Adaptor Grammars for Learning Non-Concatenative Morphology

Jan A. Botha and **Phil Blunsom**
Department of Computer Science
University of Oxford
Oxford, OX1 3QD, UK
{jan.botha,phil.blunsom}@cs.ox.ac.uk

Abstract

This paper contributes an approach for expressing non-concatenative morphological phenomena, such as stem derivation in Semitic languages, in terms of a mildly context-sensitive grammar formalism. This offers a convenient level of modelling abstraction while remaining computationally tractable. The nonparametric Bayesian framework of adaptor grammars is extended to this richer grammar formalism to propose a probabilistic model that can learn word segmentation and morpheme lexicons, including ones with discontiguous strings as elements, from unannotated data. Our experiments on Hebrew and three variants of Arabic data find that the additional expressiveness to capture roots and templates as atomic units improves the quality of concatenative segmentation and stem identification. We obtain 74% accuracy in identifying triliteral Hebrew roots, while performing morphological segmentation with an F1-score of 78.1.

1 Introduction

Unsupervised learning of morphology is the task of acquiring, from unannotated data, the intra-word building blocks of a language and the rules by which they combine to form words. This task is of interest both as a gateway for studying language acquisition in humans and as a way of producing morphological analyses that are of practical use in a variety of natural language processing tasks, including machine translation, parsing and information retrieval.

A particularly interesting version of the morphology learning problem comes from languages that use *templatic* morphology, such as Arabic, Hebrew and Amharic. These Semitic languages derive verb and noun stems by interspersing abstract *root* morphemes into *templatic* structures in a non-concatenative way. For example, the Arabic root k·t·b can combine with the template (i-a) to derive the noun stem *kitab* (book). Established morphological analysers typically ignore this process and simply view the derived stems as elementary units (Buckwalter, 2002), or their account of it coincides with a requirement for extensive linguistic knowledge and hand-crafting of rules (Finkel and Stump, 2002; Schneider, 2010; Altantawy et al., 2010). The former approach is bound to suffer from vocabulary coverage issues, while the latter clearly does not transfer easily across languages. The practical appeal of unsupervised learning of templatic morphology is that it can overcome these shortcomings.

Unsupervised learning of *concatenative* morphology has received extensive attention, partly driven by the MorphoChallenge (Kurimo et al., 2010) in recent years, but that is not the case for root-templatic morphology (Hammarström and Borin, 2011).

In this paper we present a model-based method that learns concatenative and root-templatic morphology in a unified framework. We build on two disparate strands of work from the literature: Firstly, we apply *simple Range Concatenating Grammars* (SRCGs) (Boullier, 2000) to parse contiguous and discontiguous morphemes from an input string. These grammars are *mildly-context sensitive* (Joshi, 1985), a superset of context-free grammars that retains polynomial parsing time-complexity. Secondly, we generalise the nonparametric Bayesian learning framework of *adaptor grammars* (Johnson et al., 2007) to SRCGs.[1] This should also be rel-

[1]Our formulation is in terms of SRCGs, which are equivalent in power to linear context-free rewrite systems (Vijay-Shanker et al., 1987) and multiple context-free grammars (Seki et al., 1991), all of which are weaker than (non-simple) range concatenating grammars (Boullier, 2000).

evant to other applications of probabilistic SRCGs, e.g. in parsing (Maier, 2010), translation (Kaeshammer, 2013) and genetics (Kato et al., 2006).

In addition to unannotated data, our method requires as input a minimal set of high-level grammar rules that encode basic intuitions of the morphology. This is where there would be room to become very language specific. Our aim, however, is not to obtain a best-published result in a particular language, but rather to create a method that is applicable across a variety of morphological processes. The specific rules used in our empirical evaluation on Arabic and Hebrew therefore contain hardly any explicit linguistic knowledge about the languages and are applicable across the family of Semitic languages.

2 A powerful grammar for morphology

Concatenative morphology lends itself well to an analysis in terms of finite-state transducers (FSTs) (Koskenniemi, 1984). With some additional effort, FSTs can also encode non-concatenative morphology (Kiraz, 2000; Beesley and Karttunen, 2003; Cohen-Sygal and Wintner, 2006; Gasser, 2009). Despite this seeming adequacy of regular languages to describe morphology, we see two main shortcomings that motivate moving further up the Chomsky hierarchy of formal languages: first is the issue of learning. We are not aware of successful attempts at inducing FST-based morphological analysers in an unsupervised way, and believe the challenge lies in the fact that FSTs do not offer a *convenient* way of expressing prior linguistic intuitions to guide the learning process. Secondly, an FST composed of multiple machines might capture morphological processes well and excel at analysis, but interpretability of its internal operations are limited.

These shortcomings are overcome for concatenative morphology by context-free adaptor grammars, which allowed diverse segmentation models to be formulated and investigated within a single framework (Johnson et al., 2007; Johnson, 2008; Sirts and Goldwater, 2013). In principle, that covers a wide range of phenomena (typical example language in parentheses): affixal inflection (Czech) and derivation (English), agglutinative derivation (Turkish, Finnish), compounding (German). Our agenda here is to extend that approach to include non-concatenative processes such as root-templatic derivation (Arabic), infixation (Tagalog) and circumfixation (Indonesian). In this pursuit, an abstraction that permits discontiguous constituents is a highly useful modelling tool, but requires looking beyond context-free grammars.

An idealised generative grammar that would capture all the aforementioned phenomena could look like this:

$$\text{Word} \rightarrow (\text{Pre}^{*}\ \text{Stem}\ \text{Suf}^{*})^{+} \tag{1}$$

e.g. English *un+accept+able*

$$\text{Stem} \mid \text{Pre} \mid \text{Suf} \rightarrow \text{Morph} \tag{2}$$

$$\text{Stem} \rightarrow \mathbf{intercal}\,(\text{Root}, \text{Template}) \tag{3}$$

e.g. Arabic derivation *k·t·b* + *i·a* ⇒ *kitab* (book)

$$\text{Stem} \rightarrow \mathbf{infix}\,(\text{Stem}, \text{Infix}) \tag{4}$$

e.g. Tagalog *sulat* (write) ⇒ ***s**um**ulat*** (wrote)

$$\text{Stem} \rightarrow \mathbf{circfix}\,(\text{Stem}, \text{Circumfix}) \tag{5}$$

e.g. Indonesian *percaya* (to trust)

⇒ ***ke**percaya**an*** (belief)

where the symbols (excluding Word and Stem) implicitly expand to the relevant terminal strings. The bold-faced "functions" combine the potentially discontiguous yields of the argument symbols into single contiguous strings, e.g. **infix**(s·ulat, um) produces stem *sumulat*.

