Syntax and Parsing of Semitic Languages

Reut Tsarfaty

Abstract The grammar of Semitic languages is different from that of English and many other languages. Therefore, general-purpose statistical parsers are not always equally successful when applied to Semitic data. This chapter considers the syntax of Semitic languages and how it challenges existing general-purpose parsing architectures. We then survey the different components of a generative probabilistic parsing system and show how they can be designed and implemented in order to effectively cope with Semitic data. We finally present parsing results obtained for Hebrew and Arabic using different technologies in different scenarios. While parsing Semitic languages can already be made quite accurate using the present techniques, remaining challenges leave ample space for future research.

1 Introduction

Parsing, the task of automatically analyzing the syntactic structure of natural language sentences, is a core task in Natural Language Processing (NLP). Syntactic analysis unravels the way in which words combine to form phrases and sentences. The syntactic analysis of sentences is immensely important from a linguistic point of view, as syntactic structures map language-specific phenomena (the form of words, their order, their grouping) onto abstract, language-independent notions (subject, predicate, object, etc.). It is also important from a technological viewpoint, as it helps to recover the notions of "who did what to whom" from unstructured data.

The best parsing systems to date are supervised, data-driven and statistical. Many state-of-the-art statistical parsers have been tuned to, and excel at, parsing English. Such parsers will not necessarily perform as well when trained on data from a different language — and indeed, existing parsing technologies do not lend themselves to parsing Semitic texts so easily.

Reut Tsarfaty
Uppsala University, e-mail: tsarfaty@stp.lingful.uu.se

Semitic languages such as Arabic, Hebrew, Amharic or Maltese belong to the Afro-Asian family, and they are assumed to be descendants of the same ancient ancestor, called the *Proto-Semitic*. The reliance on consonantal roots, conjugation patterns and inflections is assumed to be inherited from that Proto-Semitic ancestor. The linguistic structure of Semitic languages is very different from that of English. Most of the Semitic languages are written right-to-left and each language utilizes its own writing systems. These languages introduce vocalization patterns and orthography that are particular to the Semitic family (chapter 1, this volume), and manifest rich morphology and word-order patterns that are different from that of English (§1.2). Even though Semitic languages have an important typological and sociological status, they have been under-studied in terms of language technology. Now, with availability of annotated corpora for Hebrew and Arabic and recent advances made in parsing technology for morphologically rich languages [102], the time is ripe for an illuminating discussion of effective techniques for parsing Semitic languages.

We take up the opportunity to look at statistical parsing technology from this typological perspective. We ask questions such as: How can we build parsing systems that utilize the massive technological advances made in parsing English while doing justice to the linguistic phenomena exhibited by Semitic languages? How can we evaluate those models within and across frameworks? How well do these models perform relative to general-purpose ones in real-world parsing scenarios?

In this chapter we cannot hope to do justice to the vast literature on parsing and theoretical studies on Semitic syntax, but if you follow this chapter through, we expect that you will be able to successfully conduct (at least) the following: (i) put together a baseline parsing system that is aware of the Semitic challenges, (ii) effectively employ modeling techniques that have proven useful in parsing Semitic texts, and (iii) use a range of evaluation metrics in order to get a fair grasp of parse quality and system bottlenecks. This chapter assumes knowledge of set theory and probability theory, and familiarity with basic formal language theory (good references would be [68] or [52]). Many of the presented systems are downloadable from the web and may be used as is. If you do not intend to develop a parsing architecture yourself, this chapter will endow you with a deeper understanding that will help you set up your data, your choice of parameters, the feature design, etc., so as to avoid the common pitfalls of blindly applying general-purpose parsers across languages.

The remainder of this chapter is organized as follows. In Section §1 we chart the parsing world in terms of representations, models and algorithms (§1.1). We then describe Semitic orthographic, morphological and syntactic phenomena (§1.2) and outline the overarching challenges in parsing Semitic languages: the architectural challenge, the modeling challenge, and the lexical challenge (§1.3). In Section 2 we present in detail a case study from the constituency-based parsing paradigm. After formally defining the basic generative parsing system (§2.1), we present variations that address the architectural challenge (§2.2) , the modeling challenge (§2.3), and the lexical challenge (§2.4) for Semitic languages. Section §3 surveys the main parsing results that have been reported for Semitic languages so far. Section §4 summarizes the current state-of-affairs and concludes with open questions for further research.

1.1 Parsing Systems

Syntactic Analysis

A parsing system is a computer program that takes a sentence in a natural language as input and provides a representation of its human perceived syntactic analysis as output. For example, the syntactic analysis of sentence (1) below should identify the syntactic entities "I" and "this book", and formally represent the relations between them, that is, that "I" is the *subject* of "like", and that "this book" is its *object*.

(1) I like this book

The syntactic analysis of a sentence is formally represented as a connected graph which represents entities as nodes and relations as arcs. These representations build on formal linguistic frameworks that have been proposed and developed by linguistic theorists in the last century.

One way to represent this information is by means of a *constituency structure* [9, 25]. Formally, constituency structures are linearly-ordered trees in which internal nodes are labeled with phrase types from a finite set (including Noun Phrase (NP), Verb Phrase (VP), Sentence (S) etc). The constituency tree for the sentence in Example (1) is illustrated in Figure (1a). In some linguistic traditions [25] it is common to define the grammatical function of an element by means of its tree position. For instance, here we can identify the leftmost NP under S as the sentence *subject*, and the rightmost NP daughter of a VP as the *object*.

Instead of deriving grammatical relations based on positions, it is possible to represent them explicitly. This is be done by means of *dependency structures* which follow on the rich linguistic tradition of [97]. A dependency structure is composed of binary arcs, each arc connects a pair of words. When the arc is labeled, the label defines the type of grammatical relation between the words. A dependency structure for sentence (1) is illustrated in Figure (1b). As seen here, labeled dependency structures deliver an explicit representation of the grammatical relations that define the *argument-structure*.[1]

Formally, we treat parsing as a *structured prediction* task, where $\mathscr{X}$ is a set of sentences in a language and $\mathscr{Y}$ is a set of parse-tree representations of sentences in the language. A parsing system implements a prediction function h from sentences to parse trees,

$$h : \mathscr{X} \rightarrow \mathscr{Y}. \tag{1}$$

[1] Constituency-based trees and dependency-based trees are the most common syntactic representation types that are produced by current parsing systems, and these are the structures we discuss in this chapter. It is however worth mentioning that there exist parsing systems that produce structures assigned by so-called *deep grammars*, such as Lexical-Functional Grammar (LFG, [12]) Head-Driven Phrase-Structure Grammars (HPSG [90]) and Combinatorial Categorial Grammars (CCG [96]). The methods that we discuss (data driven and statistical) can be applied effectively to these other types of graphs too. See, for instance, [47, 76, 16].

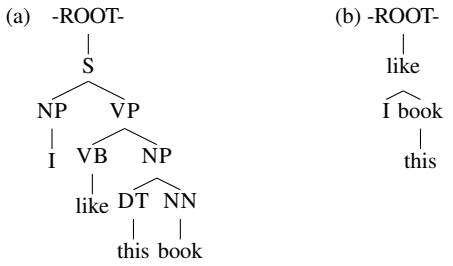

Fig. 1 (a) A phrase-structure tree for (1). (b) A dependency tree for (1).

Different parsing architectures can be thought of as different instantiations of h. For example, if $\mathscr{X}$ is the space of sentences in English and $\mathscr{Y}$ is the space of constituency trees with English words in their leaves, then (1) defines a constituency-based parser for English. If $\mathscr{X}$ is a set of sentences in French and $\mathscr{Y}$ is the set of dependency trees where internal nodes are French words, then (1) defines a dependency-based parser for French.

A pervasive problem in the automatic analysis of natural language sentences is ambiguity. A natural language sentence may be assigned different syntactic analyses, for example, the sentence "Time flies like an arrow" may admit at least the analyses demonstrated in Figure 2.

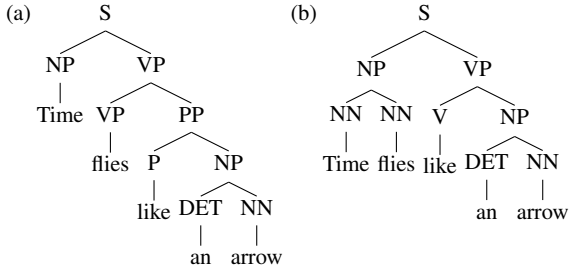

Fig. 2 Syntactic ambiguity for the sentence "Time flies like an arrow".

A parsing system aims to find a single syntactic analysis reflecting the human perceived interpretation of the sentence. That is, for the sentence "Time flies like an arrow" we would like to pick the (2a) analysis, while for the sentence "Fruit flies like a banana", though superficially similar, we would pick the analysis reflected in (2b).

Models and Algorithms

Given a finite set of annotated examples $\{\langle x,y\rangle | x \in \mathscr{X}, y \in \mathscr{Y}\}$ (henceforth, a *treebank*), a parsing model aims to induce a prediction function h such that $h(x') \in \mathscr{Y}$ is a parse-tree for $x' \in \mathscr{X}$. A parsing model is represented as a triplet where Γ is a set of constraints over the formal representation, λ is a set of model parameters and h is a decoding algorithm [59].

$$M = [\Gamma, \lambda, h] \tag{2}$$

The constraints in Γ define the form of the syntactic analysis, that is, whether they are dependency-trees or constituency-trees and what kind of tree structures are admissible. Since the syntactic analysis of a sentence is a complex structure, we cannot hope to induce a model that predicts a syntactic parse tree as a whole. A structured prediction system thus typically represents a complex structure by means of a set of simpler events that can be observed in annotated texts, and which can be used to construct novel analyses for unseen texts. These events may be the rules of a formal grammar [19], transitions in a state machine [79], pieces of the parse tree itself [10] and so on. The scores or weights of these events are the model parameters in λ, and they are estimated from annotated data based on corpus statistics.

Due to syntactic ambiguity, a sentence $x \in \mathscr{X}$ may admit more than one syntactic analysis. So a crucial task of the model is to disambiguate, to pick out the correct syntactic analysis for a particular sentence $x \in \mathscr{X}$. Let us constrain the set $\mathscr{Y}_x$ to be the set of all possible trees for a given sentence $x \in \mathscr{X}$. At the general case, h implements an algorithm that finds the highest scoring analysis or a given sentence.

$$h(x) = \text{argmax}_{y \in \mathscr{Y}_x} score(y) \tag{3}$$

The score of an analysis y is defined to be a multiplication of the model parameters λ_i with a feature vector $f(x,y)$ representing the events in the parse tree $y \in \mathscr{Y}_x$.

$$score(y) = \lambda_i f_i(x,y) \tag{4}$$

In order to implement a parsing system as in (3), we need to define the following:

- The modeled events: That is, the relevant pieces of information $f_i(x,y)$ that make up the parse tree, and can be observed in annotated data.
- The training algorithm: That is, a statistical method that is used to estimate the model parameters λ_i, that is, the weights of the modeled events.
- The decoding algorithm: That is, an algorithm that can traverse possible analyses $y \in \mathscr{Y}_x$ and pick out the highest scoring one for the sentence $x \in \mathscr{X}$.

The overall parsing architecture is depicted in Figure 3. A treebank enters a **training** phase in which the modeled events are observed and the model parameters are estimated. The estimated parameters are then fed into the **decoding** algorithm, which now accepts a new sentence as input. Based on the parameters and the formal constraints on the representation, it outputs the highest scoring analysis possible.

In order to evaluate how well a parser has performed we need to parse sentences that have not been seen during training, and compare the predictions with the correct analyses. We follow a common practice in machine learning where the annotated data (the treebank) is first split into a training set and a test set that are disjoint. The training algorithm is applied to the train set, and the decoding algorithm is applied to sentences for which we have gold trees. The part of the treebank reserved as a test set contains the gold analyses of the parsed sentences, and comparing parse hypotheses with gold trees allows us to quantify the parser's performance as scores (Figure 3).

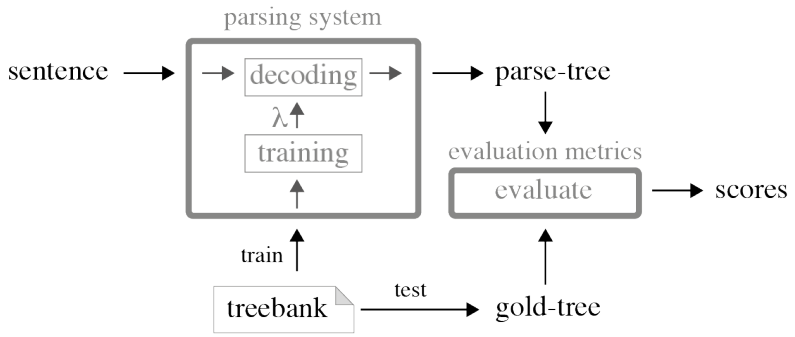

Fig. 3 An overall architecture for statistical data-driven parsing and evaluation.

This architecture presents a general framework in which data-driven parsing systems can be modeled, formalized and implemented. Currently available parsing systems can be roughly grouped into (i) grammar-based, (ii) transition-based and (iii) graph-based methods.[2] Let us characterize these systems in turn.

- Grammar-Based Parsing Systems

  A generative grammar is a formal finite system consisting of rules that can be applied sequentially in order to derive a parse tree. In generative grammar-based parsing, the modeled events are grammar rules. This means that the vector $f(x,y)$ for a parse tree $y \in \mathcal{Y}_x$ contains the rules that are used in deriving the tree according to the grammar, and λ_i are the probabilities of these rules, estimated based on their occurrences in the annotated data. When using such grammars the decoding algorithm is often based on dynamic programming, and it is designed to efficiently build, pack and explore the space of possible derivations of a sentence according to the grammar rules. Parsing models that are based on probabilistic

[2] These methods are orthogonal to the kind of formal syntactic representation being used. That is, grammar-based generative models can be used to generate constituency or dependency structures. Transition-based methods can be defined for constructing dependency graphs or constituency trees.

context-free grammars and dynamic programming lie at the backbone of the most accurate broad-coverage statistical parsers to date.

- Transition-Based Parsing Systems

  A transition-based system is a formal machine that defines states and possible transitions between them. The state describes the current state of the machine, and the transition defines an action to be applied next. The transition system defines a start state and a finish state, and transitions are defined such that every transition sequence from the initial state to the finish state corresponds to a set of actions that construct a permissible parse tree. In a transition-based system, the modeled events represented in $f(x,y)$ are state-transition pairs and the weights λ essentially score the different transitions in each state. Transition-based parsers are the fastest broad-coverage parsers to date. It is possible to construct a greedy decoding algorithm that, by selecting n transition between words, one word at a time, can predict a dependency parse tree in time complexity linear in the length of the input ($O(n)$).

- Graph-Based Parsing Systems

  In graph-based systems the modeled events are pieces or factors of the graph representation of the parse-tree, of which scores are estimated based on the occurrences of these factors in corpus data. The modeled events are then factors of the graph representation, and the weights represent how likely a factor is to occur and/or to combine with other factors. Such parsing systems typically assume a generative component that generates all possible parse trees $\mathscr{Y}_x$ for a given sentence x (usually in a packed representation) and the decoding algorithm explores the different possible trees and seeks the highest scoring one by aggregating the scores of the factors of each graph. The level of factoring determines to a large extent the complexity of the decoding algorithm, as well as the feasibility of training the weights using a given amount of annotated data.

It turns out that the challenges of parsing Semitic data transcend individual parsing paradigms and representational choices. In the next section we outline the main propertied of Semitic grammar and show how they challenge the general-purpose architectures we just described. We then focus on a particular case-study of generative methods for constituency-based parsing, and present modeling methods that adequately address the different kinds of challenges in the same general modeling architecture. Section 3 finally presents parsing results for Semitic languages using available resources, covering these three different parsing paradigms.

