard tools such as the ones discussed in Sect. 2.4.3. Indeed, [79] uses Foma [78] to efficiently implement a grammar of Arabic verbal morphology over 2,000 roots.

2.4.3 The Xerox Approach

One of the most popular toolboxes for developing finite-state grammars comes from Xerox, and is discussed in detail by Beesley and Karttunen [22]. The Xerox tools consist of several description languages, including a formalization of two-level morphology, but also a variant, XFST, of the calculus proposed by Kaplan and Kay [83]. Along with the description languages come compilers that convert morphological grammars to finite-state transducers, and programs that implement analysis and generation with these transducers.

To address the special needs of languages with non-concatenative morphology, XFST provides two special mechanisms. First, the *compile-replace* mechanism [21] facilitates the reapplication of the regular-expression compiler to its own output. This allows for a compact definition of some non-concatenative morphological processes, and [19] uses it to construct a morphological grammar of Arabic. Second, Beesley [20] proposes a method, called *flag diacritics*, which adds features to symbols in regular expressions to enforce dependencies between separated parts of a string. These dependencies are then enforced by different kinds of unification actions.

While the Xerox tools have for many years been the de-facto standard of finite-state technology, they have also been proprietary, a fact that limited their distribution and popularity. Several competing formalisms were developed over the years, of which we note Foma [78] because it is, to a large extent, compatible with the syntax of several Xerox tools, while being completely open-source.

2.4.4 Registered Automata

Finite-state registered automata [36] were developed with the goal of facilitating the expression of various non-concatenative morphological phenomena in an efficient way. The main idea is to augment standard finite-state automata with (finite) amount of memory, in the form of *registers* associated with the automaton transitions. This is done in a restricted way that saves space but does not add expressivity. The number of registers is finite, usually small, and eliminates the need to duplicate paths as it enables the automaton to 'remember' a finite number of symbols. Technically, each arc in the automaton is associated (in addition to an alphabet symbol) with an action on the registers. Cohen-Sygal and Wintner [36] define two kinds of actions, *read* and *write*. The former allows an arc to be traversed only if a designated register contains a specific symbol. The latter writes a specific symbol into a designated register when an arc is traversed.

Cohen-Sygal and Wintner [36] show that finite-state registered automata can efficiently model several non-concatenative morphological phenomena, including circumfixation, root and pattern word formation in Semitic languages, vowel harmony, limited reduplication etc. The representation is highly efficient: for example, to account for all the possible combinations of r roots and p patterns, an ordinary FSA requires $O(r \times p)$ arcs whereas a registered automaton requires only $O(r + p)$ arcs. Unfortunately, no implementation of the model exists as part of an available finite-state toolkit.

2.4.5 Analysis by Generation

Most of the approaches discussed above allow for a *declarative* specification of (morphological) grammar rules, from which both analyzers and generators can be created automatically. A simpler, less generic yet highly efficient approach to the morphology of Semitic languages had been popular with actual applications. In this framework, which we call *analysis by generation* here, the morphological rules that describe word formation and/or affixation are specified in a way that supports generation, but not necessarily analysis. Coupled with a lexicon of morphemes (typically, base forms and concatenative affixes), such rules can be applied in one direction to generate all the surface forms of the language. This can be done off-line, and the generated forms can then be stored in a database; analysis, in this paradigm, amounts more or less to simple table lookup.

Some of the very first morphological processors of Semitic languages were developed in this way. Probably the first example is the Hebrew morphological system of [29, 122], see Sect. 2.5.3. Exactly the same approach is now used in the MILA morphological analyzer of Hebrew [80] (Sect. 2.5.3). And a very similar approach underlies the most popular morphological analyzer of Arabic, BAMA [27]: Again, a set of rules (called the *compatibility table*) determines how lexemes and affixes (stored in separate lexicons) can combine; at analysis time, a surface form is divided to a sequence of prefix + lexeme + suffix in all possible ways, and the lexicons are consulted to determine which potential combination is indeed valid (see Sect. 2.5.2).

2.4.6 Functional Morphology

Functional morphology [51] is a computational framework for defining language resources, in particular lexicons. It is a language-independent tool, based on a *word-and-paradigm* model, which allows the grammar writer to specify the inflectional paradigms of words in a specific language in a similar way to printed paradigm tables. A lexicon in functional morphology consists of a list of words, each associated with its paradigm name, and an inflection engine that can apply the inflectional rules of the language to the words of the lexicon.

This framework was used to define morphological grammars for several languages, including modeling of non-concatenative processes such as vowel harmony, reduplication, and templatic morphology [50]. In particular, [125] uses this paradigm to implement a morphological processor of Arabic.

2.5 Morphological Analysis and Generation of Semitic Languages

We survey in this section the current state of the art in morphological analysis and generation of various Semitic languages. While much effort was put into the development of systems for processing (Modern Standard) Arabic and Hebrew, for other languages the development of such tools lags behind.

We use the term *analysis* in this chapter to refer to the task of producing *all* the possible analyses of a given word form, independently of its context. The problem of producing the *correct* analysis in the context is called *disambiguation* here, and is discussed in detail in Sect. 2.6.

2.5.1 Amharic

Computational work on Amharic began only recently. Fissaha and Haller [49] describe a preliminary lexicon of verbs, and discuss the difficulties involved in implementing verbal morphology with XFST. XFST is also the framework of choice for the development of an Amharic morphological grammar [8, 9]; but evaluation on a small set of 1,620 words reveal that while the coverage of the grammar on this corpus is rather high (85–94 %, depending on the part of speech), its precision is low and many word forms (especially verbs) are associated with wrong analyses.

Argaw and Asker [11] describe a stemmer for Amharic. Using a large dictionary, the stemmer first tries to segment surface forms to sequences of prefixes, stem, and affixes. The candidate stems are then looked up in the dictionary, and the longest found stem is chosen (ties are resolved by the frequency of the stem in a corpus). Evaluation on a small corpus of 1,500 words shows accuracy of close to 77 %.

The state of the art in Amharic, however, is most probably *HornMorpho* [56,57]: it is a system for morphological processing of Amharic, as well as Tigrinya (another Ethiopian Semitic language) and Oromo (which is not Semitic). The system is based on finite-state technology, but the basic transducers are augmented by feature structures, implementing ideas introduced by Amtrup [10]. Manual evaluation on 200 Tigrinya verbs and 400 Amharic nouns and verbs shows very accurate results: in over 96 % of the words, the system produced all and only the correct analyses.