Taken by themselves, the first two rules are simply a CFG that describes word formation as the concatenation of stems and affixes, a formulation that matches the underlying grammar of Morfessor (Creutz and Lagus, 2007), a well-studied unsupervised model.

The key aim of our extension is that we want the grammar to capture a discontiguous string like k·t·b as a single constituent in a parse tree. This leads to well-understood problems in probabilistic grammars (e.g. what is this rule's probability?), but also corresponds to the linguistic consideration that k·t·b is a proper morpheme of the language (Prunet, 2006).

3 Simple range concatenating grammars

In this section we define SRCGs formally and illustrate how they can be used to model non-concatenative morphology. SRCGs define languages that are recognisable in polynomial time, yet can capture discontiguous elements of a string under a single category (Boullier, 2000). An SRCG-rule operates on vectors of ranges in contrast to the way a CFG-rule operates on single ranges (spans). In other words, a non-terminal symbol in an SRCG (CFG) derivation can dominate a subset (substring) of terminals in an input string.

3.1 Formalism

An SRCG $\mathcal{G}$ is a tuple (N, T, V, P, S), with finite sets of non-terminals (N), terminals (T) and variables (V), with a start symbol $S \in N$. A rewrite rule $p \in P$ of rank $r = \rho(p) \geq 0$ has the form $A(\alpha_1, \ldots, \alpha_{\psi(A)}) \rightarrow B_1(\beta_{1,1}, \ldots, \beta_{1,\psi(B_1)}) \ldots B_r(\beta_{r,1}, \ldots, \beta_{r,\psi(B_r)})$, where each $\alpha, \beta \in (T \cup V)^*$, and $\psi(A)$ is the number of arguments a non-terminal A has, called its *arity*. By definition, the start symbol has arity 1. Any variable $v \in V$ appearing in a given rule must be used exactly once on each side of the rule. Terminating rules are written with ϵ as the right-hand side and thus have rank 0.

A *range* is a pair of integers (i, j) denoting the substring $w_{i+1} \ldots w_j$ of a string $w = w_1 \ldots w_n$. A non-terminal becomes *instantiated* when its variables are bound to ranges through substitution. Variables within an argument imply concatenation and therefore have to bind to adjacent ranges.

An instantiated non-terminal A' is said to derive ϵ if the consecutive application of a sequence of instantiated rules rewrite it as ϵ. A string w is within the language defined by a particular SRCG iff the start symbol S, instantiated with the exhaustive range $(0, w_n)$, derives ϵ.

An important distinction with regard to CFGs is that, due to the instantiation mechanism, the ordering of non-terminals on the right-hand side of an SRCG rule is irrelevant, i.e. $A(ab) \rightarrow B(a)C(b)$ and $A(ab) \rightarrow C(b)B(a)$ are the same rule.[2] Consequently, the isomorphisms of any given SRCG derivation tree all encode the same string, which is uniquely defined through the instantiation process.

3.2 Application to morphological analysis

A fragment of the idealised grammar schema from the previous section (§2) can be rephrased as an SRCG by writing the rules in the newly introduced

[2]Certain ordering restrictions over the variables *within* an argument need to hold for an SRCG to indeed be a *simple* RCG (Boullier, 2000).

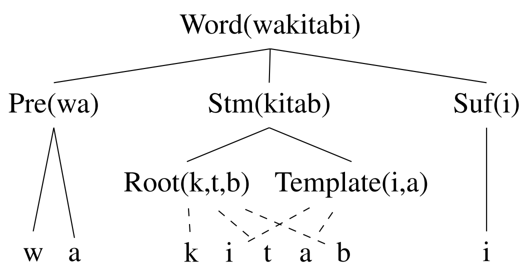


Figure 1: Example derivation for *wakitabi* (and my book) using the SRCG fragment from §3.2. CFGs cannot capture such crossing branches.

notation, and supplying a definition of the **intercal** function as simply another rule of the grammar, with instantiation for $w =$ kitab shown below:

$$\text{Word}(abc) \rightarrow \text{Pre}(a)\ \text{Stem}(b)\ \text{Suf}(c)$$
$$\text{Stem}(abcde) \rightarrow \text{Root}(a, c, e)\ \text{Template}(b, d),$$

$$\text{Stem}(\langle 0..1\rangle, \langle 1..2\rangle, \langle 2..3\rangle, \langle 3..4\rangle, \langle 4..5\rangle)$$
$$\rightarrow \text{Root}(\langle 0..1\rangle, \langle 2..3\rangle, \langle 4..5\rangle)$$
$$\text{Template}(\langle 1..2\rangle, \langle 3..4\rangle)$$

Given an appropriate set of grammar rules (as we present in §5), we can parse an input string to obtain a tree as shown in Figure 1. The overlapping branches of the tree demonstrate that this grammar captures something a CFG could not. From the parse tree one can read off the word's root morpheme and the template used.

Although SRCGs specify mildly context-sensitive grammars, each step in a derivation is context-free – a node's expansion does not depend on other parts of the tree. This property implies that a recognition/parsing algorithm can have a worst-case time complexity that is polynomial in the input length n, $O(n^{(\rho+1)\psi})$ for arity ψ and rank ρ, which reduces to $O(n^{3\psi})$ for a binarised grammar. To capture the maximal case of a root with $k-1$ characters and k discontiguous templatic characters forming a stem would require a grammar that has arity $\psi = k$. For Arabic, which has up to quadriliteral roots ($k = 5$), the time complexity would be $O(n^{15})$.[3] This is a daunting proposition for parsing, but we are careful

[3]The trade-off between arity and rank with respect to parsing complexity has been characterised (Gildea, 2010), and the appropriate refactoring may bring down the complexity for our grammars too.

to set up our application of SRCGs in such a way that this is not too big an obstacle:

Firstly, our grammars are defined over the characters that make up a word, and not over words that make up a sentence. As such, the input length n would tend to be shorter than when parsing full sentences from a corpus.

Secondly, we do type-based morphological analysis, a view supported by evidence from Goldwater et al. (2006), so each unique word in a dataset is only ever parsed once with a given grammar. The set of word types attested in the data sources of interest here is fairly limited, typically in the tens of thousands. For these reasons, our parsing and inference tasks turn out to be tractable despite the high time complexity.