1.2 Semitic Languages

The Semitic language family is a group of related Afro-Asian languages that are spoken across the middle-east and North Africa. Semitic languages are spoken by 270 million native speakers nowadays, where the most widely-spread ones are Arabic (206 million), Amharic (27 million) and Hebrew (7 million). Many of the morphological Semitic phenomena, including the rich system of conjugations (e.g., *binyanim*), inflections and other syntactic constructions (e.g., *idafa/smixut*) are shared across the Proto-Semitic descendants. This section presents the orthographic, morphological and syntactic phenomena which pose challenges to the design of statistical parsing systems of the kinds we described above.

Script and Orthography

Semitic languages are written from right to left, they employ an alphabetic system that is based on consonants and omit some or all of the vowels. Semitic languages have their own scripts relying on consonants and omitting most or all vowels. Many Semitic languages are known for their notorious use of vocalization patterns, which are indicated by means of subscript or superscript diacritics. Diacritics are explicit in educational texts, but they are usually omitted in standard texts.

Take, for instance, the Hebrew form *fmnh*. It may admit at least three readings.[3]

(2) Hebrew
   a. shmena (‘fat’, ADJ.3Fem.Sing)
   b. shimna (‘lubricated’, VB.3Fem.Sing)
   c. she-mana (‘that’ + ‘counted’, VB.3Masc.Sing)

The lack of diacritics does not pose a problem for mature readers of the language because the correct analysis is easily resolvable based on context. In the following, for example, the inflectional features of the subject help to select the right analysis.

(3) Hebrew
   a. *hxtwlh fmnh*
      the-cat.FemSing gained.FemSing weight
   b. *hild fmnh*
      the-kid.MascSing that-counted.MascSing

The selected analysis in (3b) reveals further peculiarity of the Semitic writing system. In Semitic languages, many linguistic elements such as determiners, definite articles, subordinators and conjunction markers, do not appear on their own. They are concatenated as affixes to the next open-class word in the sentence. This means that a space-delimited token may contain different segments carrying their own part-of-speech tags, as in (4)-(5).

[3] We use the Hebrew transliteration of Sima’an et al. [95] and the Arabic transliteration of [46]. Linguistic examples are taken from, or inspired by Tsarfaty [100] and Shilon et al. [93].

(4) Hebrew
wkfmhbit
a. *w*/CC *kf*/REL *m*/PREP *h*/DET *bit*/NN
and/CC when/REL from/PREP the/DET house/NN

(5) Arabic
llqlm
a. *l*/PP *Al*/DET *qlm*/NN
l/for *Al*/the *qlm*/pen

Given a word-token out of context, there is massive ambiguity as to how it should be vocalized and segmented. This has far reaching consequences for syntactic analysis, because it means that before we can syntactically analyze the sentence the system should first disambiguate its correct morphological analysis in context. However, the disambiguation may need cues from syntactic context, to be later provided by the parser. Section 1.3 discusses the challenge of dealing with this apparent loop.

Morphology

The morphological component of a grammar describes the internal structure of words. As discussed in chapter 1, Semitic languages are known for their rich morphology, encompassing at least derivational and inflectional morphemes. Due to extensive morphosyntactic interactions, some morphological phenomena are directly relevant to the parsing system.

Derivational Morphology

In Semitic languages, consonantal roots are the main carrier of meaning. Nouns, Verbs and Adjectives in Semitic languages are derived by combining consonantal roots with a range of morphological templates that combine vocalization patterns and extra consonants. For instance, from the Hebrew root *k.t.b* (k.t.b, roughly "write") one can derive the nouns *ktb* (ketav, "script") and *mktb* (mikhtav, "a letter"), the verbs *lktwb* (likhtov, "to write") *lhktib* (lehacktiv, "to dictate") and *lhtktb* (lehitkatev, "to correspond"), and the adjective/participial *ktwb* (katuv, "written"). Semitic roots are not unrelated. The Arabic root *k.t.b.* ("write") can be used to derive the nouns *mktbp* (maktaba, 'desk'), *ktAb* (kita:b, 'book') and *mktb* (maktab, 'office'), the verbs *kAtb* (ka:taba, 'correspond with'), *Akttb* (iktataba, 'signed, donated') and the participials *katb* (ka:tib, 'writing, writer') and *mktwb* (maktub, 'written, letter'). Morphological templates have implication for the grammatical structure of the sentence, for instance, they carry information concerning the event structure and the type of participants involved in the predicate [32, 98]. In this chapter we treat each root+template combination as a holistic lexical element (henceforth, a *stem* or a *lexeme*) carrying its own POS tag, its lexical meaning and its subcategorization frame.

Inflectional Morphology

Each lexical element in Semitic languages may be realized in different word forms, corresponding to a set of inflectional features that are appropriate for the word in its syntactic context. The Hebrew verbal lexeme *ktb* ("write"), for instance, realizes gender, number and person features, as illustrated in 6. This set of forms is called a paradigm [3]. The sets of inflections that are realized as morphological features are similar across Semitic languages. In Hebrew and Arabic, verbs are inflected for gender, number and person, Arabic verbs are inflected for case, aspect and mood. Hebrew and Arabic nouns are inflected for gender and number. In addition, nouns and adjective are inflected to mark their so-called state (*smixut/idafa*), determining genitive relations in complex noun compound constructions.

(6) Hebrew

	Singular			Plural		
	1st	2nd	3rd	1st	2nd	3rd
Past						
Masculine	*ktbti*	*ktbt*	*ktb*	*ktbnw*	*ktbtm*	*ktbw*
Feminine		*ktbt*	*ktbh*		*ktbtn*	
Present						
Masculine		*kwtb*			*kwtbim*	
Feminine		*kwtbt*			*kwtbwt*	
Future						
Masculine	*aktwb*	*tktwb*	*iktwb*	*nktwb*	*tktbw*	*iktbw*
Feminine		*tktbu*	*tktwb*		*tktwbnh*	*tktwbnh*

The richness of the paradigms makes it hard to observe a all possible realization possibilities of a lexeme lexeme in the treebank. This leads to a high OOV (out of vocabulary) rate, which we discuss in section 2.4.

Syntax

The Semitic grammar makes use of the rich morphological marking to indicate, in systematic ways, how words combine to form grammatical phrases and sentences. In English, the order of words and phrases is a significant factor in determining how the words can be combined to deliver certain meanings. For instance, the subject precedes the verb and the direct object follows the verb. In Semitic languages, subjects, objects and modifiers are alternatively marked means of morphology. There are two main ways to mark grammatical functions morphologically. One is by *case assignment* on a word/phrase indicating its relation to the predicate, and the other is by *agreement* of two related elements on the inflectional features. Due to the rich system of morphological argument marking, the word-order patterns in Semitic clauses are less rigid than in English.

Word-Order

According to Greenberg [44], one of the most prominent dimensions of variation across the worlds' languages is their *basic word-order*, that is, the order in which the Subject, Verb and Object appear in a canonical sentence. In the proto-Semitic language the default word-order has been Verb-Subject-Object (VSO), which has been retained in Classical Arabic and Biblical Hebrew. In Modern Semitic languages including Modern Hebrew and many Arabic dialects this has given way to a Subject-Verb-Object (SVO) default structure, as is the case in English. Ethiopic Semitic languages follow a Subject-Object-Verb (SOV) order. This variation in word-order is not only attested across languages but also within the corpora of particular languages. In naturally occurring Semitic texts we can attest both SVO and VSO constructions, as well as V2 constructions in which a preposed object or modifier triggers subject/predicate inversion (similar to Germanic languages). This relative flexibility appears mostly at the clause level [94]. Within nominal phrases, we often find relatively fixed word order patterns. In Arabic and Hebrew nominals we find possessedpossessor (NG), and nounadjective (NA) orders. Modern Ethiopic languages have a possessorpossessed, and adjectivenoun order within noun phrases.

Case-Marking

Semitic languages utilize a case system in which the nominative, accusative and genitive cases are marked. Modern Standard Arabic maintains such case endings in literary and broadcasting contexts. Ethiopian languages preserved the accusative case ending. Modern Hebrew makes use of a differential Object marking system, in which objects are marked by the acc marker *at* if and only if they are also marked for definiteness [30]. Such explicit case marking allows to determine the grammatical function of a word or a phrase regardless of its position.

(7) Hebrew Object Marking
   a. *dni ntn mtnh ldinh*
      Dani gave present to-dina
      "Dani gave a present to Dina"
   b. *dni ntn at hmtnh ldinh*
      Dani gave ACC DEF-present to-Dina
      "Dani gave the present to Dina"

Semitic nouns and adjectives can be inflected for state, also known as *idafa* in Arabic or *smixut* in Hebrew. This is the morphological marking of genitive case, which can productively be used to create arbitrarily long genitive constructions.

(8) Construct State Nouns in Hebrew
   a. *bit hspr*
      house-of the-book
      school, lit: the house of the book

b. *mnhl bit hspr*
principal-of house-of the-book
The school principal

(9) The Idafa Construction in Arabic

a. *mdyr Almdrsp*
priciple-of the-school
The school principal

b. *Abn mdyr Almdrsp*
son-of priciple-of the-school
The school principal's son

It is often the case that definite marking (*nunation* in Arabic) occurs on the genitive head of this constructions. Due to the productivity of this process, can the definite marker can be arbitrarily distant from the genitive head.

Agreement

In agreement the inflectional feature values on two different elements in the sentence share their values in order to indicate a grammatical relation between them. Semitic languages exhibit patterns of agreement both at the clause level and phrase level. For instance, Semitic subjects agree with predicates on gender, number and person – which helps to track down the subject of the predicate – regardless of its position. Within noun phrases, nominal heads agree with their modifiers on gender, number and definiteness (definiteness may be morphologically marked, or inherent [30]).

Nominal Sentences

Semitic languages exhibit sentences that lack a verbal predicate altogether. In such sentences, the main predicate is realized by a nominal phrase, an adjectival phrase or a prepositional phrase, that define propertied of the main predicate.

(10) Hebrew

a. *dni mwrh*
Danny teacher
Danny is a teacher

b. *dni xkm*
Danny smart
Danny is smart

Nominal sentences can indicate identity sentences. In this case, there is a use of pronominal elements that act as a copular element, linking the subject to the nominal predicate. In the presence of pronominal elements, there is agreement between the subject and the copular element.

(11) Hebrew

a. *dni hwa mwrh*
Danny Pron.MascSing teacher.MascSing
Danny is a teacher

b. *dinh hia mwrh*
Dina Pron.FemSing teacher.FemSing
Dina is a teacher

Clitics

Pronouns indicating grammatical functions such as subject and object may be dropped if the verb is inflected to reflect the pronominal inflectional features.

(12) Hebrew

a. *raitih*
Saw.1Sing.3FemSing
I saw her

(13) Arabic

a. *rAythA*
Saw.1Sing.3FemSing
I saw her

All in all, a Semitic word contains information about the lexical meaning, argument structure, inflectional features, and one or more grammatical relations that are realized by the word. This information provides crucial cues concerning how the syntactic structure of a sentence has been constructed. In parsing, we wish to utilize such cues when we search for the overall syntactic analysis of the sentence (§1.3).

1.3 The Main Challenges

Whether our parsing architecture aims at a dependency-based or a constituency-based analysis, whether it relies on a generative grammar, a transition-based system or a graph-based discriminative approach, some basic assumptions inherent in the design of parsers for English break down when the system is applied for parsing Semitic languages. One such assumption is the notion of an input word, which is assumed to be in the yield of the parse tree. An additional assumption made in parsing English is that the position of words largely determines their grammatical roles and how they are allowed to combine. In Semitic languages non-adjacent words may interact in non-trivial ways due to complex morphological markings, so modeling assumptions based on word-position are less adequate. There is also an implicit assumption in the modeling methods for English that it is feasible to derive a comprehensive probabilistic lexicon from treebank data. Due to very high word form variation and high ambiguity, in Semitic languages this is not so. We hereby delineate the emerging challenges as the architectural challenge, the modeling challenge, and the lexical challenge.

The Architectural Challenge

In English and similar languages, a single word token represents a minimal unit of meaning. Due to the rich morphological structure of words of Semitic languages and their complex orthography (section 1.2), every space-delimited word-token may contain multiple units which carry independent grammatical roles. To illustrate, recall the Hebrew token in example (4). It contains multiple morphological units with their own syntactic roles, idicated as part-of-speech tags (and/CC when/REL from/PREP the/DT house/NN).

In order to parse a sentence in a Semitic language, the text has to go through morphological segmentation which uncovers the sequence of word segments that are combined into phrases and sentences. Morphological analysis in Semitic languages is however highly ambiguous due to the morphological variation, complex orthography, and the omission of diacritics. A Hebrew word like *bclm* may admit different analyses, as illustrated in (14).

(14) *bclm*
   a. *b*/IN *clm*/NN
   b. *b*/IN *h*/DT *clm*/NN
   c. *b*/IN *h*/DT *cl*/NN *fl*/POSS *hm*/PRN
   d. *bcl*/NN *fl*/POSS *hm*/PRN

Such ambiguity can only be resolved in the context at which the word token occurs. Sometimes local context, such as neighboring words, is sufficient for disambiguating the morphological analysis. In other times, a word that contains disambiguating cues may be arbitrarily distant from the word that needs to be disambiguated.

Let us illustrate how the existence of an agreeing element helps to pick out the correct morphological analysis of a distant word. A Hebrew word as *hneim* may appear in different phrases:

(15) a. *bclm hneim fl hecim*
in-the-shadow the-pleasant of the-trees
in the pleasant shadow of the trees

b. *bcl fl hecim hwa at zmninw hneim*
in-shadow of the-trees he ACC time-ours made-pleasant.MS1
In the shadow of the trees he made our time pleasant

In (15a), there is agreement on the definite article with the previous noun, which renders the correct analysis of *hneim* as a definite adjective. In (15b), there is agreement on a proceeding pronoun "he", which is the subject of the entire sentence, and which appears distant from the predicate due to the flexible order of phrases in Hebrew. This agreement helps us understand *hneim* as a verb, inflected to reflect the agreeing properties. So, in order to assign a syntactic analysis we need first to identify the correct morphological segments, but in order to pick out the correct morphological segmentation in context, we need information concerning the overall syntactic structure. How can we break out of this loop?

Computationally, there are two main strategies:

- **A Pipeline Architecture.** In a pipeline architecture. we first set up the morphological disambiguation component, and then provide the most probable morphological analysis of the sentence as input to a standard parsing model that aims to assign trees to the selected sequence of morphological segments.
- **A Joint Architecture.** An alternative way to break out of the loop is to build an architecture which selects the most probable morphological and syntactic analyses at once, as advocated by [99, 26, 39, 36].

The two types of architectures for parsing Semitic languages are sketched in Figures 4 and 5 respectively. The appealing property of a pipeline approach is its simplicity and modularity. The downside of a pipeline is that errors in the morphological disambiguation component may propagate to the parser and seriously undermine its prediction accuracy. The main advantage of the joint strategy is that the joint architecture helps to use one kind of information to disambiguate the other, and avoid error propagation. An apparent downside of this strategy is that it is more complex to design and implement. It may also make the search space over morphological and syntactic possible combinations a lot larger, which may in turn lead to parser inefficiency and search errors.

Correctly resolving the morphological ambiguity in the input is essential for obtaining a correct parse, but whether this morphological disambiguation should be done before or jointly with the parser is an empirical question. In section 2 we present an efficient lattice-based decoder that cater for a joint solution for syntactic and morphological disambiguation.