2.5.2 Arabic

Recent years saw an increasing interest in computational approaches to Arabic morphology [5]. Attempts to automatically analyze the structure of Arabic words date back over 50 years [34]. Several early works were done in the finite-state framework of the Xerox tools (Sect. 2.4.3). Beesley [17] describes an early system for morphological analysis and generation. The input is given in the standard Arabic script, either vocalized or not, and the output includes the root, the pattern, a list of affixes and a plethora of morphological information in the form of feature-value pairs. The implementation was carried out in an early version of the Xerox tools, which resembles to a high degree the two-level formalism. Beesley [18] uses the newly-introduced *flag diacritics* in XFST to provide a more elegant morphological grammar, whose implementation as an online web-based service is described in [19]. In a similar vein, [88] demonstrates the utility of multi-tape automata (Sect. 2.4.2) by providing examples from Arabic.

Other works from approximately the same period seem to be more focused on an actual application, rather than on elegant and efficient representation of morphological processes. Al-Shalabi and Evens [4] extend an earlier system and a large-scale lexicon to an analyzer for (mainly regular, but also some irregular) verb forms. Berri et al. [25] describe a morphological analyzer that uses an object-oriented model to represent morphological rules affecting both verbs and nouns, along with a dedicated algorithm that identifies affixes and separates them from the stem. Rules are divided into *regular* and *exception* handling. No details are provided on the coverage of the system. Darwish [39] discusses the rapid development of a shallow morphological analyzer. Given a large set of word–root pairs, the system learns to identify the roots of (mainly regular) word forms. Evaluation on a large set reveals high coverage but, unsurprisingly, rather low accuracy.

The state of the art in Arabic morphological analysis, however, is most likely *BAMA*, the morphological analyzer of Buckwalter [27], which combines wide coverage with detailed, linguistically informative analyses. BAMA is based on a large-scale lexicon of base forms, along with lists of prefixes and suffixes. A second part of this database includes a list of compatibility rules, which govern the combination of stems with affixes. Finally, an efficient engine implements the rules as well as lexical lookup. The result is a highly-efficient, broad-coverage (and freely-available) analyzer. BAMA (most recently called *SAMA*, or *Standard Arabic Morphological Analyzer*) is the official morphological analyzer used by the Linguistic Data Consortium (LDC) for the *Penn Arabic Treebank* [93], a language resource used by most practitioners interested in Arabic disambiguation and parsing.

Based on this analyzer, [61] has built *Aragen*, a system for generating Arabic word forms from underlying morphological descriptions. Using the same databases of [27], Aragen implements a different engine that reverses the operation of the analyzer. The current state of the art in Arabic morphological generation is a revised version of Aragen, called *Almorgeana* [62].

A different approach was advanced in the context of the NooJ platform [123]. NooJ is a linguistic development environment that facilitates the definition of large-coverage dictionaries and grammars, compiling them into systems that can efficiently parse real-world corpora. NooJ has been used for the construction of Arabic morphological and syntactic processors [23], as well as for part-of-speech tagging and morphological analysis [82].

More recent approaches to Arabic morphology are done with an eye to syntactic processing. For example, [125,126] addresses Arabic morphology in the framework of Functional Morphology [51]. His system, *ElixirFM*, extends the original functionality of the framework by addressing the specific needs of Arabic morphology. The system not only provides (derivational and inflectional) analyses of word forms, but can also recognize their grammatical functions. A different system, *Kawaakib* [12], combines a set of both morphological and syntactic operators that are represented as finite-state automata.

For a full, detailed and lucid exposition of computational processing of Arabic, with a focus on morphology, refer to [63].

2.5.3 Hebrew

Computational work on Hebrew began almost fifty years ago.[2] Very early approaches [29, 122] were superseded by a large-scale project dealing with various aspects of computational linguistics, natural language processing and information retrieval: the *Responsa* project [30,31,52]. Algorithms were developed for automatic generation of all the possible inflected and derived forms of all the bases in Hebrew, including those obtained by the combination of prepositions, conjunctions, articles etc. Based on the generation algorithm, a file was created which included all the possible Hebrew word forms, approximately 2,500,000 words. The analyzer implements a program which strips the possible affixes off the input word and checks whether the obtained result is indeed a legal word. Thus, morphological analysis and generation are incorporated in a complete system for computational processing of Hebrew (albeit not Modern, contemporary Hebrew). A more modern implementation of this system was later commercialized [32, 33].

A different approach to Hebrew morphology is based on the *Phonemic Script* [105], which is an unambiguous writing system for Hebrew, preserving the deep structure of the words. Based on this script, a wide variety of programs were developed, including a program for vocalization [104], a program for the preparation of concordances and indexes [103], especially developed for a database of legal texts [106], a series of programs for morphological analysis and generation [60, 108, 120, 121] and programs for converting phonemic script to the standard Hebrew script [107].

Morphological analysis is one aspect of a commercial system, *Context*, designed for information retrieval [110]. Another commercial system, *Avgad* [24], is based on a dictionary of 25,000 entries, which form the base for "hundreds of thousands" of Hebrew words (including inflected forms). It was used by Segal (Morphological analyzer for unvocalized Hebrew words, http://www.cs.technion.ac.il/~erelsgl/hmntx.zip, unpublished work, 1997) in order to construct a freely available morphological analyzer: the analyzer was built by automatically generating possible base forms, inflecting them in all possible ways and verifying the results against the existing analyzer.

The current state of the art in Hebrew morphological analysis is based on the *HAMSAH* morphological grammar [138], which is implemented in XFST (Sect. 2.4.3). This grammar was reimplemented in Java (for the rationale behind the reimplementation, see [136]) and is currently being maintained and distributed by the Knowledge Center for Processing Hebrew [80]. The coverage of the system is high, and it is constantly being tested on new documents, in order to extend its lexicon as needed.

It is worth mentioning here that a different morphological grammar was developed for Hebrew, focusing on transcribed *spoken* interactions of children

[2]This section is adapted from [135].

and adults [102]. In the context of the CHILDES project [95], corpora of such interactions are being developed for dozens of languages, many of which are also accompanied by morphological annotations. Nir et al. [102] describe such a corpus, transcribed in a way that reflects not only the consonantal distinctions that the standard Hebrew script makes, but also vocalic distinctions that it does not, including the location of the main stress on each word. This transcription makes the morphological grammar harder to develop, but it results in a very low degree of ambiguity. The grammar now has full lexical and rule coverage of the two corpora it is applied to, and more corpora are expected to be analyzed in the near future.