4 Learning

4.1 Probabilistic SRCG

The probabilistic extension of SRCGs is similar to the probabilistic extension of CFGs, and has been used in other guises (Kato et al., 2006; Maier, 2010). Each rule $r \in P$ has an associated probability θ_r such that $\sum_{r \in P_A} \theta_r = 1$. A random string in the language of the grammar can then be obtained through a generative procedure that begins with the start symbol S and iteratively expands it until deriving ϵ: At each step for some current symbol A, a rewrite rule r is sampled randomly from P_A in accordance with the distribution over rules and used to expand A. This procedure terminates when no further expansions are possible. Of course, expansions need to respect the range concatenating and ordering constraints imposed by the variables in rules. The expansions imply a chain of variable bindings going down the tree, and instantiation happens only when rewriting into ϵs but then propagates back up the tree.

The probability $P(w, t)$ of the resulting tree t and terminal string w is the product $\prod_r \theta_r$ over the sequence of rewrite rules used. This generative procedure is a conceptual device; in practice, one would care about parsing some input string under this probabilistic grammar.

4.2 PYSRCAG

A central property of the generative procedure underlying probabilistic SRCGs is the fact that each expansion happens independently, both of the other expansions in the tree under construction and of any other trees. To some extent, this flies in the face of the reality of estimating a grammar from text, where one would expect certain sub-trees to be used repeatedly across different input strings.

Adaptor grammars weaken this independence assumption by allowing whole subtrees to be reused during expansion. Informally, they act as a cache of tree fragments whose tendency to be reused during expansion is governed by the choice of adaptor function. Following earlier applications of adaptor grammars (Johnson et al., 2007; Huang et al., 2011), we employ the Pitman-Yor process (Pitman, 1995; Pitman and Yor, 1997) as adaptor function.

A Pitman-Yor Simple Range Concatenating Adaptor Grammar (PYSRCAG) is a tuple $\mathcal{G} = (\mathcal{G}_S, M, \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{\alpha})$, where $\mathcal{G}_S$ is a probabilistic SRCG as defined before and $M \subseteq N$ is a set of *adapted non-terminals*. The vectors $\boldsymbol{a}$ and $\boldsymbol{b}$, indexed by the elements of M, are the discount and concentration parameters for each adapted non-terminal, with $a \in [0, 1], b \geq 0$. $\boldsymbol{\alpha}$ are parameters to Dirichlet priors on the rule probabilities θ.

PYSRCAG defines a generative process over a set of trees $\boldsymbol{T}$. Unadapted non-terminals $A' \in N \setminus M$ are expanded as before (§4.1). For each adapted non-terminal $A \in M$, a cache C_A is maintained for storing the terminating tree fragments expanded from A earlier in the process, and we denote the fragment corresponding to the i-th expansion of A as z_i. In other words, the sequence of indices z_i is the assignment of a sequence of expansions of A to particular tree fragments. Given a cache C_A that has n previously generated trees comprising m unique trees each used $n_1, \dots, n_m$ times (where $n = \sum_k n_k$), the tree fragment for the next expansion of A, z_{n+1}, is sampled conditional on the previous assignments $\boldsymbol{z}_<$ according to

$$z_{n+1} | \boldsymbol{z}_< \sim \begin{cases} \frac{n_k - a}{n+b} & \text{if } z_{n+1} = k \in [1, m] \\ \frac{ma+b}{n+b} & \text{if } z_{n+1} = m+1, \end{cases}$$

where a and b are those elements of $\boldsymbol{a}$ and $\boldsymbol{b}$ corresponding to A. The first case denotes the situation where a previously cached tree is reused for this $n+1$-th expansion of A; to be clear, this expands A with a fully terminating tree fragment, meaning that none of the nodes descending from A in the tree being generated are subject to further expansion. The second case by-passes the cache and expands A according to the rules P_A and rule probabilities θ_A of the underlying SRCG $\mathcal{G}_S$. Other caches $C_B(B \in M)$ may come into play during those expansions of the descendants of A; thus a PYSRCAG can define a hierarchical stochastic process. Both cases eventually result in a terminating tree-fragment for A, which is then added to the cache, updating the counts n, n_{z_n+1} and potentially m.

The adaptation does not affect the string language of $\mathcal{G}_S$, but it maps the distribution over trees to one that is distributed according to the PYP.

The invariance of SRCGs trees under isomorphism would make the probabilistic model deficient, but we side-step this issue by requiring that grammar rules are specified in a canonical way that ensures a one-to-one correspondence between the order of nodes in a tree and of terminals in the yield.

4.3 Inference under PYSRCAG

The inference procedure under our model is very similar to that of CFG PY-adaptor grammars, so we restate the central aspects here but refer the reader to the original article by Johnson et al. (2007) for further details. First, one may integrate out the adaptors to obtain a single distribution over the set of trees generated from a particular non-terminal. Thus, the joint probability of a particular sequence $\boldsymbol{z}$ for the adapted non-terminal A with cached counts $(n_1, \ldots, n_m)$ is

$$PY(\boldsymbol{z}|a,b) = \frac{\prod_{k=1}^{m}\left(a(k-1)+b\right)\prod_{j=1}^{n_k-1}(j-a)}{\prod_{i=0}^{n-1}(i+b)}. \tag{6}$$

Taking all the adapted non-terminals into account, the joint probability of a set of full trees $\boldsymbol{T}$ under the grammar $\mathcal{G}$ is

$$P(\boldsymbol{T}|\boldsymbol{a},\boldsymbol{b},\boldsymbol{\alpha}) = \prod_{A\in M}\frac{\mathrm{B}(\boldsymbol{\alpha}_A+\boldsymbol{f}_A)}{\mathrm{B}(\boldsymbol{\alpha}_A)}PY(\boldsymbol{z}(\boldsymbol{T})|\boldsymbol{a},\boldsymbol{b}), \tag{7}$$

where $\boldsymbol{f}_A$ is a vector of the usage counts of rules $r \in P_A$ across $\boldsymbol{T}$, and B is the Euler beta function.