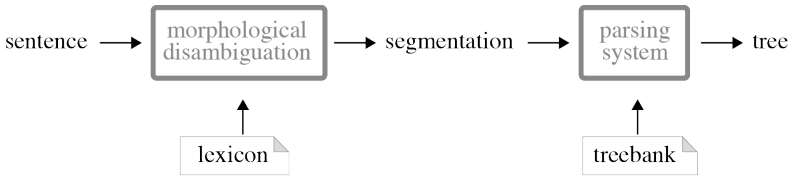

Fig. 4 A pipeline architecture for parsing Semitic languages.

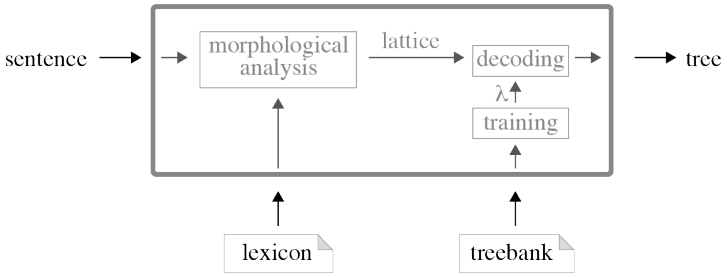

Fig. 5 A joint architecture for parsing Semitic languages.

The Modeling Challenge

A crucial factor in the modeling of a parsing system is the choice of modeled events. Intuitively, these events should allow us to learn the regular patterns of the language from annotated data and generalize from them. In practical terms, the way to ensure we pick out relevant regularities that lead to accurate predictions of a parsing model has to take into account the properties of the languages that we aim to parse.

Take, for instance, the notion of word-order in different languages. It is by now widely acknowledged that there exist different word ordering patterns in different languages, some of which more flexible than others. If we are dealing with a language with fixed word order, such as English, using the order of the different elements as model parameters may help us make very accurate predictions, because we will find repetitive patterns thereof in high frequencies. For languages with free word order patterns, picking word order patterns as model parameters will not necessarily help us in making precise predictions, since the correctness of a syntactic structure is dependent on other factors, and repetitive word-order patterns will not necessarily capture the same kind of grammatical information. In such languages we may want to parameterize other features; for instance, the morphology of words in the syntactic constructions, since they are suspected to exhibit more regular patterns in the data.

When we design a parsing model for a Semitic language, we have to take into account the following facts: (i) word order and morphological form jointly determine the predicate argument structure of sentences, (ii) word order may be flexible

for similar predicate-argument constructions, and (iii) word combinations may be licensed by matching morphological features, rather than their positions. In section 2.3 we survey different ways to parametrize such factors, implicitly or explicitly, in a generative parsing model. These factors should be likewise underscored when designing features in other parsing approaches.

The Lexical Challenge

We observed that the syntactic analysis of sentences requires us to know the correct morphological analysis of the morphological segments. The morphological analysis of Semitic words typically includes their lexical part-of-speech tags, prefixes, pronominal clitics, and various inflectional features that are reflected in the form of the words. We refer to this information as the *morphosyntactic representation (MSR)* or the *signature* of the word form in context. These MSRs present the interface between word level information and sentence syntactic structure.

One way in which we can learn how to assign MSRs to words in text is by observing the relative frequency of an analysis of a word form in annotated text. In languages such as English, there is not much variation in the form of words given their syntactic context. A word like ”worked” can occur with first, second or third person, singular or plural subjects, and we get enough attested occurrences of the verb to robustly estimate parameters based on corpus statistics. This is not the case in morphologically rich languages, and Semitic languages in particular, where a single lexeme for the past tense of ”worked” may be translated as a paradigm of seven word forms that realize additional inflectional features for gender, number and person (the complete paradigm includes seven forms *ebd, ebdti, ebdt, ebdnw, abdtn,ebdtm, ebdw*).

In Semitic languages, the observation of a word form in a particular syntactic environment does not help for analyzing the same lexeme in a different syntactic environment. Moreover, in languages such as English, the amount of annotated data appears to be sufficient for estimating the lexical parameters based the occurrence probabilities, but treebanks for languages that have been understudied have a fairly limited amount of annotated text, typically not enough for observing a sufficient number of lexical items to begin with.

In other settings, this problem is referred to as high OOV (out of vocabulary) rate, and it is frequent in domain adaptation scenarios where a trained parser is applied to text in a different domain and encounter a large number of lexical items from a new domain. Learning complex paradigms may potentially be an easier problem than analyzing out of domain OOV words, because we may have cues from different combinations of morphological signatures with previously observed stems or similar words. In §2.4 we survey methods that can be used for guessing and scoring the assignment of MSR in Semitic text for the purpose of more accurate parsing, using an external morphological analyzer and possibly with additional amounts of unannotated data.

1.4 Summary and Conclusion

The grammar of Semitic languages is different than that of English, in the sense that their word structure is a lot more complex and the syntactic structures are more flexible. These properties challenge existing parsing architectures in various ways. The parsing architecture has to handle complex, ambiguous input words. When constructing and scoring candidate parse trees, the model has to allow for flexible word-order patterns while ensuring the coherence of their morphological marking. From a machine learning point of view, a probabilistic lexicon in Semitic languages is harder to obtain, due to high word form variation and massive ambiguity.

The next section demonstrates how the model and feature design of a parsing system can effectively address these challenges. We show how morphological information can be disambiguated jointly with the parse tree, we demonstrate different ways to parametrizes a parser such that it can take into account, implicitly or explicitly, complex morphosyntactic interactions, and we also discuss how a parser can exploit additional resources (such as a dictionary and/or unannotated data) in order to handle word-form variation. We finally show how to evaluate parsers, taking into account both the morphological and syntactic disambiguation components.

2 Case Study: Generative Statistical Parsing

Probabilistic context free grammars (PCFGs) present the earliest and most commonly studied statistical models for generating constituency-based parse-trees. Despite theoretical claims concerning the inadequacy of context-free grammars for analyzing natural language data [92], statistical modeling based on probabilistic context free grammars lies at the backbone of many accurate statistical parsers for English [19, 29, 21, 82].[4] The conceptual simplicity and empirical viability of the framework has led to many adaptations, and several language-independent parsing frameworks have been developed based on it [6, 56, 82]. However, applying these 'ready-made' frameworks to parsing Semitic languages requires non-trivial adaptation or reconstruction in order to perform well on a Semitic language. In this chapter we describe building blocks of such models and demonstrate how to reconstruct the various components in ways that address the challenges we highlighted above. Addressing these challenges allows for more sophisticated statistical modeling which improves parsing results for Hebrew and Arabic significantly. Parsers for Amharic, Syriac or Maltese are relatively simple to derive based on this general framework presented here.

[4] The best reported result for English constituency parsing trained on the Penn treebank is now just above 92 F-Score [73].

2.1 Formal Preliminaries

We assume a parsing system that aims to predict, for any given sentence, a syntactic representation in the form of a constituency-based tree (Figure (1a)). We formally define the parser as a structured prediction function $h : \mathscr{X} \rightarrow \mathscr{Y}$ where $\mathscr{X}$ is a set of sentences in a language with a vocabulary $\mathscr{T}$, and $\mathscr{Y}$ be the set of linearly-ordered labeled trees with terminals drawn from $\mathscr{T}$. Due to syntactic ambiguity, an input sentence $x \in \mathscr{X}$ may admit multiple analyses $\mathscr{Y}_x$. In a probabilistic model, we aim to find the most probably parse tree given the input sentence:

$$h(x) = \text{argmax}_{y \in \mathscr{Y}_x} p(y|x) \tag{5}$$

We spell out (5) further by incorporating the definition of conditional probability and by observing that $p(x)$ is constant with respect to the maximization (8).

$$h(x) = \text{argmax}_{y \in \mathscr{Y}_x} p(y|x) \tag{6}$$

$$= \text{argmax}_{y \in \mathscr{Y}_x} \frac{p(y,x)}{p(x)} \tag{7}$$

$$= \text{argmax}_{y \in \mathscr{Y}_x} p(y,x) \tag{8}$$

In languages such as English, each sentence $x \in \mathscr{X}$ is contained in any tree $y \in \mathscr{Y}_x$, because the words stand as terminals in the tree. So we can simplify $h(x)$ further:

$$h(x) = \text{argmax}_{y \in \mathscr{Y}_x} p(y) \tag{9}$$

A parse tree y for an unseen sentence is a complex structure, too complex to be observed or learned directly from the training data. Therefore, a common strategy is to break down the construction of the tree into a finite sequence of decisions $d_1, ..., d_n$ such that $y = d_1 \circ \ldots \circ d_n$. If we assume that these decisions are independent, the probability of the tree equals the multiplication of the probabilities of the conditional decisions (9).

$$h(x) = \text{argmax}_{y \in \mathscr{Y}_x} p(d_1 \circ \ldots \circ d_n) \tag{10}$$

$$= \text{argmax}_{y \in \mathscr{Y}_x} p(d_1|\emptyset) \times \ldots \times p(d_n|d_1 \ldots d_{n-1}) \tag{11}$$

$$= \text{argmax}_{y \in \mathscr{Y}_x} \prod_{i=1}^{n} p(d_i|d_1 \ldots d_{i-1}) \tag{12}$$

The modeled events are the decisions d_i and the model parameters are the probabilities of applying each decision in a certain conditioning context. The conditioning context is narrowed down to relevant aspects of the tree generation. We indicate it by a function ψ that selects relevant features of the history.

$$h(x) = \text{argmax}_{y \in \mathscr{Y}_x} \prod_{i=1}^{n} p(d_i|\psi(d_1 \circ ... \circ d_{i-1})) \tag{13}$$

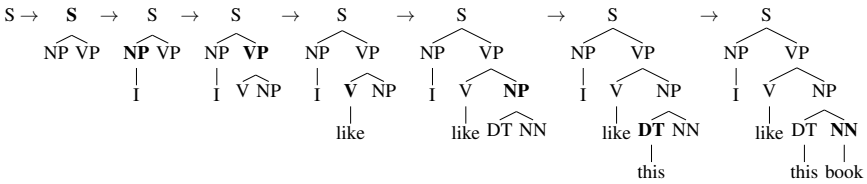

Fig. 6 A context free derivation of the sentence "I like this book". For each rule $\alpha \rightarrow \beta$, the substitution site α is in **bold**, and its dominated daughters form the β sequence.

Probabilistic Grammars

A constituency tree can be thought of as having been generated via a formal generative device that we call a *grammar*. Formally, a grammar G is a tuple

$$G = \langle \mathscr{T}, \mathscr{N}, S, \mathscr{R} \rangle$$

where $\mathscr{T}$ is a set of terminal symbols, $\mathscr{N}$ is a finite set of non-terminal symbols, $S \in \mathscr{N}$ is a designated start symbol, and $\mathscr{R}$ is a set of grammar rules. A *context free grammar* (CFG) has rules of the form $\alpha \rightarrow \beta$ where $\alpha \in \mathscr{N}$ is a non-terminal symbol and $\beta \in (\mathscr{T} \cup \mathscr{N})^*$ is an ordered sequence of terminals and non-terminal symbols. These rules indicate a substitution of a non-terminal symbol in the left hand side with the sequence of symbols at the right hand side of the rule. Such substitution is independent of the context of the non-terminal in the left hand side, and hence the name: context-free grammars [24].

A context-free grammar can be used to generate syntactic parse trees. Given a CFG we can obtain parse trees by means of sequentially applying rules from $\mathscr{R}$. The derivation of the sentence "I like this book" in Figure 1(a) is graphically depicted in Figure 6. When we fix a particular traversal order of trees, say, BFS, every context-free derivation corresponds to a unique tree and vice versa. That is, $\pi = r_1 \circ \ldots \circ r_n$.

We can extend this generative device into a probabilistic grammar that can assign non-zero probability to every sentence in the language generated by the grammar. A probabilistic context-free grammar is a CFG tuple extended with $p : \mathscr{R} \rightarrow [0,1]$, a probability mass function assigning probabilities to rules.

$$PCFG = \langle \mathscr{T}, \mathscr{N}, S, \mathscr{R}, p \rangle$$

The function p is defined such that $\forall r \in \mathscr{R} : 0 \leq p(r) \leq 1$. In order to guarantee assignment of probability mass to all sentences that may be generated by the grammar, the function p has to be such that the probability of all rules with the same symbol at the left-hand side sums up to one.

$$\sum_{\{\beta | \alpha \rightarrow \beta \in R\}} p(\alpha \rightarrow \beta) = 1 \tag{14}$$

The probability assigned to a tree by a PCFG is calculated as the probability of its unique derivation $\pi = r_1 \circ ... \circ r_n$. Due to context-freeness, the application of context-free rules is independent, so we can simply multiply rule probabilities.

$$p(y) = p(r_1 \circ ... \circ r_n) = p(r_1) \times ... \times p(r_n) = \prod_{i=1}^{n} p(r_i) \tag{15}$$

Our baseline probabilistic parsing model is now defined to be one that finds the most probable parse tree for a given sentence by computing the probability of its derivation according to a given PCFG. The decisions d_i are CFG rules, and the conditioning context is the label at the substitution site.

$$h(x) = \text{argmax}_{\{y \in \mathscr{Y}_x\}} \prod_{A \to \alpha \in y} p(A \to \alpha | A)$$

Decoding

Given a PCFG $G = \langle \mathscr{T}, \mathscr{N}, S, \mathscr{R}, p \rangle$ and an input sentence $x \in \mathscr{X}$, we need an algorithm that can find the most probable parse for a sentence according to the probabilistic model defined by the grammar. This phase is called *parsing* or *decoding*.

A naïve way to decode would be to enumerate all possible derivations of a sentence given the rules in $\mathscr{R}$, score them according to the probability model p, and pick out the analysis with the highest probability. However, since the number of analyses for a sentence of length n is exponential in n, this would be an inefficient way to decode.[5] In order to decode efficiently, we can store all parse trees in a two-dimensional chart and search for the highest probability parse-tree using dynamic programming methods. This is the idea behind the Cocke-Kasami-Younger (CKY) algorithm for parsing with PCFGs [53].[6]

A pre-condition for using the CKY algorithm is that each rule in the context-free grammar appears in one of the following two forms:

- Syntactic Rules: $A \to B\ C$; where $A, B, C \in \mathscr{N}$
- Lexical Rules: $A \to \alpha$; where $A \in \mathscr{N}$ and $\alpha \in \mathscr{T}$

Such a grammar is said to be in a Chomsky Normal Form (CNF) [24]. It can be formally shown that every PCFG can be uniquely converted into its CNF-equivalent by replacing a flat rule of n daughters with $n-1$ binary rules encoding intermediate steps in generating the sequence of daughters. In Figure (7a) we show how to assign a symbol for every intermediate step. Moreover, it is possible to convert a CFG into CNF by employing the Markov "limited history" assumption, where the right side of the rules records a limited number of previously generated sisters, as in Figure (7b).

[5] The catalan number $C_n = \prod_{k=2}^{n} \frac{n+k}{k}$ is the number of full binary trees with $n+1$ leaves.

[6] There are several algorithms that use dynamic programming and a two-dimensional chart. They can construct the tree top-down, bottom-up or left to right, they can use an agenda or not, and they can be exhaustive or greedy. Here we present an exhaustive bottom-up algorithm, used at the backbone of many state-of-the-art parsers, such as [56, 6, 82] and others.