2.5.4 Other Languages

Morphological resources for other Semitic languages are almost nonexistent. A few notable exceptions include Biblical Hebrew, for which morphological analyzers are available from several commercial enterprises; Akkadian, for which some morphological analyzers were developed [16, 85, 94]; Syriac, which inspired the development of a new model of computational morphology [88]; and dialectal Arabic [44, 65, 68, 72].

2.5.5 Related Applications

Also worth mentioning here are a few works that address other morphology-related tasks. These include a shallow morphological analyzer for Arabic [39] that basically segments word forms to sequences of (at most one) prefix, a stem and (at most one) suffix; a system for identifying the roots of Hebrew and Arabic (possibly inflected) words [40]; various programs for vocalization, or restoring diacritics, in Arabic [66, 97, 100, 118, 139] and in other Semitic languages [73]; determining case endings of Arabic words [69]; and correction of optical character recognizer (OCR) errors [96].

When downstream applications are considered, such as chunking, parsing, or machine translation, the question of *tokenization* gains much importance. Morphological analysis determines the lexeme and the inflections (and, sometimes, also the derivational) morphemes of a surface form; but the way in which a surface form is broken down to its morphemes for the purpose of further processing can have a significant impact on the accuracy of such applications. For example, it is convenient to assume that Arabic and Hebrew prefixes are separate tokens; but what about suffixes? Should there be a distinction between the plural suffixes and the pronominal enclitics of nouns? Several works address these questions, usually in the context of a specific application.

Several works investigate various pre-processing techniques for Arabic, in the context of developing Arabic-to-English statistical machine translation systems [45, 46, 67, 116]. In the reverse direction, [13] and [2] explore the impact of morphological segmentation on English-to-Arabic machine translation. The effect of multiple pre-processing schemes on statistical word alignment for machine translation is explored by Elming and Habash [47]. And Diab [41] investigates the effect of differently defined POS tagsets (more or less refined) on the task of base phrase chunking (shallow parsing).

2.6 Morphological Disambiguation of Semitic Languages

Early attempts at POS tagging and morphological disambiguation of Semitic languages relied on more "traditional" approaches, borrowed directly from the general (i.e., English) POS tagging literature. The first such work is probably [87], who defined a set of 131 POS tags, manually annotated a corpus of 50,000 words and then implemented a tagger that combines statistical and rule-based techniques that performs both segmentation and tag disambiguation. Similarly, [42] use SVM to automatically tokenize, POS-tag, and chunk Arabic texts. To this end, they use a reduced tag set of only 24 tags, with which the reported results are very high. The set of tags is extended to 75 in [41].

For Hebrew, two HMM-based POS taggers were developed. The tagger of [14] is trained on an annotated corpus [80]. The most updated version of the tagger, trained on a treebank of 4,500 sentences, boasts 97.2 % accuracy for segmentation (detection of underlying morphemes, including a possibly assimilated definite article), and 90.8 % accuracy for POS tagging [15]. Adler and Elhadad [1] train an HMM-based POS tagger on a large-scale *un*annotated corpus of 6 million words, the reported accuracy being 92.32 % for POS tagging and 88.5 % for full morphological disambiguation, including finding the correct lexical entry.

As for Amharic, [48] uses condition random fields for POS tagging. As the annotated corpus used for training is extremely small (1,000 words), it is not surprising the accuracy is rather low: 84 % for segmentation, and only 74 % for POS tagging. Two other works use a recently-created 210,000-word annotated corpus [54] to train Amharic POS taggers with a tag set of size 30. Gambäck et al. [55] experiment with HMM, SVM and Maximum Entropy; accuracy ranges between 88 and 95 %, depending on the test corpus. Similarly, [129] investigate various classification techniques, using the same corpus for the same task. The best accuracy, achieved with SVM, is over 86 %, but other classification methods, including conditional random fields and memory-based learning, perform well.

The challenge of morphological disambiguation in Semitic languages, however, as discussed in Sect. 2.3 above, prompted several novel approaches to the task. Many of them are based on the work of [75] and [74], who describe morphological disambiguation of Czech, Estonian, Hungarian, Romanian and Slovene within a single approach. The main idea is to define separate classifiers for each feature of the morphological analysis (e.g., POS, number, person, tense, case, etc.) The predictions of all the classifiers are then combined with a weighted log-linear model to produce a single, unified analysis. If a morphological analyzer for the language is available, its output is used to constrain the possible analyses predicted by the classifiers.

This approach exactly has been adapted to Arabic by Habash and Rambow [64] and to Hebrew by Shacham and Wintner [119]. Habash and Rambow [64] start with the output of a morphological analyzer [27]. They define ten classifiers, one for each feature of the morphological analysis, namely POS, gender, number, person, voice, aspect, a pronominal enclitic and two classifiers for conjunction and particle proclitics. The classifiers are implemented with SVM, using all the features of the morphological analyses of words within a ± 2 window of the target word as features. The predictions of the ten classifiers are combined to yield the most likely analysis for each word. The best results are achieved by a rule-based classifier, learned from the training data, that decides when an analysis is "good" based on the predictions of the basic classifiers. The state of the art in Arabic morphological disambiguation is represented by the MADA + TOKAN system [70, 115], which implements these ideas.

Shacham and Wintner [119] basically adapt this approach to Hebrew. They define classifiers for POS, gender, number, person, tense, definiteness, status, prefixes and suffixes, implemented with SNoW [114], using all the features of the morphological analysis in a varying window around the target word as features. They, too, investigate a few methods for combining the results of the classifiers, but the naïve, unweighted combination yields the best results.

A different approach is proposed by Smith et al. [124]. While they also use a morphological analyzer (in the case of Arabic, [27]) to constrain the possible analyses, prediction is done in the source–channel model, where the source is a factored, conditionally-estimated random field [90]. The model is applied to Arabic (and also to Czech and Korean), and the results are competitive with [64] (the same tag set of 139 tags is used).

Recently, [98, 99] proposed an alternative approach to POS tagging of Arabic, which they refer to as *full-word tagging*. Given a large annotated corpus of some 500,000 words, they observe that almost 1,000 different (complex) tags occur in the corpus. They use Memory-based Learning [38] to train a classifier to assign any one of those tags. This is a difficult task: almost one quarter of the tags occur only once in the corpus, so data sparseness is a serious issue. On the other hand, the ambiguity of full word forms is low: only 1.1 analyses per word, on average, with a maximum of 7. This approach results in more accurate disambiguation than any other approach. Furthermore, projecting the complex POS tags to simpler ones, in this case the extra-reduced tagset of [64], results in more accurate "basic" POS tagging than a direct approach that predicts the simpler tags only. Most interestingly, the results show that running either a segmentation classifier or a vocalization one as a pre-process does not improve the accuracy of morphological disambiguation.