The posterior distribution over a set of *strings* $\boldsymbol{w}$ is obtained by marginalising (7) over all trees that have $\boldsymbol{w}$ as their yields. This is intractable to compute directly, so instead we use MCMC techniques to obtain samples from that posterior using a component-wise Metropolis-Hastings sampler. The sampler works by visiting each string w in turn and drawing a new tree for it under a proposal grammar $\mathcal{G}_Q$ and randomly accepting that as the new analysis for w according to the Metropolis-Hastings accept-reject probability. As proposal grammar, we use the analogous approximation of our $\mathcal{G}$ as Johnson et al. used for PCFGs, namely by taking a static snapshot $\mathcal{G}_Q$ of the adaptor grammar where additional rules rewrite adapted non-terminals as the terminal strings of their cached trees. Drawing a sample from the proposal distribution is then a matter of drawing a random tree from the parse chart of w under G_Q.

Lastly, the adaptor hyperparameters $\boldsymbol{a}$ and $\boldsymbol{b}$ are modelled by placing flat $\mathrm{Beta}(1,1)$ and vague $\mathrm{Gamma}(10,0.1)$ priors on them, respectively, and inferring their values using slice sampling (Johnson and Goldwater, 2009).

5 Modelling root-templatic morphology

We start with a CFG-based adaptor grammar[4] that models words as a stem and any number of prefixes and suffixes:

$$\text{Word} \rightarrow \underline{\text{Pre}}^{*}\ \underline{\text{Stem}}\ \underline{\text{Suf}}^{*} \tag{8}$$

$$\underline{\text{Pre}} \mid \underline{\text{Stem}} \mid \underline{\text{Suf}} \rightarrow \text{Char}^{+} \tag{9}$$

This fragment can be seen as building on the stem-and-affix adaptor grammar presented in (Johnson et al., 2007) for morphological analysis of English, of which a later version also covers multiple affixes (Sirts and Goldwater, 2013). In the particular case of Arabic, multiple affixes are required to handle the attachment of particles and proclitics onto base words.

To extend this to *complex stems* consisting of a root with three radicals we have rules like the following:

$$\underline{\text{Stem}}(abcdefg) \rightarrow \underline{\text{R3}}(b,d,e)\ \text{T4}(a,c,e,g) \tag{10}$$

$$\underline{\text{Stem}}(abcdef) \rightarrow \underline{\text{R3}}(a,c,e)\ \text{T3}(b,d,f) \tag{11}$$

$$\underline{\text{Stem}}(abcde) \rightarrow \underline{\text{R3}}(a,c,e)\ \text{T2}(b,d) \tag{12}$$

$$\underline{\text{Stem}}(abcd) \rightarrow \underline{\text{R3}}(a,c,d)\ \text{T1}(b) \tag{13}$$

$$\underline{\text{Stem}}(abc) \rightarrow \underline{\text{R3}}(a,b,c) \tag{14}$$

[4]Adapted non-terminals are indicated by underlining and we use the following abbreviations: X $\rightarrow$ Y^{+} means one or more instances of Y and encodes the rules X $\rightarrow$ Ys and Ys $\rightarrow$ Ys Y | Y. Similarly, X $\rightarrow$ Y* Z allows zero or more instances of Y and encodes the rules X $\rightarrow$ Z and X $\rightarrow$ Y^{+} Z. Further relabelling is added as necessary to avoid cycles among adapted non-terminals.

The actual rules include certain permutations of these, e.g. rule (13) has a variant $\underline{\mathrm{R3}}(a, b, d)\mathrm{T1}(c)$. In unvocalised text, the standard written form of Modern Standard Arabic (MSA), it may happen that the stem and the root of a word form are one and the same. So while rule (14) may look trivial, it ensures that in such cases the radicals are still captured as descendants of the non-terminal category $\underline{\mathrm{R3}}$, thereby making their appearance in the cache.

A discontiguous non-terminal An is rewritten through recursion on its arity down to 1, i.e. $\mathrm{A}n(v_1, \ldots, v_n) \rightarrow \mathrm{A}l(v_1, \ldots, v_{n-1})\ \mathrm{Char}(v_n)$ with base case $\mathrm{A1}(v) \rightarrow \mathrm{Char}(v)$, where Char rewrites all individual terminals as ϵ, v_i are variables and $l = n-1$.[5] Note that although we provide the model with two sets of discontiguous non-terminals R and T, we do not specify their mapping onto the actual terminal strings; no subdivision of the alphabet into vowels and consonants is hard-wired.

6 Experiments

We evaluate our model on standard Arabic, Quranic Arabic and Hebrew in terms of segmentation quality and lexicon induction ability. These languages share various properties, including morphology and lexical cognates, but are sufficiently different so as to require manual intervention when transferring rule-based morphological analysers across languages. A key question in this evaluation is therefore whether an appropriate instantiation of our model successfully generalises across related languages.

6.1 Data sets

Our models are unsupervised and therefore learn from raw text, but their evaluation requires annotated data as a gold-standard and these were derived[6] as follows:

Arabic (MSA) We created the dataset BW by synthesising 50k morphotactically correct word types from the morpheme lexicons and consistency rules supplied with the Buckwalter Arabic Morphological Analyser (BAMA).[7] This allowed control over the word shapes, which is important to focus the evaluation, while yielding reliable segmentation and root annotations. BW has no vocalisation; we denote the corresponding *vocalised* dataset as BW$'$.

Quranic Arabic We extracted the roughly 18k word types from a morphologically analysed version of the Quran (Dukes and Habash, 2010). As an additional challenge, we left all given diacritics intact for this dataset, QU$'$.

Hebrew We leveraged the Hebrew CHILDES database as an annotated resource (Albert et al., 2013) and were able to extract 5k word types that feature at least one affix to use as dataset HEB. The corrected versions of words marked as non-standard child language were used, diacritics were dropped, and we conflated stressed and unstressed vowels to overcome inconsistencies in the source data.

	Types	Stems	Roots	m/w	c/w
BW	48428	24197	4717	2.3	6.4
BW$'$	48428	30891	4707	2.3	10.7
QU$'$	18808	12021	1270	1.9	9.9
HEB	5231	3164	492	2.1	6.7

Table 1: Corpus statistics, including average number of morphemes (m/w) and characters (c/w) per word, and total surface-realised roots of length 3 or 4.

6.2 Models

We consider two classes of models. The first is the strictly context-free adaptor grammar for morphemes as sequences of characters using rules (8)-(9), which we denote as **Concat** and **MConcat**, where the latter allows multiple prefixes/suffixes in a word. These serve as baselines for the second class in which non-concatenative rules are added. **MTpl** and **Tpl** denote the canonical ver-

[5]Including the arity as part of the non-terminal symbol names forms part of our convention here to ensure that the grammar contains no cycles, a situation which would complicate inference under PYSRCAG.

[6]Our data preprocessing scripts are obtainable from `http://github.com/bothameister/pysrcag-data`.