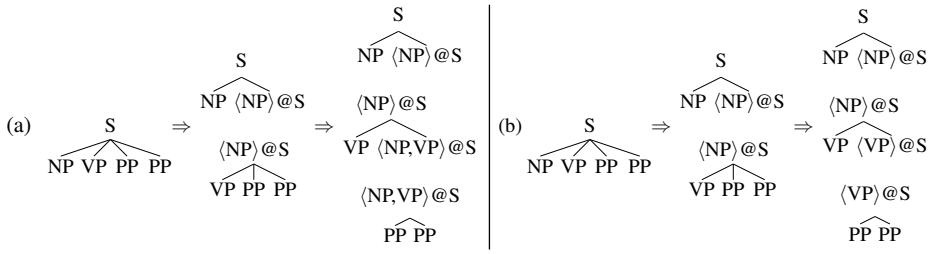

Fig. 7 (a) Conversion into Chomsky Normal Form (b) CNF 1st-order Markovization

The CKY algorithm assumes a data-structure called a chart which is a two-dimensional array of size $(n+1) \times (n+1)$ for a sentence of length n. We refer to a cell in the chart using its index pair $[i, j]$. Each cell may contain multiple chart-items of the form $A[i, j]$ where A is a non terminal symbol labeling the span between two indices $0 \leq i < j \leq n$ in the sentence.

We store a tree in a chart by storing its labeled spans in the relevant chart cells and recording the back-pointers to the labeled daughters that created this span, as shown in Figure 8. When we store multiple parse trees in one chart, there is a lot of overlap, such as the NP "an arrow" in Figure 9. This is why we can store to an exponential number of analyses for a sentence of length n, in polynomial space $O(n^2)$. Each chart cell can potentially store as many items as the numbers of non-terminals in the grammar. So in practice the space complexity of the algorithm is bounded by $O(n^2 \times |G|)$ where $|G|$ is the number of non-terminal symbols $|\mathscr{N}|$.

Algorithm 1 presents the execution of the CKY algorithm. Given a PCFG with a set of rules $\mathscr{R}$, a sentence of length n and an empty chart, the algorithm proceeds by filling in the chart with all the analyses that are licensed by the grammar, scoring them, and picking the most probable analysis.

In states 1-4 we fill in the labels that can span 1-length terminals according to the grammar. Let L be the index of a part-of-speech label $A_L \in \mathscr{N}$. For each terminal t_i, the δ values thus store the rule probabilities for each part of speech tag $A_L \in \mathscr{N}$.

$$\delta_L([i-1, i]) = p(A_L \rightarrow t_i | A_L)$$

In states 5-10 the algorithm considers sequences of length $1 < span \leq n$. For each cell $[i, j]$ it adds a label A_L into this chart item iff there exists an index $i < m < j$ and a rule $A_L \rightarrow A_J A_K \in \mathscr{R}$ such that $A_J[i, m]$ and $A_k[m, j]$ exist in the chart. For every label stored as a chart-item the algorithm stores the accumulated probability δ based on the maximal probability of obtaining the daughters:

$$\delta_L(i, j) \leftarrow \max_{\langle m,J,K \rangle} p(A_L \rightarrow A_J A_K | A_L) \times \delta_J(i, m) \times \delta_K(m, j)$$

For the purpose of recovering the most probable parse tree, the algorithm also keeps the back pointers of the most probable chart items to the chart cells using the β values. Once the chart is completely filled, the sentence is recognized by the grammar

if we find the start symbol $\delta_S[0, n+1]$ where S indexes the start symbol, and we can traverse the back-pointer indicating the most probable chart items that construct the most probable tree rooted in this symbol.

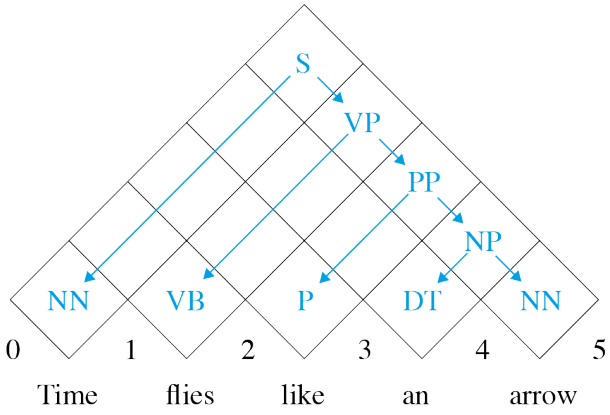

Fig. 8 The correct analysis of "Time flies like an arrow" stored in a chart.

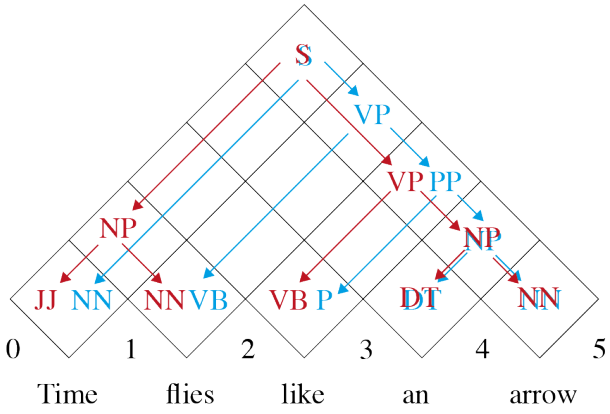

Fig. 9 The ambiguous analyses of "Time flies like an arrow" packed in a chart.

Algorithm 1 The CKY Algorithm for Chart Parsing (with Back-Pointers)

for $i = 1 \rightarrow n$ **do**
  for $L = 1 \rightarrow |\mathscr{N}|$ **do**
    $\delta_L[i-1,i] \leftarrow p(A_L \rightarrow w_i|A_L)$ ▷ Initiate pre-terminal probs
    $\beta_L[i] \leftarrow \langle w_i \rangle$ ▷ Store words
for $span = 2 \rightarrow n$ **do** ▷ Fill in the chart
  for $end = span \rightarrow n$ **do**
    $begin \leftarrow end - span$
    for $L = 1 \rightarrow |\mathscr{N}|$ **do**
      $\delta_L(begin,end) \leftarrow \max_{\langle m,J,K \rangle} p(A_L \rightarrow A_J A_K|A_L) \times \delta_J(begin,m) \times \delta_K(m,end)$
      $\beta_L(begin,end) \leftarrow \text{argmax}_{\langle m,J,K \rangle} p(A_L \rightarrow A_J A_K|A_L) \times \delta_J(begin,m) \times \delta_K(m,end)$
return RECONSTRUCT-TREE $(\delta_S[0,n], \beta_S[0,n])$ ▷ Follow back-pointers

Training

The CKY decoding algorithm makes notorious use of rule probabilities of the form $p(\alpha \rightarrow \beta|\alpha)$. In a data-driven setting like ours, we get both the grammar rules (the modeled events) and the parameters (their probability estimates) from a treebank containing sentences annotated with their correct constituency trees.

The trees in the treebank are decomposed into context-free rules, and the probability of each rule may be estimated using relative frequency counts. Formally, if $Count : \mathscr{R} \rightarrow \mathbb{N}$ indicating corpus counts of the rules occurrences, then:

$$\hat{p}(\alpha \rightarrow \beta|\alpha) = \frac{Count(\alpha \rightarrow \beta)}{\sum_\gamma Count(\alpha \rightarrow \gamma)}$$

Probabilistic grammars extracted in this way are called treebank grammars [19]. Every treebank grammar satisfies the condition that p assigns non-zero probabilities for every rule in the grammar, and the sum for all rules with the same left hand side equals to 1. This guarantees that we can assign non-zero probabilities for all the trees generated by the grammar for sentences drawn from $\mathscr{T}^*$. For every sentence in the language generated by the treebank grammar it now holds that:

$$0 \leq \hat{p}(x) \leq 1$$

It can be shown that this method of grammar extraction provide unbiased and consistent statistical estimates. That is, assuming that the corpus is generated by some context free grammar, if we had an infinite amount of annotated data, the probabilities estimated in this way will converge to the true probabilities in the grammar that generated the corpus [84].

Evaluation

The performance of a constituency based parsing model is standardly evaluated using the ParseEval measures [8], which calculate the precision and recall of labeled spans that have been correctly predicted. Formally, let us assume a sentence of length n. Now, let P be the set of labeled spans $\langle i, L, j \rangle$ in the parse-tree and let G be the set of labeled span in the gold tree for the sentence. We use the $\hat{C}$ term to denote only the root part of a tree C, that is, discarding the terminals and pre-terminal nodes. The notation $|x|$ indicates the number of tuples, discarding the root category. The precision, recall and F_1-Score (their harmonic means) are calculated as follows:

$$LP = \frac{|\hat{P} \cap \hat{G}|}{|\hat{P}|} \tag{16}$$

$$LR = \frac{|\hat{P} \cap \hat{G}|}{|\hat{G}|} \tag{17}$$

$$F_1 = \frac{2 \times LP \times LR}{LP + LR} \tag{18}$$

2.2 An Architecture for Parsing Semitic Languages

Preliminaries

We continue to assume that our parser implements a structured prediction function $h : \mathscr{X} \rightarrow \mathscr{Y}$, where each element $x \in \mathscr{X}$ is a sequence of space-delimited word tokens $x = w_1, \ldots, w_n$ from a set $\mathscr{W}$ and each output elements is a constituency tree. However, a major difference between parsing architectures for English and parsing Semitic languages is that Semitic words do not uniquely determine the segments that define the leaves of the parse tree (§1.3). Therefore, we have to alter our formal prediction function. Let $\mathscr{S}$ be a vocabulary containing finite set of valid morphological segments in the language, and let us continue to assume a finite set of non-terminal categories $\mathscr{N}$. Every parse tree $y \in \mathscr{Y}$ for a word sequence $x = w_1 \ldots w_n$ is a constituency tree over a sequence of segments $s_1 \ldots s_m$, where $(s_i \in \mathscr{S})$ and $n \neq m$.

We are then require to relate words $\mathscr{W}$ to valid morphological segments in $\mathscr{T}$. We assume a symbolic lexicon $\mathscr{L}$, which is simply a set of morphological segments associated with their corresponding part-of-speech tags. The tag labels are possibly fine-grained to reflect the values of inflectional features:

$$\mathscr{L} = \{\langle s, l \rangle | s \in \mathscr{S}, l \in \mathscr{N}\}$$

We demonstrated in Example (4) that a token $w_i \in \mathscr{W}$ may admit to multiple morphological segmentation possibiities. The set of analyses is constrained by a language-specific morphological analyzer $\mathscr{M}$. Formally, $\mathscr{M}$ is a function from words to sets of sequences of lexemes $\mathscr{M} : \mathscr{W} \rightarrow \mathscr{P}(\mathscr{L}^*)$.

The morphological analysis of an input word $\mathscr{M}(w_i)$ can be represented as a lattice L_i in which every arc corresponds to a specific lexeme $\langle s,l\rangle \in \mathscr{L}$. The morphological analysis of an entire input sequence $x = w_1 \ldots w_n$ is then a lattice L obtained through the concatenation of the lattices $\mathscr{M}(w_1) = L_1, \ldots, \mathscr{M}(w_n) = L_n$. An example for a complete morphological analysis lattice is shown in Figure 11.

We refer to a set of trees $y \in \mathscr{Y}$ which dominated sequences of segments which are contained in the morphological lattice $\mathscr{M}(x) = L$ as $\mathscr{Y}_L$. A parser for Semitic languages then has to predict both the correct sequence of morphological segments for the words and the parse tree that spans this sequence.

$$h(x) = argmax_{\{y \in \mathscr{Y}_L\}} p(y) \tag{19}$$

Joint Probabilistic Modeling

There are two different approaches to implement (19), as a pipeline scenario, or as a joint prediction. In a pipeline scenario, the morphological lattice L is provided as input to a morphological disambiguation component that aims to predict the correct segmentation of the input word tokens.

$$s_1^{m*} = \underset{s_1^m \in \mathscr{M}(w_1^n)}{\text{argmax}}\ p(s_1^m | w_1^n) \tag{20}$$

Then, the most probable segmentation s_1^m enters a standard parsing system which takes that morphological segments to be the tree terminals.

$$y^* = \underset{y \in \{y' : yield(y') = s_1^{n*}\}}{\text{argmax}}\ p(y | s_1^{m*}) \tag{21}$$

In a joint architecture, the lattice is provided as input to the parser, and the parsing algorithm aims to jointly predict the most probable parse tree, and the most probable segmentation in the morphological analysis lattice. Since the morphological lattice for every sequence of words is unique, we further can simplify:

$$y^* = \underset{y \in \mathscr{M}(x)}{\text{argmax}} p(y | \mathscr{M}(w_1^n)) = \underset{y \in \mathscr{M}(x)}{\text{argmax}} p(y) \tag{22}$$

This is the joint segmentation and parsing approach, advocated by [26, 39, 43]. If we assume that all the morphological paths in the lattice are equally likely, as in [39], we are in fact selecting both the path and the segmentation based on the probability of the trees only [39, 43]. It is also possible to assign probability to the tree taking into account scores indicating the likelihood of the different paths [26].

To illustrate, consider the morphological lattice of the Hebrew phrase BCLM HNEIM, as depicted in Figure 10. All trees in Figure 11 dominate paths in the lattice. The goal of a joint probabilistic model is to select the most probable tree according to the probabilistic model, and this tree will pick out a single path through the lattice.

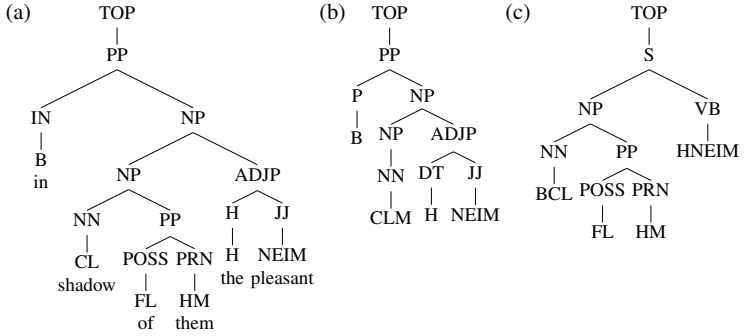

Fig. 10 Three possible parses of the phrase BCLM HNEIM. Tree (a) is the gold parse.

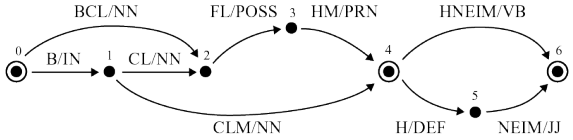

Fig. 11 Morphological segmentation possibilities of the phrase BCLM HNEIM. The pairs on each arc s/l indicate the segment s and a part of speech label l, as assigned by the lexicon $\mathscr{L}$.

Lattice-Based Decoding

Fortunately, we can still use the CKY decoding algorithm for finding the most probable tree over a lattice, by altering a few of its underlying assumptions [39]. We can no longer assume a chart of length n, since we do not know up front the number of morphological segments for $x = w_1^n$. We do know the morphological lattice, and we know that the the trees in $\mathscr{Y}_{\mathscr{M}(x)}$ will always span between states in the lattice $\mathscr{M}(x)$. So we can bound out chart size in the number of states in the morphological lattice $|\mathscr{M}(x)|$, and define the lattice states as the indices of the array. Some of the segments in the lattice inevitably have a span longer than one chart cell. At initiation time, we then fill the part-of-speech categories not for all span=1 cells as we did before, but we seek to fill in part of speech for all the segments in the lattice, even those that have a span greater than 1.

Consider for example the chart in Figure 12. It holds a tree that spans a path in the morphological lattice. Both the NN and the VB elements span more than one chart item, with no dominated daughters of span 1. In Figure 13 the span [4,6] can be covered by a single part of speech tag VB, or a phrase ADJP dominating two spans of length 1. At decoding time, we traverses all possible trees licensed by the grammar over all possible segmentations possibilities encoded in the lattice. The highest probability tree than further defines the choice among the segmentation alternatives. The pseudo code for the altered algorithm is given in Algorithm 2. Notebly, only the initiation phase (lines 1-3) has to be altered.