Again, the reader is referred to [63] for a full discussion of Arabic morphological disambiguation. For other Semitic languages than the ones described here, unfortunately, we are unaware of any works addressing morphological disambiguation.

2.7 Future Directions

The discussion above establishes the inherent difficulty of morphological processing with Semitic languages, as one instance of languages with rich and complex morphology. Having said that, it is clear that with a focused effort, contemporary computational technology is sufficient for tackling the difficulties. As should be clear from Sect. 2.5, the two Semitic languages that benefitted from most attention, namely MSA and Hebrew, boast excellent computational morphological analyzers and generators. Similarly, Sect. 2.6 shows that morphological disambiguation of these two languages can be done with high accuracy, nearing the accuracy of disambiguation with European languages.

However, for the less-studied languages, including Amharic, Maltese and others, much work is still needed in order to produce tools of similar precision. Resembling the situation in Arabic and Hebrew, this effort should focus on two fronts: development of formal, computationally-implementable sets of rules that describe the morphology of the language in question; and collection and annotation of corpora from which morphological disambiguation modules can be trained.

As for future technological improvements, we note that "pipeline" approaches, whereby the input text is fed, in sequence, to a tokenizer, a morphological analyzer, a morphological disambiguation module and then a parser, have probably reached a ceiling, and the stage is ripe for more elaborate, unified approaches. Several works indeed explore such possibilities, focusing in particular on joint morphological disambiguation and parsing [35, 59, 92, 132]. We defer an extensive discussion of these (and other) approaches to the next Chapter on parsing.

Acknowledgements I am tremendously grateful to Nizar Habash for his help and advice; it would have been hard to complete this chapter without them. All errors and misconceptions are, of course, solely my own.

References

1. Adler M, Elhadad M (2006) An unsupervised morpheme-based HMM for Hebrew morphological disambiguation. In: Proceedings of the 21st international conference on computational linguistics and 44th annual meeting of the Association for Computational Linguistics, Sydney. Association for Computational Linguistics, pp 665–672. http://www.aclweb.org/anthology/P/P06/P06-1084
2. Al-Haj H, Lavie A (2010) The impact of Arabic morphological segmentation on broad-coverage English-to-Arabic statistical machine translation. In: Proceedings of the conference of the Association for Machine Translation in the Americas (AMTA), Denver
3. Alkuhlani S, Habash N (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: human language technologies, Portland. Association for Computational Linguistics, pp 357–362. http://www.aclweb.org/anthology/P11-2062

4. Al-Shalabi R, Evens M (1998) A computational morphology system for Arabic. In: Rosner M (ed) Proceedings of the workshop on computational approaches to Semitic languages, COLING-ACL'98, Montreal, pp 66–72
5. Al-Sughaiyer IA, Al-Kharashi IA (2004) Arabic morphological analysis techniques: a comprehensive survey. J Am Soc Inf Sci Technol 55(3):189–213
6. Altantawy M, Habash N, Rambow O, Saleh I (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
7. Altantawy M, Habash N, Rambow O (2011) Fast yet rich morphological analysis. In: Proceedings of the 9th international workshop on finite-state methods and natural language processing (FSMNLP 2011), Blois
8. Amsalu S, Gibbon D (2005) A complete finite-state model for Amharic morphographemics. In: Yli-Jyrä A, Karttunen L, Karhumäki J (eds) FSMNLP. Lecture notes in computer science, vol 4002. Springer, Berlin/New York, pp 283–284
9. Amsalu S, Gibbon D (2005) Finite state morphology of Amharic. In: Proceedings of RANLP, Borovets, pp 47–51
10. Amtrup JW (2003) Morphology in machine translation systems: efficient integration of finite state transducers and feature structure descriptions. Mach Transl 18(3):217–238. doi:http://dx.doi.org/10.1007/s10590-004-2476-5
11. Argaw AA, Asker L (2007) An Amharic stemmer: reducing words to their citation forms. In: Proceedings of the ACL-2007 workshop on computational approaches to Semitic languages, Prague
12. Audebert C, Gaubert C, Jaccarini A (2009) Minimal resources for Arabic parsing: an interactive method for the construction of evolutive automata. In: Choukri K, Maegaard B (eds) Proceedings of the second international conference on Arabic language resources and tools, The MEDAR Consortium, Cairo
13. Badr I, Zbib R, Glass J (2008) Segmentation for English-to-Arabic statistical machine translation. In: Proceedings of ACL-08: HLT, short papers, Columbus. Association for Computational Linguistics, pp 153–156. http://www.aclweb.org/anthology/P/P08/P08-2039
14. Bar-Haim R, Sima'an K, Winter Y (2005) Choosing an optimal architecture for segmentation and POS-tagging of Modern Hebrew. In: Proceedings of the ACL workshop on computational approaches to Semitic languages, Ann Arbor. Association for Computational Linguistics, pp 39–46, http://www.aclweb.org/anthology/W/W05/W05-0706
15. Bar-haim R, Sima'an K, Winter Y (2008) Part-of-speech tagging of Modern Hebrew text. Nat Lang Eng 14(2):223–251
16. Barthélemy F (1998) A morphological analyzer for Akkadian verbal forms with a model of phonetic transformations. In: Proceedings of the Coling-ACL 1998 workshop on computational approaches to Semitic languages, Montreal, pp 73–81
17. Beesley KR (1996) Arabic finite-state morphological analysis and generation. In: Proceedings of COLING-96, the 16th international conference on computational linguistics, Copenhagen
18. Beesley KR (1998) Arabic morphological analysis on the internet. In: Proceedings of the 6th international conference and exhibition on multi-lingual computing, Cambridge
19. Beesley KR (1998) Arabic morphology using only finite-state operations. In: Rosner M (ed) Proceedings of the workshop on computational approaches to Semitic languages, COLING-ACL'98, Montreal, pp 50–57
20. Beesley KR (1998) Constraining separated morphotactic dependencies in finite-state grammars. In: FSMNLP-98, Bilkent, pp 118–127
21. Beesley KR, Karttunen L (2000) Finite-state non-concatenative morphotactics. In: Proceedings of the fifth workshop of the ACL special interest group in computational phonology, SIGPHON-2000, Luxembourg
22. Beesley KR, Karttunen L (2003) Finite-state morphology: xerox tools and techniques. CSLI, Stanford
23. Belguith LH, Aloulou C, Ben Hamadou A (2008) MASPAR: De la segmentation à l'analyse syntaxique de textes arabes. Rev Inf Interact Intell I3 7(2):9–36