[7]We used version 2.0, LDC2004L02, and sampled word types having a single stem and at most one prefix, suffix or both, according to the following random procedure: Sample a shape (stem: 0.1, pre+stem: 0.25 stem+suf: 0.25, pre+stem+suf: 0.4). Sample uniformly at random (with replacement) a stem from the BAMA stem lexicon, and affix(es) from the ones consistent with the chosen stem. The BAMA lexicons contain affixes and their legitimate concatenations, so some of the generated words would permit a linguistic segmentation into multiple prefixes/suffixes. Nonetheless, we take as gold-standard segmentation precisely the items used by our procedure.

sions with stems as shown in the set of rules above, and we experiment with a variant **Tpl3Ch** that allows the non-terminal T1 to be rewritten as up to three Char symbols, since the data indicate there are cases where multiple characters intervene between the radicals of a root.

These models exclude rule (10), which we include only in the variant **Tpl+T4**. Lastly, **TplR4** is the extension of **Tpl+T4** to include a stem-forming rule that uses $R4$.

As external baseline model we used **Morfessor** (Creutz and Lagus, 2007), which performs decently in morphological segmentation of a variety of languages, but only handles concatenation.

6.3 Method

The MCMC samplers converged within a few hundred iterations and we collected 100 posterior samples after 900 iterations of burn-in. Collected samples, each of which is a set of parse trees of the input word types, are used in two ways:

First, by averaging over the samples we can estimate the joint probability of a word type w and a parse tree t under the adaptor grammar, conditional on the data and the model's hyperparameters. We take the most probable parse of each word type and evaluate the implied segmentation against the gold standard segmentation. Likewise, we evaluate the implied lexicon of stems, affixes and roots against the corresponding reference sets. It should be emphasised that using this maximally probable analysis is aimed at simplifying the evaluation set-up; one could also extract multiple analyses of a word since the model defines a distribution over them.

The second method abstracts away from individual word-types and instead averages over the union of all samples to obtain an estimate of the probability of a string s being generated by a certain category (non-terminal) of the grammar. In this way we can obtain a lexicon of the morphemes in each category, ranked by their probability under the model.

6.4 Inducing Morpheme Lexicons

The quality of each induced lexicon is measured with standard set-based precision and recall with respect to the corresponding gold lexicon. The results are summarised by balanced F-scores in Table 2.

The main result is that all our models capable of forming complex stems obtain a marked improvement in F-scores over the baseline concatenative adaptor grammar, and the margin of improvement grows along with the expressivity of the complex-stem models tested. This applies across prefix, stem and suffix categories and across our datasets, with the exception of $\text{Q}\textsc{u}'$, which we elaborate on in §6.5.

Stem lexicons of Arabic were learnt with relatively constant precision ($\sim$70%), but modelling complex stems broadened the coverage by about 3000 stems over the concatenative model (against a reference set of 24k stems). On vocalised Arabic, the improvements for stems are along both dimensions. In contrast, affix lexicons for both $\text{B}\textsc{w}$ and $\text{B}\textsc{w}'$ are noisy and the models all generate greedily to obtain near perfect recall but low precision.

On our Hebrew data, which comprises only 5k words, the gains in lexicon quality from modelling complex stems tend to be larger than on Arabic. This is consistent with our intuition that an appropriate, richer Bayesian prior helps overcome data sparsity.

Extracting a lexicon of roots is rendered challenging by the unsupervised nature of the model as the labelling of grammar symbols is ultimately arbitrary. Our simple approach was to regard a character tuple parsed under category R3 as a root. This had mixed success, as demonstrated by the outlier scores in Table 2. In the one case where it was obvious that T3 had been been co-opted for the role, we report the F-score obtained on the union of R3 and T3 strings.

Soft decisions The preceding set-based evaluation imposes hard decisions about category membership. But adaptor grammars are probabilistic by definition and should thus also be evaluated in terms of probabilistic ability. One method is to turn the model predictions into a binary classifier of strings using Receiver-Operator-Characteristic (ROC) theory. We plot the true positive rate versus the false positive rate for each prediction lexicon L_τ containing strings that have probability greater than τ under the model (for a grammar category of interest). A perfect classifier would rank all true positives (e.g. stem strings) above false positives (e.g. non-stem strings), corresponding to a curve in the upper left corner of the ROC plot. A random guesser would trace a diagonal line. The area under the curves (AUC) is the probability that the classifier would discriminate correctly.

	Vocalised Arabic (BW′)				**Unvocalised Arabic** (BW)				**Hebrew** (HEB)			
	Pre	Stem	Suf	R3	Pre	Stem	Suf	R3	Pre	Stem	Suf	R3
Concat	15.0	20.2	25.4	-	32.8	44.1	40.3	-	18.7	20.9	29.2	-
Tpl	24.7	39.4	35.2	†42.4	45.9	54.7	47.9	62.7	35.1	59.6	52.9	34.8
Tpl3Ch	28.4	36.0	36.5	5.2	50.3	55.1	48.5	62.4	**38.6**	61.5	**56.6**	7.1
Tpl+T4	29.0	44.8	41.0	3.9	46.2	54.2	47.7	62.3	32.5	59.6	53.0	**36.4**
TplR4	**37.8**	**60.3**	**47.0**	5.2	**53.0**	**57.7**	**51.9**	62.4	38.0	**62.4**	55.2	34.7

Table 2: Morpheme lexicon induction quality. F1-scores for lexicons induced from the most probable parse of each different dataset under each models. †42.4 was obtained by taking the union of R3 and T3 items to match the way the model used them (see §6.4).

	BW′	BW	QU′	HEB
Morfessor	55.57	40.04	**44.34**	24.20
Concat	47.36	64.22	19.64	60.05
Tpl	60.42	71.91	22.53	77.26
Tpl3Ch	60.52	72.20	25.72	77.41
Tpl+T4	64.49	71.59	24.81	77.14
TplR4	**74.54**	**73.66**	-	**78.14**

Table 3: Segmentation quality in SBF1. The QU′ results are for the corresponding **M*** models .

Our models with complex stem formation improve over the baseline on the AUC metric too. We include the ROC plots for Hebrew stem and root induction in Figure 2, along with the roots the model was most confident about (Table 4).

6.5 Morphological Analysis per Word Type

In this section we turn to the analyses our models assign to each word type. Two aspects of interest are the segmentation into sequential morphemes and the identification of the root.