Algorithm 2 The CKY Algorithm for Lattice Parsing (with Back-Pointers)

for $\langle i, A_L, j \rangle \in \text{EDGES}(MA(x))$ **do** ▷ traverse the morphological analysis lattice
  $\delta_{A_L}[i,j] \leftarrow p(A_L \rightarrow s_{i,j})$ ▷ Initiate segments probs
  $\beta lattice_{A_L}[i,j] \leftarrow \langle s_{i,j} \rangle$ ▷ Store segments
for $span = 2 \rightarrow n$ **do** ▷ Fill in the chart
  for $end = span \rightarrow n$ **do**
    $begin \leftarrow end - span$
    for $L = 1 \rightarrow |\mathcal{N}|$ **do**
      $\delta_L(begin, end) \leftarrow \max_{\langle m,J,K \rangle} p(A_L \rightarrow A_J A_K) \times \delta_J(begin, m) \times \delta_K(m, end)$
      $\beta_L(begin, end) \leftarrow \text{argmax}_{\langle m,J,K \rangle} p(A_L \rightarrow A_J A_K) \times \delta_J(begin, m) \times \delta_K(m, end)$
return BUILD-TREE $\delta_S[0,n]), \beta_S[0,n]$ ▷ Follow back-pointers

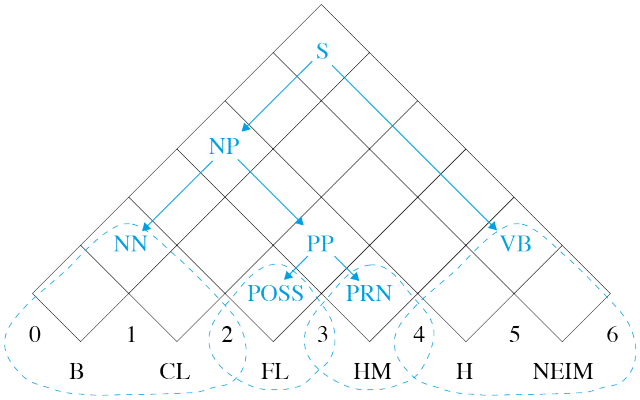

Fig. 12 Extended CKY Lattice Parsing: segment-based indexing of the phrase BCLM HNEIM.

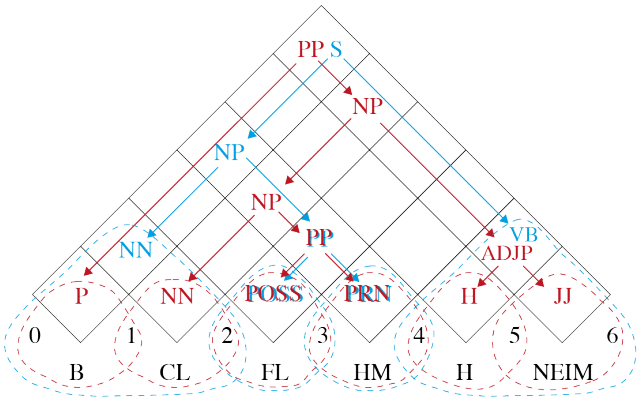

Fig. 13 Extended CKY Lattice Parsing: two possible analyses of the phrase BCLM HNEIM.

Evaluation

The structure and properties of Semitic languages pose challenges not only for parsing, but also for evaluating parser performance. Standard evaluation metrics for parsing assume identity between the input word sequence and the terminals in the tree. The ParseEval metrics for evaluating constituency-based parsing [8] assume that every tree can be represented as a set of labeled spans of the form $\langle i, L, j \rangle$ such that a labeled constituent L spans word indices $i < j$. if P and G are sets of such tuples for the gold and parse trees, the ParseEval scores are as follows.[7]

$$LP = \frac{|\hat{P} \cap \hat{G}|}{|\hat{P}|} \tag{23}$$

$$LR = \frac{|\hat{P} \cap \hat{G}|}{|\hat{G}|} \tag{24}$$

$$F_1 = \frac{2 \times LP \times LR}{LP + LR} \tag{25}$$

Applying ParseEval for cases where the parsed tree spans a different yield from that gold tree causes the evaluation procedure to break down. Consider for example the trees in Figure 14, which all correspond to the same original raw sentence BCLM HBEIM with the morphological lattice in Figure 11. The trees have segmentation hypotheses. The gold segmentation standard of BCLM HNEIM is more fine-grained then the coarse-grained segmentation hypothesis. The syntactic trees are very similar. Both of them the overall PP and the attached NP, but fails to identify the prepositional phrase "shadow of them" spanning indices 1–4. Figure 15 shows the word-based indexing of the gold tree and parse hypothesis. There are 5 syntactic nodes in the gold tree and 2 syntactic nodes in the gold tree, however the intersection of labeled spans in empty. The ParsEval score then calculates as $F_1(gold, tree) = 0$.

To circumvent this, early studies on parsing Modern Hebrew propose a generalized version of Parseval, in which tuples of the form $\langle i, L, j \rangle$ indicate labeled constituents where indices $i < j$ indicate character indices [99, 39]. This solution does not solve the problem adequately, since many morphological processes go beyond concatenation and the sequence of characters themselves may be misaligned. Figure 15 shows the character-based indexing of the gold tree and parse tree in our example. Because of an incorrect segmentation hypothesis, some phonological forms are not identified, and the sequences of characters are of different lengths. Again, the intersection of labeled spans is empty, and the ParseEval score is again 0.

In both cases, the metrics reduce the structures to sets. We can use edit distance operations to calculate the similarity of the trees themselves, rather than their set reduction. A few studies addresses this evaluation challenge by aiming to correct the segmentation using string edit operations, then score the parse tree dominating a corrected segment [89, 26]. This method of evaluation has been used successfully to evaluate parse trees over speech lattices, where the input signal is ambiguous.

[7] The standard implementation of ParseEval, Evalb, is available at http://nlp.cs.nyu.edu/evalb/

However, in both implementation, string edit correction may lead to discarding syntactic that were dominating removed segments, so the overall result may be biased. In our example, changing the parses segmentation into the gold segmentation involves deleting segment CLM and adding segments CL FL HM. The normalized string edit distance is then $\frac{1+3}{6+4} = 0.4$. Changing gold segmentation into the parsed one we would remove CL FL HM and add BCLM, and get the same segmentation error. The problem now is that in the first case the trees are much larger than in the second case, and the normalization of the tree scores are different. These normalization changes introduce noise and distort the empirical results which are based on such string-edit corrections.

To solve this challenge, Tsarfaty et al . [101] use tree-edit distance metrics to score trees over morphological lattices directly.[8]. TedEval requires a set $\mathscr{A}$ of operations such that, for every $y_1, y_2 \in \mathscr{Y}_{\S}$, there is some sequence of operations $\langle a_1, \ldots, a_m \rangle$ ($a_i \in \mathscr{A}$) turning y_1 into $y2$. It defines the cost of an operation sequence $\langle a_1, \ldots, a_m \rangle$ as the sum of the costs of all operations in the sequence $\textsc{c}(\langle a_1, ..., a_m \rangle) = \sum_{i=1}^{m} \textsc{c}(a_i)$ according to some cost function $\textsc{c} : \mathscr{A} \rightarrow \mathscr{R}^+$.

An *edit script* $\textsc{es}(y_1, y_2) = \langle a_1, \ldots, a_m \rangle$ is a sequence of operations from $\mathscr{A}$ that turns y_1 into y_2. The *minimum edit distance* $\textsc{med}(y_1, y_2)$ is the minimal cost edit scrip:

$$\textsc{med}(y_1, y_2) = \min_{\textsc{es}(y_1, y_2)} \textsc{c}(\textsc{es}(y_1, y_2))$$

The error of a predicted structure $p \in \mathscr{Y}_x$ with respect to a gold structure $g \in \mathscr{Y}_x$ is defined to be the edit distance $\textsc{med}(p, g)$. The score of a predicted structure p with respect to the gold g is defined by normalizing the distance and subtracting it from a unity:

$$\textsc{eval}(p, g) = 1 - \frac{\textsc{med}(p, g)}{\textsc{n}(p, g)}$$

The normalization factor $\textsc{n}(p, g)$ is normally defined as the worst possible MED given the sizes of p and g. In the current setting, this corresponds to the case where every all the nodes and lexemes in p have to be removed and all the nodes and lexemes in g have to be inserted, so we interpret the measure as the size of the tree ($|x|$ discards the root node, following common practice).

$$\textsc{eval}(p, g) = 1 - \frac{\textsc{med}(p, g)}{|p| + |g|}$$

The set $\mathscr{A}$ contains four edit operations: $\textsc{add}(l, i, j)$ and $\textsc{del}(l, i, j)$ that adds and deletes a nonterminal node with label $l \in \mathscr{N}$ that dominates the span from state i to state j in the lattice L; $\textsc{add}(\langle s, t \rangle, i, j)$ and $\textsc{del}(\langle s, t \rangle, i, j)$ that adds and deletes a lexeme, that is, a pair consisting of a preterminal node labeled $p \in N$ and a terminal node labeled $s \in T$. These operations are properly constrained by the lattice $\mathscr{M}(x)$, that is, we can only add and delete lexemes that belong to $\mathscr{L}$, and we can only add and delete them between states where they are allowed to occur in the lattice.

[8] http://stp.lingfil.uu.se/ tsarfaty/unipar/download.html

In our example, can use tree edit distance to turn the parsed hypothesis into the gold tree directly. For that we assume the lattice-based indices as depicted at the bottom of Figure 15. We now have to apply the following operations over the lattice-indexed lexemes and nodes: DEL(NNT/CLM,1,4), ADD(NN/CL,1,2), ADD(POSS/FL,2,3), ADD(PRN/HM,3,4), and ADD(PP,2,4). The number of nodes and lexemes in the gold tree is 11 and in the parsed hypothesis it is 7 (discarding the roots). TedEval thus provides the following score:

$$\text{EVAL}(p_1, g_1) = 1 - \frac{5}{7+11} = \frac{13}{18}$$

The TedEval score, taking into account both nodes and lexemes and normalizing for the size of trees with the corresponding segmentation, provides a less biased score than those that reduce structures into sets. In our example, the score given by TedEval reflects the fact that the system identified correctly two of the three major components in the sentence – the overall PP and a modified NP, and it avoids penalizing syntactic error over misaligned segmentation.

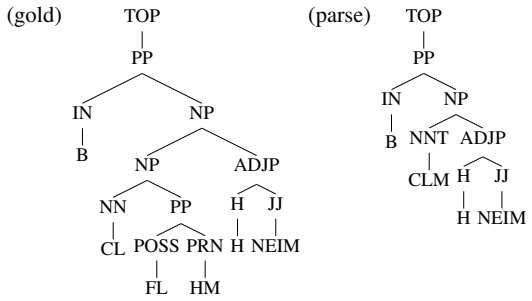

Fig. 14 A gold tree and a parse hypothesis for evaluation.

Summary and Conclusion

We have shown an architecture for parsing Semitic languages, in which we assume a morphologically ambiguous signal and predict the correct morphological segmentation along with the parse tree. We have further shown how to adapt the CKY decoding algorithm to such architectures, and demonstrated different ways which have been suggested for evaluating parsing results over ambiguous input. The architecture we presented here underlies the best constituency parsers for Modern Hebrew and Modern Standard Arabic. This architecture provides only a general framework for parser development. In the next section we show how to tune the grammar rules (and hence, the model parameters) so that they can effectively cope with rich morphosyntactic interactions reflected in the Semitic data.

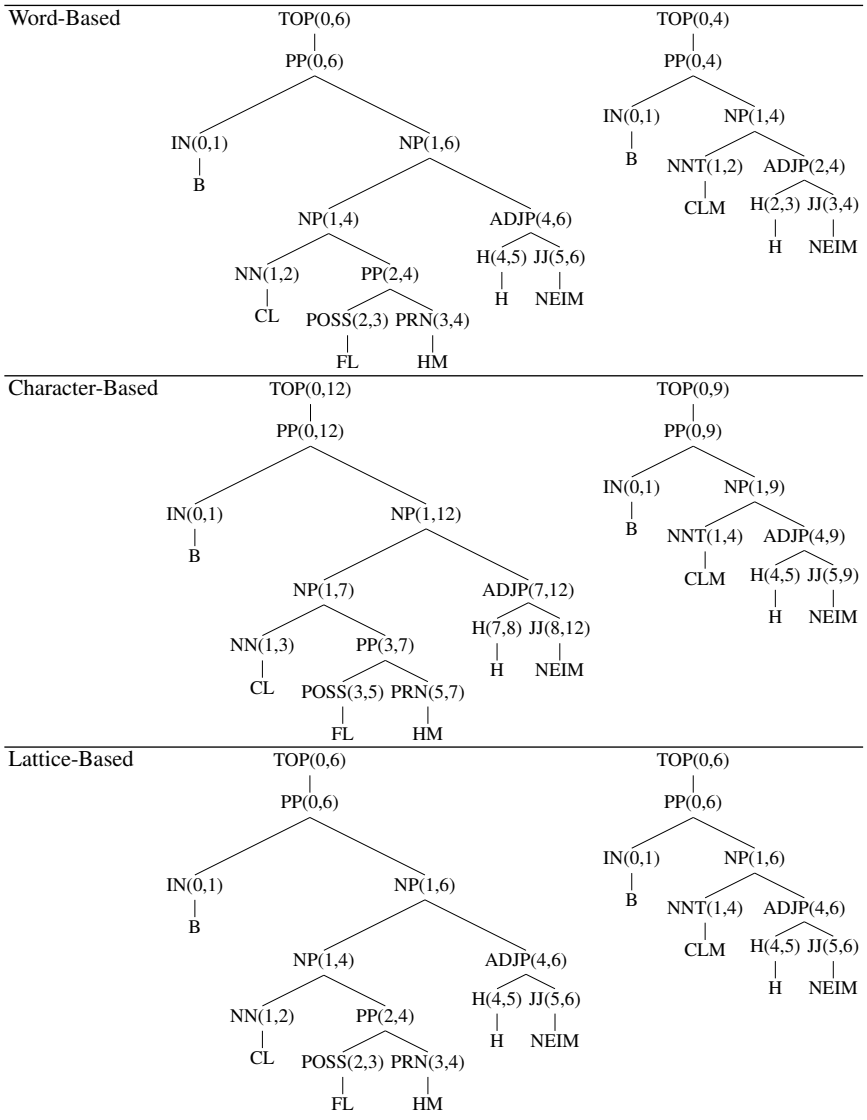

Fig. 15 Comparing the gold segmentation/tree to the segmentation/tree of the parse hypothesis. The different rows demonstrate different ways of indexing. The lattice-based indices refer to the lattice states in Figure 11.

2.3 The Syntactic Model

PCFG Refinements

The greatest limitation of PCFG-based statistical parsing models is the context-freeness inherent in the formalism [92]. This assumption imposes independence between model parameters — the probabilities of the rules that are used to derive the syntactic trees. These independence assumptions are both too strong and too weak for natural language data. Too strong because the context of a node has implications for what can appear at that position, and too weak because a node sequence at the right hand side of a rule cannot decompose and recombine to give category sequences which were unseen in the data. This section discusses techniques that effectively address these two types of limitations. We present methods that involve (semi)-manual tuning of model parameters, automatic tuning of model parameters, and a complete remodeling of the trees. We then discuss constrained parsing, and we finally comment on discriminative methods that discard such independence assumptions altogether. This section concentrates on theoretical foundations, the empirical results obtained by different models for Semitic languages are detailed in §3.