24. Bentur E, Angel A, Segev D (1992) Computerized analysis of Hebrew words. Hebrew Linguist 36:33–38. (in Hebrew)
25. Berri J, Zidoum H, Atif Y (2001) Web-based Arabic morphological analyzer. In: Gelbukh A (ed) CICLing 2001. Lecture notes in computer science, vol 2004. Springer, Berlin, pp 389–400
26. Brants T (2000) TnT: a statistical part-of-speech tagger. In: Proceedings of the sixth conference on applied natural language processing, Seattle. Association for Computational Linguistics, pp 224–231. doi:10.3115/974147.974178, http://www.aclweb.org/anthology/A00-1031
27. Buckwalter T (2004) Buckwalter Arabic morphological analyzer version 2.0. Linguistic Data Consortium, Philadelphia
28. Buckwalter T (2004) Issues in Arabic orthography and morphology analysis. In: Farghaly A, Megerdoomian K (eds) COLING 2004 computational approaches to Arabic script-based languages, COLING, Geneva, pp 31–34
29. Choueka Y (1966) Computers and grammar: mechnical analysis of Hebrew verbs. In: Proceedings of the annual conference of the Israeli Association for Information Processing, Rehovot, pp 49–66. (in Hebrew)
30. Choueka Y (1972) Fast searching and retrieval techniques for large dictionaries and concordances. Heb Comput Linguist 6:12–32. (in Hebrew)
31. Choueka Y (1980) Computerized full-text retrieval systems and research in the humanities: the Responsa project. Comput Humanit 14:153–169
32. Choueka Y (1990) MLIM – a system for full, exact, on-line grammatical analysis of Modern Hebrew. In: Eizenberg Y (ed) Proceedings of the annual conference on computers in education, Tel Aviv, p 63. (in Hebrew)
33. Choueka Y (1993) Response to "computerized analysis of Hebrew words". Heb Linguist 37:87. (in Hebrew)
34. Cohen D (1970) Essai d'une analyse automatique de l'arabe. In: Etudes de linguistique sémitique et arabe, De Gruyter, Germany, pp 49–78
35. Cohen SB, Smith NA (2007) Joint morphological and syntactic disambiguation. In: Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL), Prague. Association for Computational Linguistics, pp 208–217. http://www.aclweb.org/anthology/D/D07/D07-1022
36. Cohen-Sygal Y, Wintner S (2006) Finite-state registered automata for non-concatenative morphology. Comput Linguist 32(1):49–82
37. Collins M (2002) Discriminative training methods for hidden markov models: theory and experiments with perceptron algorithms. In: Proceedings of the ACL-02 conference on empirical methods in natural language processing, EMNLP '02, Philadelphia, Vol 10. Association for Computational Linguistics, pp 1–8. doi:http://dx.doi.org/10.3115/1118693.1118694
38. Daelemans W, van den Bosch A (2005) Memory-based language processing. Studies in natural language processing. Cambridge University Press, Cambridge
39. Darwish K (2002) Building a shallow Arabic morphological analyzer in one day. In: Rosner M, Wintner S (eds) ACL'02 workshop on computational approaches to Semitic languages , Philadelphia, pp 47–54
40. Daya E, Roth D, Wintner S (2007) Learning to identify Semitic roots. In: Soudi A, Neumann G, van den Bosch A (eds) Arabic computational morphology: knowledge-based and empirical methods, text, speech and language technology, vol 38. Springer, Dordrecht, pp 143–158
41. Diab M (2007) Improved Arabic base phrase chunking with a new enriched POS tag set. In: Proceedings of the 2007 workshop on computational approaches to Semitic languages: common issues and resources, Prague, pp 89–96. http://www.aclweb.org/anthology/W/W07/W07-0812
42. Diab M, Hacioglu K, Jurafsky D (2004) Automatic tagging of Arabic text: from raw text to base phrase chunks. In: Proceedings of HLT-NAACL 2004, Boston

43. Dichy J, Farghaly A (2003) Roots and patterns vs. stems plus grammar-lexis specifications: on what basis should a multilingual lexical database centered on Arabic be built. In: Proceedings of the MT-Summit IX workshop on machine translation for Semitic languages, New Orleans, pp 1–8
44. Duh K, Kirchhoff K (2005) POS tagging of dialectal Arabic: a minimally supervised approach. In: Proceedings of the ACL workshop on computational approaches to Semitic languages, Ann Arbor. Association for Computational Linguistics, pp 55–62. http://www.aclweb.org/anthology/W/W05/W05-0708
45. El Kholy A, Habash N (2010) Orthographic and morphological processing for English-Arabic statistical machine translation. In: In actes de traitement automatique des langues naturelles (TALN), Montréal
46. El Kholy A, Habash N (2010) Techniques for Arabic morphological detokenization and orthographic denormalization. In: Proceedings of LREC-2010, Valletta (Malta)
47. Elming J, Habash N (2007) Combination of statistical word alignments based on multiple preprocessing schemes. In: Human language technologies 2007: the conference of the North American chapter of the Association for Computational Linguistics, Companion Volume, Short Papers, Prague, pp 25–28. http://www.aclweb.org/anthology/N/N07/N07-2007
48. Fissaha Adafre S (2005) Part of speech tagging for Amharic using conditional random fields. In: Proceedings of the ACL workshop on computational approaches to Semitic languages, Ann Arbor. Association for Computational Linguistics, pp 47–54. http://www.aclweb.org/anthology/W/W05/W05-0707
49. Fissaha S, Haller J (2003) Amharic verb lexicon in the context of machine translation. In: Proceedings of the TALN workshop on natural language processing of minority languages, Batz-sur-Mer
50. Forsberg M (2007) Three tools for language processing: BNF converter, functional morphology, and extract. PhD thesis, Göteborg University and Chalmers University of Technology
51. Forsberg M, Ranta A (2004) Functional morphology. In: Proceedings of the ninth ACM SIGPLAN international conference on functional programming (ICFP'04), Snowbird. ACM, New York, pp 213–223
52. Fraenkel AS (1976) All about the Responsa retrieval project – what you always wanted to know but were afraid to ask. Jurimetrics J 16(3):149–156
53. Gadish R (ed) (2001) Klalei ha-Ktiv Hasar ha-Niqqud, 4th edn. Academy for the Hebrew Language, Jerusalem. (in Hebrew)
54. Gambäck B, Olsson F, Argaw AA, Asker L (2009) An Amharic corpus for machine learning. In: Proceedings of the 6th world congress of African linguistics, Cologne
55. Gambäck B, Olsson F, Argaw AA, Asker L (2009) Methods for Amharic part-of-speech tagging. In: Proceedings of the first workshop on language technologies for African languages, Athen. Association for Computational Linguistics, Stroudsburg, pp 104–111
56. Gasser M (2009) Semitic morphological analysis and generation using finite state transducers with feature structures. In: Proceedings of the 12th conference of the European chapter of the ACL (EACL 2009), Athens. Association for Computational Linguistics, pp 309–317. http://www.aclweb.org/anthology/E09-1036
57. Gasser M (2011) HornMorpho: a system for morphological processing of Amharic, Oromo, and Tigrinya, Bibliotheca Alexandrina, Alexandria, pp 94–99
58. Giménez J, Màrquez L (2004) SVMTool: a general POS tagger generator based on support vector machines. In: Proceedings of 4th international conference on language resources and evaluation (LREC), Lisbon, pp 43–46
59. Goldberg Y, Tsarfaty R (2008) A single generative model for joint morphological segmentation and syntactic parsing. In: Proceedings of ACL-08: HLT, Columbus. Association for Computational Linguistics, pp 371–379. http://www.aclweb.org/anthology/P/P08/P08-1043
60. Goldstein L (1991) Generation and inflection of the possession inflection of Hebrew nouns. Master's thesis, Technion, Haifa (in Hebrew)
61. Habash N (2004) Large scale lexeme based arabic morphological generation. In: Proceedings of traitement automatique du langage naturel (TALN-04), Fez