Our intercalating adaptor grammars consistently obtain large gains in segmentation accuracy over the baseline concatenative model, across all our datasets (Table 3). We measure segmentation quality as *segment border F1-score* (SBF) (Sirts and Goldwater, 2013), which is the F-score over word-internal segmentation points of the predicted analysis with respect to the gold segmentation.

Of the two MSA datasets, the vocalised version BW′ presents a more difficult segmentation task as its words are on average longer and feature 31k unique contiguous morphemes, compared to the 24k in BW for the same number of words. It should thus benefit more from additional model expressivity, as is reflected in the increase of 10 SBF when adding the **TplR4** rule to the other triliteral ones.

The best triliteral root identification accuracy (on a per-word basis) was found for HEB (74%) and BW (67%).[8,9] Refer to Figure 3 for example analyses.

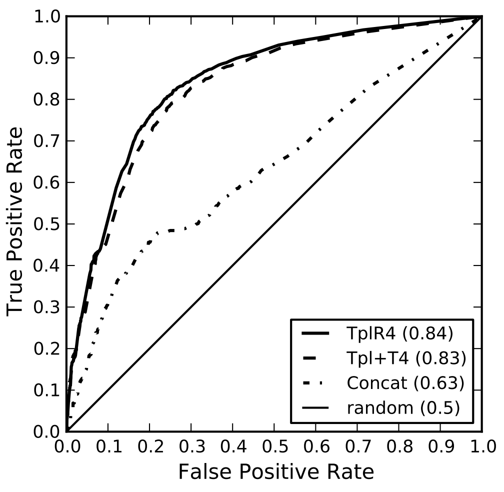
An interesting aspect of these results is that templatic rules may aid segmentation quality without necessarily giving perfect root identification. Modelling stem substructure allows any regularities that give rise to a higher data likelihood to be picked up.

The low performance on the Quran demands further explanation. All our adaptor grammars severely oversegmented this data, although the mistakes were not uniformly distributed. Most of the performance loss is on the 79% of words that have 1-2 morphemes. On the remaining words (having 3-5 morphemes), our models recover and approach the Morfessor baseline (**MConcat**: 32.7 , **MTpl3Ch**: 38.6).

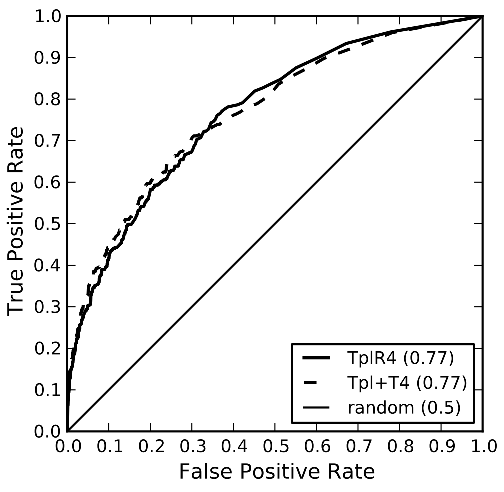
Preliminary experiments on BW had indicated that adaptation of (single) affix categories is crucial for good performance. Our multi-affixing models used on QU′ lacked a further level of adaptation for composite affixes, which we suspect as a contributing factor to the lower performance on that dataset. This remains to be confirmed in future experiments, but would be consistent with other observations on the role of hierarchical adaptation in adaptor grammars (Sirts and Goldwater, 2013). The trend that intercalated rules improve segmentation (compared to the concatenative grammar) remains consistent

[8]When excluding cases where root equals stem, root identification on BW is 55%. Those cases are still not trivial, since words without roots also exist.

[9]By way of comparison, Rodrigues and Ćavar (2007) presented an unsupervised statistics-based root identification method that obtained precision ranging between 50-75%, the higher requiring vocalised words.



(a) Stems



(b) Triliteral roots

Figure 2: ROC curves for predicting the stem and root lexicons for the Heb dataset. The area under each curve (AUC), as computed with the trapezium rule, is given in parentheses.

across datasets, despite the lower absolute performance on Qu′.

The performance of the Morfessor baseline was quite mixed. Contrary to our expectations, it performs best on the "harder" Bw′, worst on the arguably simpler Heb and struggled less than the adaptor grammars on Qu′.

One factor here is that it learns according to a grammar with multiple consecutive affixes and stems, whereas all our experiments (except on Qu′) presupposed single affixes. This biases the evaluation slightly in our favour, but works in Morfessor's favour on the Qu′ data which is annotated with multiple affixes.

Root		Example instances
1. spr ✓	G	**š**a**p̌**a**ř**.ti te.**š**a**p̌ř** ye.**š**a**p̌ř**.u
	B	**s**i**p**u**r**.im hi$_x$**št**a**p̌**$_x$ař_t
2. lbs ✓	G	**ľ**a**b̌**a**Š**.t li.**ľb̌**o**Š** ti.**ľb̌**e**Š**.i
	B	le_ha$_x$**ľb̌**i**Š** ti_t$_x$**ľ**a**b̌Š**.i
3. ptx ✓	G	**p̌**a**ť**a**x̌**.ti ti.**p̌ť**e**x̌**.i
	B	li.**p̌**ť**o**a**x̌** ni$_x$**p̌ť**a**x̌**.at
5. !al ×	B	ya.**!al**.u m̌a$_x$**!aľ**.a **!ačľ**$_x$an_it

Table 4: Top Hebrew roots hypothesised by **Tpl+T4**. Numbers indicate position when ranked by model probability. (G)ood and (B)ad instances from the corpus are given with morpheme boundaries marked: true positive (.), false negative (_) and false positive ($_x$). Hypothesised root characters are boldfaced, while accent (ˇ) marks gold root characters.

7 Related work

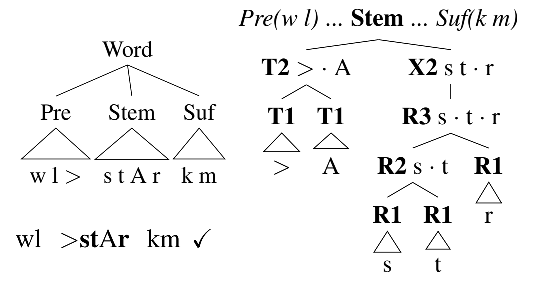
The distinctive feature of our morphological model is that it jointly addresses root identification and morpheme segmentation, and our results demonstrate the mutual benefit of this.

In contrast, earlier unsupervised approaches tend to focus on these tasks in isolation.