History-Based Modeling

In simple PCFG-based models, the modeled parameters are the weights of CFG rules of the form $A \rightarrow \alpha$. These parameters may be viewed as instantiating a history based model, where the decision d_i takes the form of substituting A into α, and the function ψ selects the symbol at the substitution site A as conditioning context.

$$h(x) = \text{argmax}_{y \in \mathscr{Y}_x} \prod_{A \rightarrow \alpha \in \pi} p(A \rightarrow \alpha | A) \tag{26}$$

The independence assumptions imposed by this choice of parameters are too strong for natural language parsing. Consider the sentence “He likes her” as annotated in Figure 16. The probability assigned to the tree according to the PCFG on the left is equal to the probability of the tree for the ungrammatical sentence “Her likes he” on the right. This PCFG model is then said to *over-generate*, that is, it assigns probability mass to sentences that should not be allowed by an English grammar.

Johnson [49] observed that it is easy to alter such independence assumptions by relabeling each category label with its parent category. By adding parent information, one enriches the conditioning context for the application of a rule, and this in turn creates a link between the generated daughter and higher levels of the tree. This modeling choice may be seen as a slightly more complex instantiation of the function ψ with the current label of its immediately dominating parent. (We use the X@A notation when indicating elements in the syntactic environment of a node A).

$$h(x) = \text{argmax}_{x \in \mathscr{Y}_x} \prod_{A \rightarrow \alpha \in x} p(A \rightarrow \alpha | A, \mathit{parent}@A) \tag{27}$$

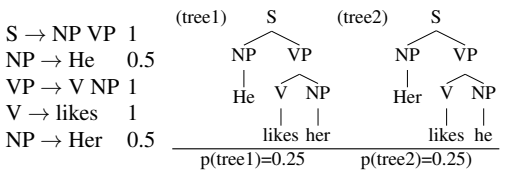

Fig. 16 Over generalization of a simple PCFG. Both (tree1) and (tree2) may be generated by the probabilistic grammar on the left, and they are assigned prob=0.25.

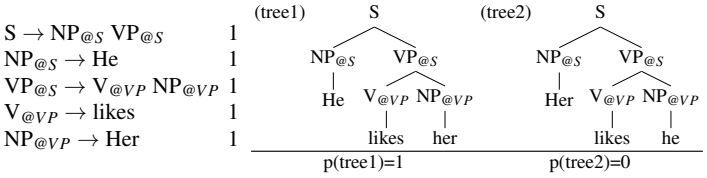

Fig. 17 Parent Annotation of a simple PCFG. The tree (tree1) is the only tree generated by the probabilistic grammar on the left. The tree (tree2) is assigned 0 probability.

In technical terms, Johnson proposes parsing a treebank variation as detailed in Algorithm 3. In this method we first transform the training trees to their new format, obtain a probabilistic grammar with the updated relative frequencies, use the grammar for parsing the test data, and detransform the parse trees back to their original representation prior to evaluation. For English parsing, it turns out that a parsing model obtained in this way improves the broad-coverage parsing of newswire texts substantially [49]. It was further shown by Klein and Manning [56] that annotating grand-parent information for some of the nodes in the treebank further improved English parsing accuracy.

Consider now the annotation of parent category for every node in the trees, as in Figure 17. Here the annotation distinguishes NPs that occur in subject position ($NP_{@S}$) from NPs that occur in object position ($NP_{@VP}$). A model can thus learn to generate the right kind of pronoun at each position of the tree. This helps to differentiate the grammatical sentence on the left from the ungrammatical sentence on the right of Figure 17, which is now assigned zero probability mass.

Would this modeling choice have the same effect when parsing another language? As it turns out, this partially depends on the properties of the language. Recall that we observed that Semitic languages have a more flexible phrase structure than sentences in English. Thus, the Hebrew version of the phrase for "I like her" could occur as *ani awhb awth* as well as *awth ani awhb*, *awhb ani awth* as is annotated in the right hand side of Figure 18. When applying the parent annotation to the Hebrew trees in this example, as in Figure 19, it adds no significant information to the probability model. Because of the flexible phrase structure, the different phrases hang directly under S and the parent annotation does not help to distinguish the distribution of phrases in different positions.

Algorithm 3 The Transform-Detransform method for history-based PCFG parsing

```
1: trans-trainset ← TRANSFORM(original-trainset)        ▷ enrich treebank trees
2: grammar ← TRAIN(trans-trainset)                      ▷ obtain a probabilistic grammar
3: for i = 1 → n do                                     ▷ traverse the test-set
4:     tree_i ← DECODE(grammar, x_i)                    ▷ parse the input sentence
5:     y_i ← DE-TRANSFORM(tree_i)                       ▷ discard enrichment
6: return {y_i}_1^n                                     ▷ return prediction
```

Empirically, this parent encoding has shown to provide some improvement on parsing Semitic languages though to a lesser extent than in parsing English (see section §3), because it mainly benefited those structures that exhibit stricter word-order – for instance, modified noun phrases. In order to parse Semitic languages more accurately, one has to provide a parsing method which is robust in the face of order freeness in the language and long flat productions of CFG rules.

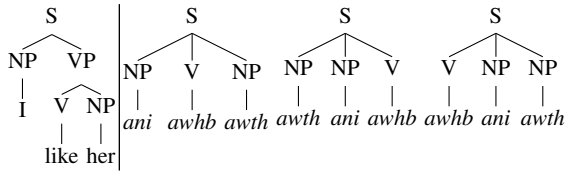

Fig. 18 Syntactic Analyses in English and Hebrew transitive sentences.

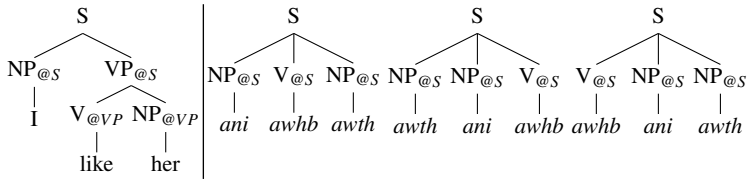

Fig. 19 Parent Annotation Features in English and Hebrew transitive sentences.

Head-Driven Models

The capacity to generate trees with flexible phrase structure crucially depends on the ability of a grammar to generalize from observed examples to new rules. Assume for instance that during training we observed the parse tree for *ani awhb awth* which uses the rule S → NP VP NP. Also assume that the decoder needs to find the correct parse for the input sentence *awth ani awhb* (her I like) which involves the structure S → NP NP VP . If we have not seen this rule permutation in the training data,

we will not be able to generalize from the former structure to the latter. This is a problem of *under-generation*, and it leads to a lack of *coverage*. A common way to solve it is to break down the generation of standard rules into smaller pieces, (much like we do in binarization §2.1), but define pieces that then may recombine to form rules that were unseen during training.

To solve this problem for English, Collins and others [67, 29, 20] proposed the *Head-Driven* modeling method. In this modeling method we view a context free rule of the form $A \rightarrow \alpha$, as a complex object where we distinguish a *head* element H from its left and right sisters.[9]

$$A \rightarrow L_n ... L_1 H R_1 ... R_m$$

Head-driven models, as proposed by [67, 20, 27], break down the generation of the daughters sequence into incremental steps of generating sisters from the head outwards and assuming independence between the generation steps.

$$h(x) = \text{argmax}_{x \in \mathscr{Y}_x} \prod_{A \rightarrow \alpha \in x} p(A \rightarrow L_n ... L_1 H R_1 ... R_m | A) \tag{28}$$

$$= \text{argmax}_{x \in \mathscr{Y}_x} \prod_{A \rightarrow \alpha \in x} p(H|A) \times \prod_{i=1}^{n} p(L_i | A, H, L_1 .. L_{i-1}) \times \prod_{i=1}^{m} p(R_i | A, H, R_1 .. R_{i-1}) \tag{29}$$

$$= \text{argmax}_{x \in \mathscr{Y}_x} \prod_{A \rightarrow \alpha \in x} p(H|A) \times \prod_{i-1}^{n} p(L_i | \psi_L(A, H, L_1 .. L_{i-1})) \times \prod_{i=1}^{m} p(R_i | \psi_R(A, H, R_1 .. R_{i-1})) \tag{30}$$

The model in (30) is in fact another instance of history-based modeling, where we generate one node at the time. For each node generation we specify conditioning context as the relevant information from amongst these sisters, by defining a history-mapping function ψ over the sisters that we have already generated. For instance, the ψ functions can select immediately proceeding daughters in a Markovian sister generation process. A 0-order Markovized head-driven process will proceed as follows, considering only the parent and the head daughter as conditioning context.

$$h(x) = \text{argmax}_{x \in \mathscr{Y}_x} \prod_{A \rightarrow \alpha \in x} p(H|A) \times \prod_{i=1}^{n} p(L_i | A, H) \times \prod_{i=1}^{m} p(R_i | A, H) \tag{31}$$

A 1st-order Markovized process records one immediately preceding daughter in addition to the head, a 2nd-order process records two immediately preceding daughters, and so on. Higher order Markovian processes are also conceivable but they impose stronger dependency between the different daughters of a CFG rules, which

[9] A head is a linguistic notion coined in the generative linguistic tradition. For instance, in the rule $VP \rightarrow V\ NP$, the verb is considered the most important. The criteria for selecting heads are subject for debates (see, for instance, the discussion in [107]), here we leave head selection criteria out of the discussion and assume that deterministic rules for finding heads can be defined.

greatly undermines its generalization capacity.[10] In the original head-driven models, each upper case symbol A, H, L_i, R_i in fact signals the category of the node and the lexical head information. In unlexicalized parsing, lexical items are omitted, and only the categories remain.

Head-driven lexicalized parsing has earned the language-independent implementation of Bikel [6], which was later applied to many different languages. As it turns out, however, the benefits of this modeling method have not been as significant for Semitic languages parsing as they were for parsing English (see section §3). Various reasons for that have been isolated, for instance, lack of sufficient training data, annotation inconsistencies, and inherent bias of evaluation metrics. At the same time, a more fundamental cause had been isolated. Because of flexible word-ordering patterns, choosing model parameters that distinguish left and right generation does not help to pick out the correct daughters in the right context. As can be seen in Figure 20 there is no significant correlation between the position of Noun phrases and the form of the pronouns. A more helpful enrichment here could be, for instance, encoding morphological case, as in Figure 21, which is a better predictor for the form of pronoun that has to be used.

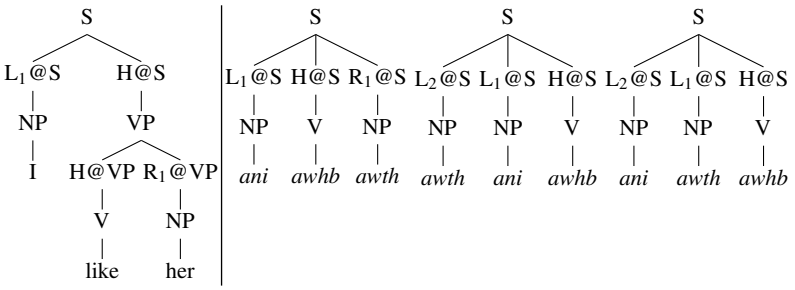

Fig. 20 Head-outward generation of nodes in English and Hebrew trees.

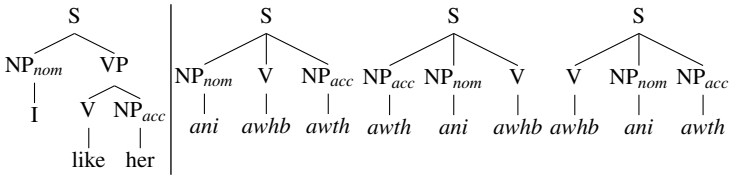

Fig. 21 Morphological features of syntactic nodes in English and Hebrew trees.

[10] Additional information may also be added to the ψ function. Collins [29] has included the presence of punctuation and verbal elements in the conditioning context of model 1, and information concerning subcategorization frames (the expected sisters of the verb) in model 2. In all cases, the model continues to assume a strict separation between generating the left and right daughters.

Three-dimensional Parameterization

Parent encoding on top of syntactic categories and Markovization of CFG rule productions are two instances of the same idea, that of encoding the generation history of a node. The different history-mapping functions ϕ and ψ correspond to two dimensions that define the parameters' space for parsers [56]. The *vertical* dimension (v), capturing the history of the node's ancestors (its parent and grandparents), and the *horizontal* dimension (h), capturing the previously generated horizontal ancestors of a node (its sisters) in a head-outward generation process. By varying the value of h and v along this two-dimensional grid and following the recipe of Algorithm 3, one can change the independence assumptions inherent in the treebank grammar. Klein and Manning showed that both horizontal and vertical history can improve parsing performance of unlexicalized treebank grammars considerably [56].

The two-dimensional space provides information concerning the position of a node in the tree. In the case of Semitic languages, the presence or absence of a morphological feature may substantially affect the probability of generating nodes in these positions. Adding morphological information to syntactic category labels is technically referred to as a *state-split*. In English, for instance, VBS distinguishes singular verbs from other types of verbs, and VBG marks gerunds. Both VBG and VBS are state-splits of the verb category VB. On the one hand they share something with the VB category, but they exhibit different behavior in different syntactic contexts.

Tsarfaty and Sima'an [103] describe a universe in which bare-category skeletons of Hebrew constituency trees reside in a bi-dimensional parametrization space in which the vertical and horizontal parameter choice elaborate the generation history of each non-terminal node. Specialized structures enriched with morphological features reside deeper along a third dimension they refer to as *depth*. This dimension can be thought of as encoding aspects of morphological analysis orthogonal to the node's position. Agreement features, for instance, apply to a set of nodes all at once. Figure 22 illustrates an instantiation of $depth$ with a single definiteness (**D**) feature. Figure 23 selects an NP node from Figure 22 and shows its expansion in three dimensions.

In the history-based perspective, we view the function ξ as selecting morphological aspects of the structure generated so far.

$$h(x) = \text{argmax}_{x \in \mathscr{Y}_x} \prod_{d_i \in x} p(d_i | \phi(d_0 \circ ... \circ d_{i-1}), \psi(d_0 \circ ... \circ d_{i-1}), \xi(d_0 \circ ... \circ d_{i-1})) \tag{32}$$

In parsing engines such as [29, 6] it is often the case that $\xi(d_0 \circ .. \circ d_k) = \emptyset$, because English is characterized by impoverished morphology. For Semitic languages, where morphological information interacts with syntactic structures more closely, discarding such morphological state-splits runs the risk of loosing important disambiguating information.

There are different ways in which such morphological state-splits may be explicitly incorporated into the parsing architecture.

- **Treebank grammars**. This technique involves repeating Algorithm 3 for different combinations of different sorts of conditioning information, and manually (or semi-automatically) tuning the choice of parameter enrichment on a development set. This is a simple and intuitive way to set a strong baseline. However it relies on linguistic intuitions and may be labour intensive and time consuming.
- **Probabilistic feature grammars.** Goodman [41] proposes a history-based model in which features are generated one by one, conditioned on previously generated features. The choice of features, their order and the independence between them are determined by a human expert and are tuned on a development set.
- **Coarse-to-fine parsing.** The coarse-to-fine parsing scheme is a popular way to approach decoding when a huge space of split categories is defined [22]. In this approach one learns multiple treebank grammars with increasing depth. The first decoding round uses a coarse-grained treebank grammar, and later parses uses the more elaborate treebank grammars, but the search is constrained by the most probable projected structures. This method avoids the need for human interference. However, it runs the risk that good parses may be pruned too early.

The success of such methods relies, firstly, on explicitly annotating this information in the training data. Moreover, certain feature engineering or dimensionality reduction is required in case our data set is too small to statistically learn parameters with such increased variance.