62. Habash N (2007) Arabic morphological representations for machine translation. In: van den Bosch A, Soudi A (eds) Arabic computational morphology: knowledge-based and empirical methods. Springer, Dordrecht
63. Habash N (2010) Introduction to Arabic natural language processing. Synthesis lectures on human language technologies. Morgan & Claypool, San Rafael. doi:http://dx.doi.org/10.2200/S00277ED1V01Y201008HLT010
64. Habash N, Rambow O (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), University of Michigan. Association for Computational Linguistics, Ann Arbor, pp 573–580. http://www.aclweb.org/anthology/P/P05/P05-1071
65. Habash N, Rambow O (2006) MAGEAD: a morphological analyzer and generator for the Arabic dialects. In: Proceedings of the 21st international conference on computational linguistics and 44th annual meeting of the Association for Computational Linguistics, Sydney. Association for Computational Linguistics, pp 681–688. http://www.aclweb.org/anthology/P/P06/P06-1086
66. Habash N, Rambow O (2007) Arabic diacritization through full morphological tagging. In: Human language technologies 2007: the conference of the North American chapter of the association for computational linguistics; Companion Volume, Short Papers, Rochester. Association for Computational Linguistics, pp 53–56. http://www.aclweb.org/anthology/N/N07/N07-2014
67. Habash N, Sadat F (2006) Arabic preprocessing schemes for statistical machine translation. In: Moore RC, Bilmes JA, Chu-Carroll J, Sanderson M (eds) HLT-NAACL, New York. The Association for Computational Linguistics
68. Habash N, Rambow O, Kiraz G (2005) Morphological analysis and generation for Arabic dialects. In: Proceedings of the ACL workshop on computational approaches to Semitic languages, Ann Arbor. Association for Computational Linguistics, pp 17–24. http://www.aclweb.org/anthology/W/W05/W05-0703
69. Habash N, Gabbard R, Rambow O, Kulick S, Marcus M (2007) Determining case in Arabic: learning complex linguistic behavior requires complex linguistic features. In: Proceeings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL 2007), Prague
70. Habash N, Rambow O, Roth R (2009) MADA+TOKAN: a toolkit for Arabic tokenization, diacritization, morphological disambiguation, POS tagging, stemming and lemmatization. In: Choukri K, Maegaard B (eds) Proceedings of the second international conference on Arabic language resources and tools, Cairo, The MEDAR Consortium
71. Habash N, Diab M, Rabmow O (2012) Conventional orthography for Dialectal Arabic. In: Proceedings of the language resources and evaluation conference (LREC), Istanbul
72. Habash N, Eskander R, Hawwari A (2012) A morphological analyzer for Egyptian Arabic. In: Proceedings of the twelfth meeting of the special interest group on computational morphology and phonology, Montréal. Association for Computational Linguistics, pp 1–9. http://www.aclweb.org/anthology/W12-2301
73. Haertel RA, McClanahan P, Ringger EK (2010) Automatic diacritization for low-resource languages using a hybrid word and consonant CMM. In: Human language technologies: the 2010 annual conference of the north american chapter of the Association for Computational Linguistics, HLT '10, Stroudsburg. Association for Computational Linguistics, pp 519–527
74. Hajič J (2000) Morphological tagging: Data vs. dictionaries. In: Proceedings of ANLP-NAACL conference, Seattle, pp 94–101
75. Hajič J, Hladká B (1998) Tagging inflective languages: prediction of morphological categories for a rich, structured tagset. In: Proceedings of the 36th annual meeting of the Association for Computational Linguistics and 17th international conference on computational linguistics, Montreal. Association for Computational Linguistics, Stroudsburg, pp 483–490. doi:http://dx.doi.org/10.3115/980845.980927, http://dx.doi.org/10.3115/980845.980927