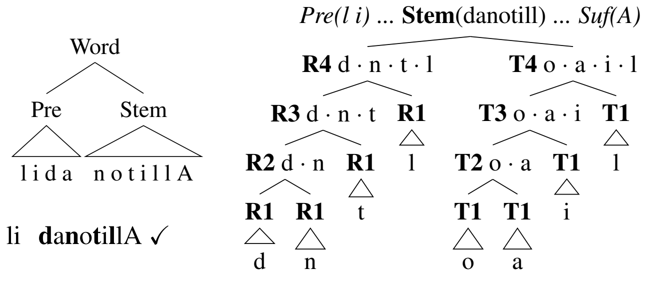
In unsupervised Arabic segmentation, the parametric Bayesian model of (Lee et al., 2011) achieves F1-scores in the high eighties by incorporating sentential context and inferred syntactic categories, both of which our model forgoes, although theirs has no account of discontiguous root morphemes.

Previous approaches to Arabic root identification that sought to use little supervision typically constrain the search space of candidate characters within a word, leveraging pre-existing dictionaries (Darwish, 2002; Boudlal et al., 2009) or rule constraints (Elghamry, 2005; Rodrigues and Ćavar, 2007; Daya et al., 2008).

In contrast to these approaches, our model requires no dictionary, and while our grammar rules effect some constraints on what could be a root, they are specified in a convenient and flexible manner that



(a) **Concat** & **Tpl+T4**, "wl>stArkm" (BW)



(b) **Concat** & **TplR4**, "lidanotillA" (BW′)

Figure 3: Parse trees produced for words in the two standard Arabic datasets that were incorrectly segmented by the baseline grammar. The templatic grammars correctly identified the triliteral and quadriliteral roots, also fixing the segmentation of (a). In (b), the templatic grammar improved over the baseline by finding the correct prefix but falsely posited a suffix. Unimportant subtrees are elided for space, while the yields of discontiguous constituents are indicated next to their symbols, with dots marking gaps. Crossing branches are not drawn but should be inferrable. Root characters are bold-faced in the reference analysis ✓. The non-terminal X2 in (a) is part of a number of implementation-specific helper rules that ensure the appropriate handling of partly contiguous roots.

makes experimentation with other phenomena easy.

Recent work by Fullwood and O'Donnell (2013) goes some way toward jointly dealing with non-concatenative and concatenative morphology in the unsupervised setting, but their focus is limited to inflected stems and does not handle multiple consecutive affixes. They analyse the Arabic verb stem (e.g. kataba "he wrote") into a templatic bit-string denoting root and non-root characters (e.g. r-r-r-) along with a root morpheme (e.g. ktb) and a so-called residue morpheme (e.g. aaa). Their nonparametric Bayesian model induces lexicons of these entities and achieves very high performance on templates. The explicit formulation of templates alleviates the labelling ambiguity that hampered our evaluation (§6.4), but we believe their method of analysis can be simulated in our framework using the appropriate SRCG-rules.

Learning root-templatic morphology is loosely related to morphological paradigm induction (Clark, 2001; Dreyer and Eisner, 2011; Durrett and DeNero, 2013). Our models do not represent templatic paradigms explicitly, but it is interesting to note that preliminary experiments with German indicate that our adaptor grammars pick up on the past participle forming circumfix in ab+**ge**+spiel+**t** (*played back*).

8 Conclusion and Outlook

We presented a new approach to modelling non-concatenative phenomena in morphology using simple range concatenating grammars and extended adaptor grammars to this formalism. Our experiments show that this richer model improves morphological segmentation and morpheme lexicon induction on different languages in the Semitic family.

Various avenues for future work present themselves. Firstly, the lightly-supervised, meta-grammar approach to adaptor grammars (Sirts and Goldwater, 2013) can be extended to this more powerful formalism to lessen the burden of defining the "right" grammar rules by hand, and possibly boost performance. Secondly, the discontiguous constituents learnt with our framework can be used as features in other downstream applications. Especially in low-resource languages, the ability to model non-concatenative phenomena (e.g. circumfixing, ablaut, etc.) can play an important role in reducing data sparsity for tasks like word alignment and language modelling. Finally, the PYSRCAG presents another way of learning SRCGs in general, which can thus be employed in other applications of SRCGs, including syntactic parsing and translation.

Acknowledgements

We thank the anonymous reviewers for their valuable comments. Our PYSRCAG implementation leveraged the adaptor grammar code released by Mark Johnson, whom we thank, along with the individuals who contributed to the public data sources that enabled the empirical elements of this paper.

References

Aviad Albert, Brian MacWhinney, Bracha Nir, and Shuly Wintner. 2013. The Hebrew CHILDES corpus: transcription and morphological analysis. *Language Resources and Evaluation*, pages 1–33.

Mohamed Altantawy, Nizar Habash, Owen Rambow, and Ibrahim Saleh. 2010. Morphological Analysis and Generation of Arabic Nouns: A Morphemic Functional Approach. In *Proceedings of LREC*, pages 851–858.

Kenneth R Beesley and Lauri Karttunen. 2003. *Finite state morphology*, volume 18. CSLI publications Stanford.

Abderrahim Boudlal, Rachid Belahbib, Abdelhak Lakhouaja, Azzeddine Mazroui, Abdelouafi Meziane, and Mohamed Bebah. 2009. A Markovian approach for Arabic Root Extraction. *The International Arab Journal of Information Technology*, 8(1):91–98.

Pierre Boullier. 2000. A cubic time extension of context-free grammars. *Grammars*, 3(2-3):111–131.

Tim Buckwalter. 2002. Arabic Morphological Analyzer. Technical report, Linguistic Data Consortium, Philedelphia.

Alexander Clark. 2001. Learning Morphology with Pair Hidden Markov Models. In *Proceedings of the ACL Student Workshop*, pages 55–60.

Yael Cohen-Sygal and Shuly Wintner. 2006. Finite-state registered automata for non-concatenative morphology. *Computational Linguistics*, 32(1):49–82.

Mathias Creutz and Krista Lagus. 2007. Unsupervised models for morpheme segmentation and morphology learning. *ACM Transactions on Speech and Language Processing*, 4(1):1–34.

Kareem Darwish. 2002. Building a shallow Arabic morphological analyzer in one day. In *Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages*, pages 47–54. Association for Computational Linguistics.

Ezra Daya, Dan Roth, and Shuly Wintner. 2008. Identifying Semitic Roots: Machine Learning with Linguistic Constraints. *Computational Linguistics*, 34(3):429–448.