Automatic State-Splits

We have so far exploited the multidimensional space of parameters by explicitly augmenting node labels with information concerning position, morphology, lexical information, and so on. These techniques require the parser developer to explicitly define the different dimensions, annotate the information in treebank data, and retrain the treebank grammar. This approach is simple and intuitively appealing, but it requires investment of human effort, firstly in adding extra annotations, and secondly in order to find the optimal combination of annotated features. Is it possible to search for an optimal set of parameters automatically, without human intervention?

Studies such as [72, 85, 82] explore the possibility of employing general machine learning techniques in order to automatically find grammars with performance-improving state-splits. These grammars assume that each non-terminal symbol L can have different instantiations (splits) $L_1, L_2, L_3 \ldots$ which are used in different contexts. These split-states encode latent information, which is hidden from the point of view of the model. The only split criteria that matter are those that empirically improve parsing performance on the original coarse-grain categories. These state-split grammars are also called latent annotation PCFGs (in short, PCFG-LA). The technique of obtaining accurate PCFG-LA parsers has been mastered and excelled by Petrov et al. [82], delivering a language-independent parsing framework.

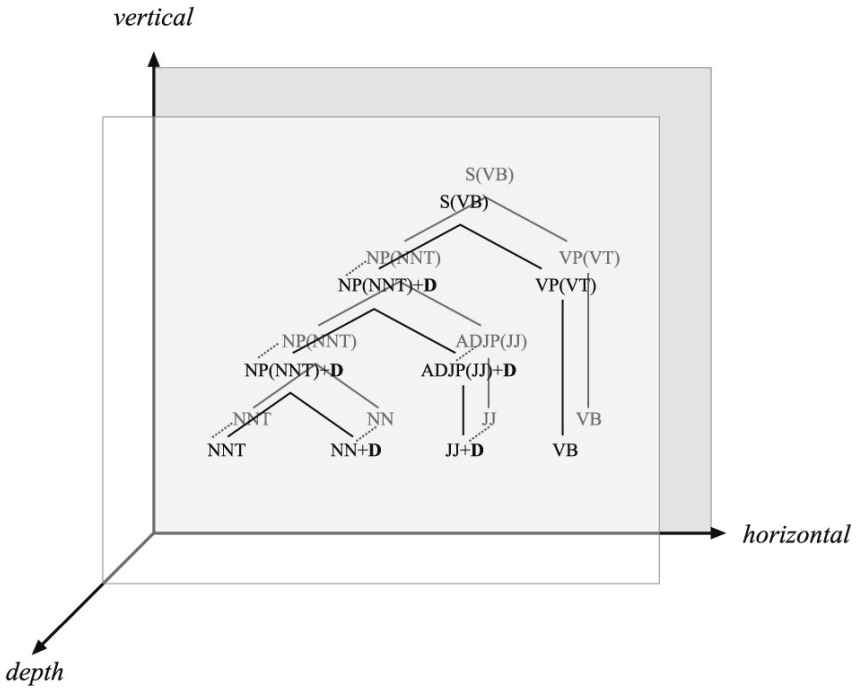

Fig. 22 The Three-Dimensional Parametrization Space

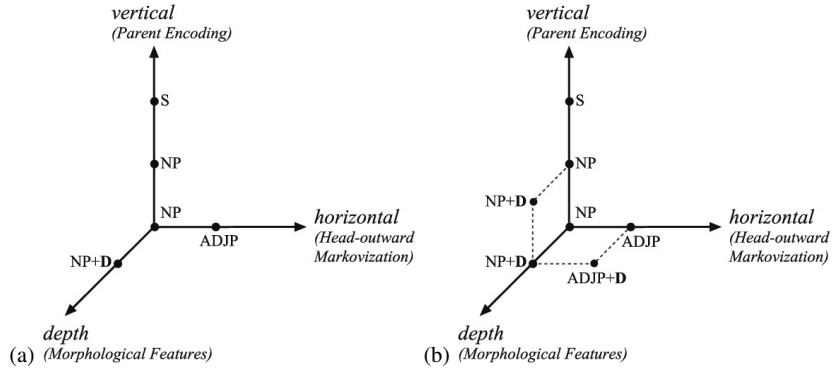

Fig. 23 (a) Local history of node generation in three dimensions. (b) An orthogonal dimension of correlated splits: Looking at a complex NP phrase in a three-dimensional Space

In the training phase the model can learn an accurate state-split grammar from a given treebank. It then uses a coarse-to-fine decoding algorithm to efficiently parse unseen data. Training a state-split grammar in the system of Petrov et al. [82] starts off with completely binarized treebank with bare phrase labels. It then follows a cycle that repeats the following steps: (i) split (ii) merge, and (iii) smooth.

- At the **split** phase, every existing label is split into two categories. A rule of the form $A \rightarrow B\ C$ now has eight instantiations: $A_1 \rightarrow B_1\ C_1$, $A_1 \rightarrow B_1\ C_2$, $A_1 \rightarrow B_2\ C_1$, $A_1 \rightarrow B_2\ C_2$, $A_2 \rightarrow B_1\ C_1$, $A_2 \rightarrow B_1\ C_2$, $A_2 \rightarrow B_2\ C_1$, $A_2 \rightarrow B_2\ C_2$.
  An Expectation Maximization procedure [84] then applies to set the split-rule probabilities. In the Expectation step, one computes the posterior probability of each annotated rule in each position in each training tree. In the Maximization step, these posterior probabilities as used as weighted observations, and updates the rule probabilities to their observed relative frequencies.
- Not all splits are empirically equally useful. Furthermore, too many splits may lead to over-fitting. At the **merge** phase, state-splits that are found not to be useful according to an empirical gain criteria are merged back. Merging is followed by an EM procedure that sets the probabilities of the rules in the merged grammar.
- While the grammar considers the label-splits as disjoint states, they still share some of their behavior with the original L type category. So, at the **smooth** phase some of the probability of the split labels L_1, L_2 etc. is passed back to the original label L, shrinking the distribution of the L_i labels towards their base symbol.

At decoding time, the score of each split rule is calculated by marginalizing out its splits. For efficiency reasons, multi-level coarse-to-fine pruning is used [83].

The PCFG-LA implementation of Petrov et al. [82] is publicly available[11] and has been applied to different languages (English, German, Chinese, French and Arabic) obtaining good results. Applying it out of the box for Semitic language data deserves certain extra considerations:

- It has been repeatedly shown that using the training procedure over a refined grammar does not yield major improvements, because the training procedure cannot merge splits that were not perform by the system. So the treebank labels that one starts with has to be sufficiently coarse.
- The implementation assumes a procedure for treating unknown words smoothing [82, footnote 3]. This method proved useful for English, where upper case, digits and suffixes are good predictors of the syntactic category. Semitic languages show a different behavior, and it is worthwhile to add a specialized unknown words treatment which is more appropriate [77].
- The default training setup assumes 2nd order vertical Markovization and 1st order horizontal Markovization, But for languages with more flexible ordering it appears beneficial to set 1st-order vertical Markovization and 0th-order horizontal Markovization [35]. The generation of a daughter is then conditioned on the parent only, and the state-splits implicitly alter the independence between sisters.

[11] The Java implementation can be downloaded from http://code.google.com/p/berkeleyparser/. The extended implementation including lattice parsing was made available by Goldberg and Elhadad [36] and can be found at http://www.cs.bgu.ac.il/ yoavg/software/blatt/.

Relational-Realizational Parsing

The previous modeling strategies we discussed assume that the probabilistic grammar defines the probabilities of phrase-structure trees, and hand-coded constraints encode linguistic (hard or soft) constraints and derive their grammatical functions. It is possible to define the probabilistic grammar itself as a set of soft morphosyntactic constraints that ensures coherent predicate argument structure for every phrase structure tree. The benefits of such approach are twofold. Firstly, the functional information will help to improve the accuracy of the prediction. Secondly, the predicate argument structure may be read off directly from the trees, without the need to post-process the data. This is the motivation underlying the Relational-Realizational modeling framework proposed by Tsarfaty [100] and applied to modeling Semitic phenomena in Tsarfaty et al. [104, 106, 105].

The Relational-Realizational model assumes a grammar called RR-PCFG which is a tuple $RR = \langle \mathscr{T}, \mathscr{N}, \mathscr{GR}, S \in \mathscr{N}, \mathscr{R} \rangle$ where PCFG is extended with a finite set of grammatical relations labels $\mathscr{GR} = \{gr_1, gr_2, ...\}$ such as subject, predicate, object, modifier etc. ($\mathscr{GR} \cap \mathscr{N} = \emptyset$). The set $\mathscr{R}$ contains three kinds of rules which alternate the generation of structural and functional notions:

$$\mathscr{R} = \mathscr{R}_{projection} \cup \mathscr{R}_{configuration} \cup \mathscr{R}_{realization}$$

Let us assume that $C_P \rightarrow C \dots C$ is a context free production in the original phrase structure tree where each daughter constituent $C_i \in \mathscr{N}$ has the grammatical relation $gr_i \in \mathscr{GR}$ to the parent category. The set $gr_1 ... gr_n$ contains is defined as the *relational network* (a.k.a. the *projection* or *subcatgorization*) of the parent categories. Each grammatical relation $gr_i @ C_P$ is realized as a syntactic category C_i and carry morphological marking appropriate to signaling this grammatical relation. The RR grammar articulates the generation of such a construction in three phases:

- **Projection:** $C_p \rightarrow \{gr_i\}_{i=1}^{n} @ C_P$
  The projection stage generates the function of a constituent as a set of grammatical relations (henceforth, the relational network) between its subconstituents.
- **Configuration:** $\{gr_i\}_{i=1}^{n} @ C_P \rightarrow gr_1 @ C_P \dots gr_n @ C_P$
  The configuration stage orders the grammatical relations in the relational network in a linear order in which they occur on the surface.[12]
- **Realization:** $gr_i @ C_P \rightarrow C_i$
  In realization, every grammatical relation generates the morphosyntactic representation of the child constituent that realizes the already-generated function.

The RR model is recursive, where every generated constituent node is a root for the repeated cycle. Every time the projection-configuration-realization cycle is applied, the probability of this construction is replaced with with the probabilities of the three stages, multiplied:

[12] The configuration phase can place realizational slots between the ordered elements signaling periphrastic adjunction and/or punctuation. Modification may generate more than one constituent. See further details in the original publication [104].

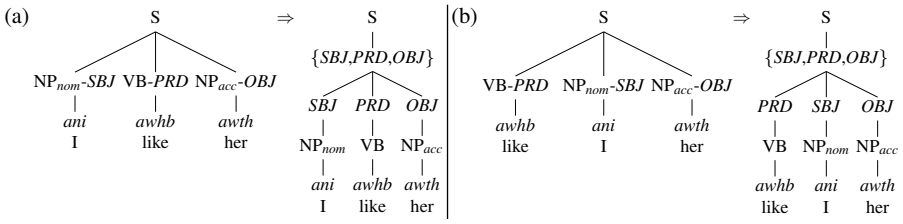

Fig. 24 Generating canonical and non-canonical configurations using the RR model: S level CFG productions at the LHS of (a) and (b) are different. The RR-CFG representations at the RHS of (a) and (b) share the projection and realization of GRs and differ only in their configuration parameters.

$$
\begin{array}{ll}
\mathbf{P_{RR}}(r) = & \\
\textit{Projection} & \mathbf{P_{projection}}(\{gr_i\}_{i=1}^{n}|C_P) \times \\
\textit{Configuration} & \mathbf{P_{configuration}}(\langle gr_0 : gr_1, g_1, \ldots\rangle|\{gr_i\}_{i=1}^{n}, C_P) \times \\
\textit{Realization} & \prod_{i=1}^{n} \mathbf{P_{realization}}(C_i|gr_i, C_P) \times \\
 & \mathbf{P_{adjunction}}(\langle C_{0_1}, \ldots, C_{0_{m_0}}\rangle|gr_0 : gr_1, C_P) \times \\
 & \prod_{i=1}^{n} \mathbf{P_{adjunction}}(\langle C_{i_1}, \ldots, C_{i_{m_i}}\rangle|gr_i : gr_{i+1}, C_P)
\end{array}
$$

The first multiplication implements an independence assumption between grammatical relations and configurational positions. The second multiplication implements an independence assumption between grammatical positions and morphological markings of the nodes at that position. these assumptions are appropriate for languages with flexible word order and rich morphology. The later multiplications implement independence between the generation of complements and the generation of adjuncts. This is a distinction that is relevant to all of the world languages (discussion omitted).

One of the main advantages of the RR modeling method is that it makes explicit the commonalities and differences we expect to find across syntactic structures. For instance, Figure 24 shows the RR generation process for the two clauses we discussed before. In their simple context free representation, there is no parameter sharing between the two constructions. In the RR representation, the trees share the *projection* and *realization* parameters, and only differ in the *configuration* parameter, capturing their alternative word ordering patterns. This factoring of the model helps to generalize morphosyntactic patterns (agreement, case marking, etc.) by directly observing them and estimating their probability from corpus statistics.

Training and decoding proceeds confines with the transfrom-detramsform method outlined in algorithm 3. Training starts off with a set of phrase structure trees in which every node is specifies the syntactic phrase-label, grammatical function, and morphological features. The treebank then goes through a transformation process including (i) separating relations from their realization, (ii) separating position of every relation from its morphological marking. Training can be done in a standard way, using maximum likelihood estimates. Since the resulting grammar assumes independence between the rules, we can use a simple chart parser (over sequences and/or over lattices) to parse unseen sentences using the resulting Relational-Realizational treebank grammar.

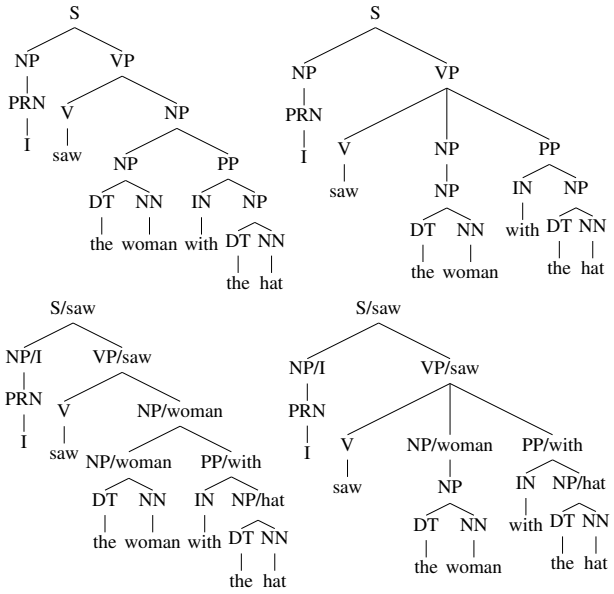

Fig. 25 Unlexicalized and lexicalized syntactic analyses of "I saw he woman with the hat".

Lexicalization and Factored Modeling

Lexical information provide information that is often orthogonal to syntactic information. Consider for instance the alternative English trees for the sentence "I saw the woman with hat." in Figure 25. The analyses on the left and right are syntactically equally plausible, but it is the strong affinity of the lexical items "woman with hat" over and above "seeing with hat" that makes us prefer the analysis on the left. Therefore, it is customary to *lexicalize* the trees. That is, for each node we add the information concerning the lexical head of the dominated constituent, as demonstrated in the bottom of Figure 25. As may be seen, the tri-gram "woman with hat" may be explicitly read off from the left hand tree, while "saw with hat" is the one that we read off from the right hand tree. If we can statistically capture these lexical affinities, we can effectively choose between competing structural alternatives.