76. Harley HB (2006) English words: a linguistic introduction. The language library. Wiley-Blackwell, Malden
77. Hetzron R (ed) (1997) The Semitic languages. Routledge, London/New York
78. Hulden M (2009) Foma: a finite-state compiler and library. In: Proceedings of the demonstrations session at EACL 2009, Athens. Association for Computational Linguistics, pp 29–32. http://www.aclweb.org/anthology/E09-2008
79. Hulden M (2009) Revisiting multi-tape automata for Semitic morphological analysis and generation. In: Proceedings of the EACL 2009 workshop on computational approaches to Semitic languages, Athens. Association for Computational Linguistics, pp 19–26. http://www.aclweb.org/anthology/W09-0803
80. Itai A, Wintner S (2008) Language resources for Hebrew. Lang Resour Eval 42(1):75–98
81. Johnson CD (1972) Formal aspects of phonological description. Mouton, The Hague
82. Kammoun NC, Belguith LH, Mesfar S (2010) Arabic POS tagging based on NooJ grammars and the Arabic morphological analyzer MORPH2. In: Proceedings of NooJ 2010, Komotini
83. Kaplan RM, Kay M (1994) Regular models of phonological rule systems. Comput Linguist 20(3):331–378
84. Karttunen L, Beesley KR (2001) A short history of two-level morphology. In: Talk given at the ESSLLI workshop on finite state methods in natural language processing. http://www.helsinki.fi/esslli/evening/20years/twol-history.html
85. Kataja L, Koskenniemi K (1988) Finite-state description of Semitic morphology: a case study of ancient Akkadian. In: COLING, Budapest, pp 313–315
86. Kay M (1987) Nonconcatenative finite-state morphology. In: Proceedings of the third conference of the European chapter of the Association for Computational Linguistics, Copenhagen, pp 2–10
87. Khoja S (2001) APT: Arabic part-of-speech tagger. In: Proceedings of the student workshop at the second meeting of the North American chapter of the Association for Computational Linguistics (NAACL2001), Pittsburgh
88. Kiraz GA (2000) Multitiered nonlinear morphology using multitape finite automata: a case study on Syriac and Arabic. Comput Linguist 26(1):77–105
89. Koskenniemi K (1983) Two-level morphology: a general computational model for word-form recognition and production. The Department of General Linguistics, University of Helsinki
90. Lafferty J, McCallum A, Pereira F (2001) Conditional random fields: probabilistic models for segmenting and labeling sequence data. In: Proceedings of the 18th international conference on machine learning (ICML-01), Williamstown, pp 282–289
91. Lavie A, Itai A, Ornan U, Rimon M (1988) On the applicability of two-level morphology to the inflection of Hebrew verbs. In: Proceedings of the international conference of the ALLC, Jerusalem
92. Lee J, Naradowsky J, Smith DA (2011) A discriminative model for joint morphological disambiguation and dependency parsing. In: Proceedings of the 49th annual meeting of the Association for Computational Linguistics: human language technologies, Portland. Association for Computational Linguistics, pp 885–894. http://www.aclweb.org/anthology/P11-1089
93. Maamouri M, Bies A, Buckwalter T, Mekki W (2004) The Penn Arabic treebank: building a large-scale annotated Arabic corpus. In: NEMLAR conference on Arabic language resources and tools, Cairo, pp 102–109
94. Macks A (2002) Parsing Akkadian verbs with Prolog. In: Proceedings of the ACL-02 workshop on computational approaches to Semitic languages, Philadelphia
95. MacWhinney B (2000) The CHILDES project: tools for analyzing talk, 3rd edn. Lawrence Erlbaum Associates, Mahwah
96. Magdy W, Darwish K (2006) Arabic OCR error correction using character segment correction, language modeling, and shallow morphology. In: Proceedings of the 2006 conference on empirical methods in natural language processing, Sydney. Association for Computational Linguistics, pp 408–414. http://www.aclweb.org/anthology/W/W06/W06-1648

97. Mohamed E, Kübler S (2009) Diacritization for real-world Arabic texts. In: Proceedings of the international conference RANLP-2009, pp 251–257. http://www.aclweb.org/anthology/R09-1047
98. Mohamed E, Kübler S (2010) Arabic part of speech tagging. In: Proceedings of the seventh conference on international language resources and evaluation (LREC'10), European Language Resources Association (ELRA), Valletta
99. Mohamed E, Kübler S (2010) Is Arabic part of speech tagging feasible without word segmentation? In: Human language technologies: the 2010 annual conference of the North American chapter of the Association for Computational Linguistics, HLT'10, Los Angeles. Association for Computational Linguistics, Stroudsburg, pp 705–708. http://dl.acm.org/citation.cfm?id=1857999.1858104
100. Nelken R, Shieber SM (2005) Arabic diacritization using weighted finite-state transducers. In: Proceedings of the ACL workshop on computational approaches to Semitic languages, Ann Arbor. Association for Computational Linguistics, pp 79–86. http://www.aclweb.org/anthology/W/W05/W05-0711
101. Netzer Y, Adler M, Gabay D, Elhadad M (2007) Can you tag the modal? You should. In: Proceedings of the ACL-2007 workshop on computational approaches to Semitic languages, Prague
102. Nir B, MacWhinney B, Wintner S (2010) A morphologically-analyzed CHILDES corpus of Hebrew. In: Proceedings of the seventh conference on international language resources and evaluation (LREC'10), Valletta. European Language Resources Association (ELRA), pp 1487–1490
103. Ornan U (1985) Indexes and concordances in a phonemic Hebrew script. In: Proceedings of the ninth world congress of Jewish studies, World Union of Jewish Studies, Jerusalem, pp 101–108. (in Hebrew)
104. Ornan U (1985) Vocalization by a computer: a linguistic lesson. In: Luria BZ (ed) Avraham Even-Shoshan book, Kiryat-Sefer, Jerusalem, pp 67–76. (in Hebrew)
105. Ornan U (1986) Phonemic script: a central vehicle for processing natural language – the case of Hebrew. Technical report 88.181, IBM Research Center, Haifa
106. Ornan U (1987) Computer processing of Hebrew texts based on an unambiguous script. Mishpatim 17(2):15–24. (in Hebrew)
107. Ornan U, Katz M (1995) A new program for Hebrew index based on the Phonemic Script. Technical report LCL 94-7, Laboratory for Computational Linguistics, Technion, Haifa
108. Ornan U, Kazatski W (1986) Analysis and synthesis processes in Hebrew morphology. In: Proceedings of the 21 national data processing conference, Israel. (in Hebrew)
109. Owens J (1997) The Arabic grammatical tradition. In: Hetzron R (ed) The Semitic languages. Routledge, London/New York, chap 3, pp 46–58
110. Pinkas G (1985) A linguistic system for information retrieval. Maase Hoshev 12:10–16. (in Hebrew)
111. Ratnaparkhi A (1996) A maximum entropy model for part-of-speech tagging. In: Brill E, Church K (eds) Proceedings of the conference on empirical methods in natural language processing, Copenhagen. Association for Computational Linguistics, pp 133–142
112. Roark B, Sproat RW (2007) Computational approaches to morphology and syntax. Oxford University Press, New York
113. Roche E, Schabes Y (eds) (1997) Finite-state language processing. Language, speech and communication. MIT, Cambridge
114. Roth D (1998) Learning to resolve natural language ambiguities: a unified approach. In: Proceedings of AAAI-98 and IAAI-98, Madison, pp 806–813
115. Roth R, Rambow O, Habash N, Diab M, Rudin C (2008) Arabic morphological tagging, diacritization, and lemmatization using lexeme models and feature ranking. In: Proceedings of ACL-08: HLT, Short Papers, Columbus. Association for Computational Linguistics, pp 117–120. http://www.aclweb.org/anthology/P/P08/P08-2030