Markus Dreyer and Jason Eisner. 2011. Discovering Morphological Paradigms from Plain Text Using a Dirichlet Process Mixture Model. In *Proceedings of EMNLP*, pages 616–627, Edinburgh, Scotland.

Kais Dukes and Nizar Habash. 2010. Morphological Annotation of Quranic Arabic. In *Proceedings of LREC*.

Greg Durrett and John DeNero. 2013. Supervised Learning of Complete Morphological Paradigms. In *Proceedings of NAACL-HLT*, pages 1185–1195, Atlanta, Georgia, June. Association for Computational Linguistics.

Khaled Elghamry. 2005. A Constraint-based Algorithm for the Identification of Arabic Roots. In *Proceedings of the Midwest Computational Linguistics Colloquium. Indiana University. Bloomington, IN.*

Raphael Finkel and Gregory Stump. 2002. Generating Hebrew verb morphology by default inheritance hierarchies. In *Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages*. Association for Computational Linguistics.

Michelle A. Fullwood and Timothy J. O'Donnell. 2013. Learning non-concatenative morphology. In *Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics*, pages 21–27, Sofia, Bulgaria. Association for Computational Linguistics.

Michael Gasser. 2009. Semitic morphological analysis and generation using finite state transducers with feature structures. In *Proceedings of EACL*, pages 309–317. Association for Computational Linguistics.

Daniel Gildea. 2010. Optimal Parsing Strategies for Linear Context-Free Rewriting Systems. In *Proceedings of NAACL*, pages 769–776. Association for Computational Linguistics.

Sharon Goldwater, Thomas L. Griffiths, and Mark Johnson. 2006. Interpolating Between Types and Tokens by Estimating Power-Law Generators. In *Advances in Neural Information Processing Systems, Volume 18*.

Harald Hammarström and Lars Borin. 2011. Unsupervised Learning of Morphology. *Computational Linguistics*, 37(2):309–350.

Yun Huang, Min Zhang, and Chew Lim Tan. 2011. Nonparametric Bayesian Machine Transliteration with Synchronous Adaptor Grammars. In *Proceedings of ACL (Short papers)*, pages 534–539.

Mark Johnson and Sharon Goldwater. 2009. Improving nonparameteric Bayesian inference: Experiments on unsupervised word segmentation with adaptor grammars. In *Proceedings of NAACL-HLT*, pages 317–325. Association for Computational Linguistics.

Mark Johnson, Thomas L. Griffiths, and Sharon Goldwater. 2007. Adaptor Grammars: A Framework for Specifying Compositional Nonparametric Bayesian Models. In *Advances in Neural Information Processing Systems*, volume 19, page 641. MIT.

Mark Johnson. 2008. Unsupervised word segmentation for Sesotho using Adaptor Grammars. In *Proceedings of ACL Special Interest Group on Computational Morphology and Phonology (SigMorPhon)*, pages 20–27. Association for Computational Linguistics.

Aravind K. Joshi. 1985. Tree adjoining grammars: How much context-sensitivity is required to provide reasonable structural descriptions? In D.R. Dowty, L. Karttunen, and A.M. Zwicky, editors, *Natural Language Parsing*, chapter 6, pages 206–250. Cambridge University Press.

Miriam Kaeshammer. 2013. Synchronous Linear Context-Free Rewriting Systems for Machine Translation. In *Proceedings of the Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 68–77, Atlanta, Georgia. Association for Computational Linguistics.

Yuki Kato, Hiroyuki Seki, and Tadao Kasami. 2006. Stochastic Multiple Context-Free Grammar for RNA Pseudoknot Modeling. In *Proceedings of the International Workshop on Tree Adjoining Grammar and Related Formalisms*, pages 57–64.

George Anton Kiraz. 2000. Multitiered Nonlinear Morphology Using Multitape Finite Automata: A Case Study on Syriac and Arabic. *Computational Linguistics*, 26(1):77–105, March.

Kimmo Koskenniemi. 1984. A general computational model for word-form recognition and production. In *Proceedings of the 10th international conference on Computational Linguistics*, pages 178–181. Association for Computational Linguistics.

Mikko Kurimo, Sami Virpioja, Ville T. Turunen, Graeme W. Blackwood, and William Byrne. 2010. Overview and Results of Morpho Challenge 2009. In *Multilingual Information Access Evaluation I. Text Retrieval Experiments*, volume 6241 of *Lecture Notes in Computer Science*, pages 578–597. Springer Berlin / Heidelberg.

Yoong Keok Lee, Aria Haghighi, and Regina Barzilay. 2011. Modeling syntactic context improves morphological segmentation. In *Proceedings of CoNLL*.

Wolfgang Maier. 2010. Direct Parsing of Discontinuous Constituents in German. In *Proceedings of the NAACL-HLT Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 58–66. Association for Computational Linguistics.

Jim Pitman and Marc Yor. 1997. The Two-Parameter Poisson-Dirichlet Distribution Derived from a Stable Subordinator. *The Annals of Probability*, 25(2):855–900.

Jim Pitman. 1995. Exchangeable and partially exchangeable random partitions. *Probability Theory and Related Fields*, 102:145–158.

Jean-François Prunet. 2006. External Evidence and the Semitic Root. *Morphology*, 16(1):41–67.

Paul Rodrigues and Damir Ćavar. 2007. Learning Arabic Morphology Using Statistical Constraint-Satisfaction Models. In Elabbas Benmamoun, editor, *Perspectives on Arabic Linguistics: Proceedings of the 19th Arabic Linguistics Symposium*, pages 63–75, Urbana, IL, USA. John Benjamins Publishing Company.

Nathan Schneider. 2010. Computational Cognitive Morphosemantics: Modeling Morphological Compositionality in Hebrew Verbs with Embodied Construction Grammar. In *Proceedings of the Annual Meeting of the Berkeley Linguistics Society*, Berkeley, CA.

Hiroyuki Seki, Takashi Matsumura, Mamoru Fujii, and Tadao Kasami. 1991. On multiple context-free grammars. *Theoretical Computer Science*, 88(2):191–229.

Kairit Sirts and Sharon Goldwater. 2013. Minimally-Supervised Morphological Segmentation using Adaptor Grammars. *Transactions of the ACL*.

K. Vijay-Shanker, David J. Weir, and Aravind K. Joshi. 1987. Characterizing structural descriptions produced by various grammatical formalisms. In *Proceedings of ACL*, pages 104–111.