116. Sadat F, Habash N (2006) Combination of Arabic preprocessing schemes for statistical machine translation. In: Proceedings of the 21st international conference on computational linguistics and 44th annual meeting of the Association for Computational Linguistics, Sydney. Association for Computational Linguistics, pp 1–8. http://www.aclweb.org/anthology/P/P06/P06-1001
117. Schippers A (1997) The Hebrew grammatical tradition. In: Hetzron R (ed) The Semitic languages. Routledge, London/New York, chap 4, pp 59–65
118. Shaalan K, Abo Bakr HM, Ziedan I (2009) A hybrid approach for building Arabic diacritizer. In: Proceedings of the EACL 2009 workshop on computational approaches to Semitic languages, Semitic'09, Athens. Association for Computational Linguistics, Stroudsburg, pp 27–35
119. Shacham D, Wintner S (2007) Morphological disambiguation of Hebrew: a case study in classifier combination. In: Proceedings of EMNLP-CoNLL 2007, the conference on empirical methods in natural language processing and the conference on computational natural language learning, Prague. Association for Computational Linguistics
120. Shany-Klein M (1990) Generation and analysis of Segolate noun inflection in Hebrew. Master's thesis, Technion, Haifa. (in Hebrew)
121. Shany-Klein M, Ornan U (1992) Analysis and generation of Hebrew Segolate nouns. In: Ornan U, Arieli G, Doron E (eds) Hebrew computational linguistics. Ministry of Science and Technology, Jerusalem, chap 4, pp 39–51. (in Hebrew)
122. Shapira M, Choueka Y (1964) Mechanographic analysis of Hebrew morphology: possibilities and achievements. Leshonenu 28(4):354–372. (in Hebrew)
123. Silberztein M (2004) NooJ: an object-oriented approach. In: Muller C, Royauté J, Silberztein M (eds) INTEX pour la linguistique et le traitement automatique des Langues, cahiers de la MSH Ledoux, Presses Universitaires de Franche-Comté, pp 359–369
124. Smith NA, Smith DA, Tromble RW (2005) Context-based morphological disambiguation with random fields. In: Proceedings of human language technology conference and conference on empirical methods in natural language processing, Vancouver. Association for Computational Linguistics, Morristown, pp 475–482
125. Smrž O (2007) ElixirFM: implementation of functional Arabic morphology. In: Proceedings of the 2007 workshop on computational approaches to Semitic languages: common issues and resources, Prague. Association for Computational Linguistics, Stroudsburg, pp 1–8
126. Smrž O (2007) Functional Arabic morphology. Prague Bull Math Linguist 88:5–30
127. Soudi A, van den Bosch A, Neumann G (2007) Arabic computational morphology: knowledge-based and empirical methods. Springer, Dordrecht
128. Sproat RW (1992) Morphology and computation. MIT, Cambridge
129. Tachbelie MY, Abate ST, Besacier L (2011) Part-of-speech tagging for under-resourced and morphologically rich languages – the case of Amharic, Bibliotheca Alexandrina, Alexandria, pp 50–55. http://aflat.org/files/HLTD201109.pdf
130. Toutanova K, Manning CD (2000) Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In: Proceedings of the 2000 joint SIGDAT conference on empirical methods in natural language processing and very large corpora, Morristown. Association for Computational Linguistics, pp 63–70. doi:http://dx.doi.org/10.3115/1117794.1117802
131. Toutanova K, Klein D, Manning CD, Singer Y (2003) Feature-rich part-of-speech tagging with a cyclic dependency network. In: NAACL '03: Proceedings of the 2003 conference of the North American chapter of the Association for Computational Linguistics on human language technology, Edmonton. Association for Computational Linguistics, Morristown, pp 173–180. doi:http://dx.doi.org/10.3115/1073445.1073478
132. Tsarfaty R (2006) Integrated morphological and syntactic disambiguation for Modern Hebrew. In: Proceedings of the COLING/ACL 2006 student research workshop, Sydney. Association for Computational Linguistics, pp 49–54. http://www.aclweb.org/anthology/P/P06/P06-3009

133. Tsuruoka Y, Tsujii J (2005) Bidirectional inference with the easiest-first strategy for tagging sequence data. In: Proceedings of the conference on human language technology and empirical methods in natural language processing, HLT'05, Vancouver. Association for Computational Linguistics, Stroudsburg, pp 467–474. doi:http://dx.doi.org/10.3115/1220575.1220634, http://dx.doi.org/10.3115/1220575.1220634
134. Tsuruoka Y, Tateishi Y, Kim JD, Ohta T, McNaught J, Ananiadou S, Tsujii J (2005) Developing a robust part-of-speech tagger for biomedical text. In: Bozanis P, Houstis EN (eds) Advances in informatics. LNCS, vol 3746. Springer, Berlin/Heidelberg, chap 36, pp 382–392. doi:10.1007/11573036_36, http://dx.doi.org/10.1007/11573036_36
135. Wintner S (2004) Hebrew computational linguistics: past and future. Artif Intell Rev 21(2):113–138. doi:http://dx.doi.org/10.1023/B:AIRE.0000020865.73561.bc
136. Wintner S (2008) Strengths and weaknesses of finite-state technology: a case study in morphological grammar development. Nat Lang Eng 14(4):457–469. doi:http://dx.doi.org/10.1017/S1351324907004676
137. Wintner S (2009) Language resources for Semitic languages: challenges and solutions. In: Nirenburg S (ed) Language engineering for lesser-studied languages. IOS, Amsterdam, pp 277–290
138. Yona S, Wintner S (2008) A finite-state morphological grammar of Hebrew. Nat Lang Eng 14(2):173–190
139. Zitouni I, Sorensen JS, Sarikaya R (2006) Maximum entropy based restoration of Arabic diacritics. In: Proceedings of the 21st international conference on computational linguistics and 44th annual meeting of the Association for Computational Linguistics, Sydney. Association for Computational Linguistics, pp 577–584. http://www.aclweb.org/anthology/P/P06/P06-1073
140. Zwicky AM, Pullum GK (1983) Cliticization vs. inflection: English *n't*. Language 59(3): 502–513