The statistical distribution learned by a refined PCFG-based model in this case may be hard, or impossible to learn, based on a limited amount of treebank data. Klein and Manning [55] suggest instead a factor model of syntactic and lexical information, this providing a practical way to model the predicate-argument structure of a sentence as orthogonal to its tree structure. They present an A* search decoding algorithm over hyper-graphs in an agenda-based parser where admissible heuristics are used to estimate of the probabilities of parse completions that constrain the search for the best parse. They exploit admissible heuristics to introduce additional information into the derivation to skew the search towards high affinity head-dependent pairs projected by the structure.

Formally, the factored *constituency/dependency* model assumes an agenda-based parser that guides the search through the space of structures $y \in \mathscr{Y}$ looking for the most likely lexicalized tree $L = \langle y, d \rangle$ with $d \in \mathscr{D}$ a dependency structure projected by the phrase structure $y \in \mathscr{Y}$. Now, if Z $\rightarrow$ X Y is a CF rule and $h = head(Z) = head(X)$ and $h' = head(Y)$, the their solution is defined as follows.[13]

$$L^* = \arg\max_{L=(\tau,d)} p_{factored}(\tau, d|s) = \prod_i p_{rule}(XY|Z) \times p_{dep}(h'|h)$$

The word-level dependency information in the model is formally incorporated through the probability distribution $p(h'|h)$, and it is completely independent of structural information coming from the PCFG productions $p(XY|Z)$.

This model underlies the Stanford Parser[14] which has been applied successfully to parsing (at least) English, Chinese, German and Arabic. All that the application of the model requires is a specification of language specific head selection rules (and possibly a language-specific unknown words treatment). At the same time, it is not clear that the model can be effectively used to model intricate morphological-syntactic interactions, because morphological information captured in h, h' in this model is orthogonal to syntactic configurations. Factored modeling thus goes hand in hand with a factored lexicon, as we describe in §2.4.

Constrained Parsing

From an empirical point of view, it is unclear that state-splits learned for non-terminals symbols will constrain the distribution of trees sufficiently. A study of a state-split grammar learned off of Hebrew trees, for instance, inspects a state-split grammar there is no observable patterns agreement [35]. In order to enforce linguistically grammatical patterns, it is possible to impose constraints on trees and select only the ones that obey certain rules or conventions.

- **Parse Selection**

  Instead of providing a single output structure, most parsers can provide as set of n most probable trees. It is then possible to devise hand-coded linguistic rules as constraints and pick out the first tree that does not violate any constraint. This view of parsing is a computational emulation of the Optimality theoretic framework of Prince and Smolensky [86], in which the grammar is viewed as composed of a GEN component that defines possible structure and a set of violable constraint. According to them, grammatically amounts to the tree that violates the least amount of constraints.

[13] The probability model defined by the formula is deficient [57, footnote 7].

[14] http://nlp.stanford.edu/software/lex-parser.shtml

- **Parse Reranking**

  Re-ranking may be viewed as a refinement of the parse selection idea. In reranking frameworks, rather than selecting the first tree that obeys a set of hard constraints, we encode linguistic information as soft constraints. The parses in the n-best list are re-ranked according to the the weights of features defined over the structure, where the weights are trained using a discriminative framework (see §2.3). The best tree according to the re-ranked probability model is selected as the best parse [21].

- **Chart Pruning**

  The above approaches are applied to n-best lists, but it may be the case that a gold parse does not make it to the n-best list in the first place. A different way to incorporate constraints is by way of pruning the CFG production in a chart. In chart pruning, one observes every CFG production in the chart and prunes states that violate linguistic constraints (e.g., morphological agreement). This view of parsing is akin to theoretical frameworks such as LFG or HPSG that view the construction of the tree as a constraint satisfaction problem [87]. The problem with such pruning is that the probability model is leaking – that is, probability mass is lost due to failed derivations. Therefore, when using these methods we run the risk of tilting the probability ranking in unpredicted directions.

Discriminative Approaches

Encoding non-local features in a History-Based model requires learning fine-grained states that encode very subtle state variation, which is challenging to statistically learn from a finite amount of data. A different statistical approach to defining probability distributions over structures which is gaining increasing popularity is based on *Maximum Entropy (MaxEnt)* principles [4].

MaxEnt models may be defined over arbitrary graphical representations, and it need not assume any independence between different sub-events in the representation of the tree. All information in MaxEnt models in which all information is expressed through *feature functions* that count the number of occurrences of a certain pattern in the representation. Anything that is definable in terms of the representation format can serve as a feature that feeds into such functions.

The parameters of the model are weights that reflect the importance or the usefulness of the respective features. The feature-function values are multiplied by their estimated weights to yield the score of the structure defined by means of those features. The structure that maximizes $p(\tau)$ is the selected parse, where:

$$p(y) = \frac{1}{Z_\lambda} e^{\sum_i \lambda_i f_i(y)} \tag{33}$$

$$Z_\lambda = \sum_{y \in \mathscr{Y}} e^{\sum_i \lambda_i f_i(y)} \tag{34}$$

The main challenge in using such models is estimating their parameters from data. There is no practical way of summing over all trees to calculate the normalization constant Z_λ, so estimation procedures that maximize the joint likelihood of the data and structures (MLE) are generally intractable. Johnson [51] suggests to use *maximum conditional-likelihood estimation (MCLE)* instead, where one maximizes the conditional likelihood of the structure given its yield and the normalizing factor Z_λ is replaced with the following, more manageable $Z_\lambda(y)$.

$$Z_\lambda(y) = \sum_{y' \in \mathscr{Y}_x} e^{\sum_i \lambda_i f_i(y)}$$

This estimation procedure is consistent for the conditional distribution $p(y|x)$ but not the joint distribution $p(y,x)$ we have considered so far. This means that the parser is optimized to *discriminate* between different candidates for a single sentence, and hence the name, *discriminative* approaches. There are multiple ways to incorporate a discriminative method into the parsing model and we briefly review three of them.

- **Discriminative Reranking** [28, 21]. In this kind of model, a probabilistic generative component generates a list of N-best candidates (or represents all possible candidates in a parse forest, [48]) and then re-ranks candidates using a discriminative procedure. Features for discrimination are selected based on pre-defined feature-schemata and an automatic procedure selects the ones that show the best gains. These feature selection procedures are computationally heavy since they require re-parsing the corpus in every training iteration.
- **Discriminative Estimation** It is possible to limit the application of the discriminative method to the estimation of individual parameters, while still using a model for joint inference [88, 50]. The conditional estimation allows to incorporate arbitrary features, but since the parameters are employed in a simple generative process, the feature combination has to remain local. Conditional estimation procedures allow for potentially incorporating more information into individual parameters when training on a fixed amount of data.
- **Discriminative Parsing** Finkel et al. [33] propose an end-to-end CFG parsing system based on Conditional Random Fields (CRF-CFG) in which the estimated probabilities are normalized globally for undirected representations of complete trees. They use an estimation procedure that maximizes the conditional likelihood instead of the joint likelihood, and enrich their parameters with non-local features. The features they use are selected from feature-schemata and the best features are detected using a small development set. Their parser has been applied only to English, and using an efficient decoding procedure it obtains state-of-the-art parsing results on the WSJ, on a par with [48, 11].

2.4 The Lexical Model

Combatting Lexical Sparsity

A parser has to assign morphosyntactic categories to the word-forms in the input. These morphosyntactic categories reflect the part-of-speech tag of the word and a set morphological feature values. Morphosyntactic categories provide the interface between the structure of words (morphology) and the structure of sentences (syntax). Whichever syntactic probabilistic model we decide to use for parsing (PCFG, PCFG-LA, RR-PCFG, Factored-PCFG, etc.) we need to estimate the lexical insertion probabilities $p(w|t)$, that is, assign probability distribution to word forms.

The probability of lexical insertion rules may be estimated using maximum likelihood estimates, just like the syntactic rules, by counting the relative frequency of occurrences.

$$\hat{p}_{tb}(w|t) = \frac{\#(w,t)}{\sum_w \#(w,t)} = \frac{\#(w,t)}{\#t}$$

In Semitic languages, it is hard to inspect all possible morphosyntactic assignments in advance, because of the high variation in the morphological form of words. This is further complicated by the fact that the amount of resources available for parsing such languages is not as extensive as it is for English. Therefore we should also estimate a probability distribution for Out-Of-Vocabulary (OOV) words, and possibly also smooth the probability distribution over rarely occurring ones. It is then customary to fix a rare threashold value β and estimate for the following probability distributions separately.

$$p(w|t) = \begin{array}{ll} \hat{p}_{tb}(w|t) & \#w > \beta \\ \hat{p}_{rare}(w|t) & 0 < \#w \leq \beta \\ \hat{p}_{oov}(w|t) & \#w = 0 \end{array}$$

The simplest way to assign probabilities to rare and OOV words is by assuming a single distribution of tags for unknown words. That is, one simply assumes an abstract UNKNOWN word and estimates the probability of tagging an UNKNOWN word with a certain tag using the distribution of tags on rare words in the data. Every word which is under the β threshold is considered a member of a set RARE. We can calculate the OOV distribution as the probability distribution of RARE words.

$$p_{oov}(w|t) = p_{rare}(w|t) = \frac{\sum_{w \in \text{RARE}} \#(w,t)}{\sum_w \#(w,t)}$$

This estimate embodies a very crude assumption, that the tag assignment to unknown words is not related to its surface form. Obviously, for morphologically rich languages this is not the case, because the form of the word has clear implications concerning its syntactic role in context. Therefore, a refined way to estimate the rare and OOV probability distributions using external resources.

Using an External Lexicon

An external lexicon provides the list of word forms in the language along with their morphological analyses. A morphological analyzer can be thought of as a function mapping word-forms to a set of morphosyntactic signatures as licensed by the lexicon. The information provided by an external lexicon can be used for improving the statistical estimates of lexical insertion probabilities for unseen or rare words.

Signatures

The basic estimates the for the lexical insertion probability given a tag $p(w|t)$ are estimated according to observation in the training set. Additionally, we can use the lexicon to analyze input words and to include parameter estimates for unknown word signatures $p(s|t)$.

A study of parsing Arabic and French by Green et al. [42], for example, estimates the probability distribution of $p(t|w)$ and $p(t|s)$ directly from the training data using relative frequency estimates, and then use Bayes law to get the estimates of $p(w|t)$ (where p_{smooth} interpolates the $p(t|w)$ and $p(s|t)$ estimates).

$$\begin{aligned}
\hat{p}_{tb}(w|t) &= \tfrac{p(t|w)\times p(w)}{p(t)} && \#w > \beta \\
\hat{p}_{rare}(w|t) &= \tfrac{p_{smooth}(t|w)\times p(w)}{p(t)} && 0 < \#w \leq \beta \\
\hat{p}_{oov}(w|t) &= \tfrac{p(t|s)\times p(s)}{p(t)} && \#w = 0
\end{aligned}$$

Factored Lexicon

One way to combat sparsity is to parse a factored representation of the terminals. Each complex terminal symbol may be factored into a word form, the lemma, grammatical features such as gender, number, and person, a morphological template, and so on. In the factored lexicon, each token has an associated morphological analysis m, which is a string describing various grammatical features. Green et al. [42] generate the word and morphological tag using a product of independent factors, conditioned on the part of speech tag.

$$p(w;m|t) = p(w|t)p(m|t)$$

The probability $p(m|t)$ is estimated using the procedure of estimating $p(w|t)$ specified above. Since the number of m,t tuples in the training data is a lot smaller than the number of w,t tuples, many of the probability distributions $p(m|t)$ may be estimated directly directly from treebank observation.

Using External Unannotated Data

Given a morphological analyzer for a language, we would like to use the information concerning the morphosyntactic signatures in order to obtain robust probabilistic estimates for rare and unknown words. There are two main challenges that stand in the way of using a morphological analyzer in this way. One challenge is that the morphological analyzer is symbolic, that means that it can provide the list of analyses, but it does not provide the probabilities $p(analysis|word)$ for the different analyses, so we need to estimate them from annotated or unannotated data. The other challenge is a more subtle one. It turns out that in many cases the morphological analyzer and the syntactic treebank do not respect the same annotation scheme. In some cases, the morphological analyzer categories are more fine-grained, in other cases the treebank part of speech categories are more fine grained. In any event there is no simple mapping between the two set of categories. In this cases, the mapping itself may have to be learned directly from the data.

Layering

The study of Goldberg et al. [40] presents a method for enhancing unlexicalized parsing performance using a wide-coverage lexicon, fuzzy tag-set mapping, and EM-HMM-based lexical probabilities which are obtained using extra amounts of unannotated data. In their setup, the treebank annotation scheme is different from the analyses provided by the morphological analyzer. They show that simply retagging the corpus gives inferior results, since syntactically relevant distinctions are missed. Using treebank tags only precludes the use of the morphological analyzer, leading to higher OOV rate and sparse estimates for the lexical probabilities.

Instead, they proposed a generative approach in which the emission probability is decomposed. First, the tree-bank tag generates a lexicon category, and then the lexicon category generates the word.

$$p(w|t_{tb}) = p(t_{lex}|t_{tb})p(w|t_{lex})$$

The transfer probabilities $p(t_{lex}|t_{tb})$ are estimated using maximum likelihood estimates on a layered representation of the treebank, in which every production is extended with the lexical category assigned in the lexicon. The lexicon categories assignment was performed semi-automatically by human experts. The lexical probabilities $p(w|t_{lex})$ are estimated using an HMM tagger in a semi-supervised manner, using a large set of unannotated data [2]. Goldberg et al. [40] show that using this generative process of assigning lexical categories and probabilities so estimated they significantly improve the performance of a manually refined treebank PCFG.

Marginalizing

Manually tagging the corpus with two layers of part of speech categories may be labor intensive, and in any event, human annotators themselves do not always reach an agreement on the correct mapping. Goldberg [35] in his thesis present an alternative method for assigning lexical probabilities, by marginalizing over all possible lexical tag assignements. Goldberg first assumes an inversion using Bayes law.

$$P(w|t_{tb}) = \frac{P(t_{tb}|w)P(w)}{P(t_{tb})}$$

Then, he provides different ways to calculate $P(t_{tb}|w)$.

$$\begin{array}{ll} \hat{p}_{tb}(t_{tb}|w) = \frac{\#w,t_{tb}}{\#t_{tb}} & \#w > \beta \\ \hat{p}_{oov}(t_{tb}|w) = \sum_{t_{lex}} p(t_{lex}|w)p(t_{tb}|t_{lex}) & \#w = 0 \\ \hat{p}_{rare}(t_{tb}|w) = \frac{\#w \times p_{tb}(t_{tb}|w) + p_{oov}(t_{tb}|w)}{\#w+1} & 0 < \#w \leq \beta \end{array}$$

For frequent words, he uses counts on the treebank. For unseen words, he estimates $p(t_{lex}|w)$ using pseudocounts taken from an HMM tagger applied to unannotated data [1]. Because the HMM tagger uses an external lexicon, the lexicon tags are different from the treebank tags. The transfer probabilities $p(t_{tb}|t_{lex})$ are treated as hidden variables and are marginalized out.

Clustering

An additional way to utilize extra amounts of unannotated data is by means of unsupervised clustering. First, we use a large unannotated corpus to learn word clusters (e.g., Brown clusters [13]). Then, these clusters can be used as features in a statistical parser. Koo et al. [58] used s a cluster-based feature mapping for a discriminative learner. They replaced words with full description of the cluster and pos-tags with shorter description of the hierarchical cluster. A possible way to incorporate these feature into a constituency-based generative parser is, for instance, to change the pos-rags and word-form representation into clusters. Candito and others [17, 18] tested the method of replacing inflected forms by clusters of forms in a generative probabilistic parser, and obtained good results. Levinger et al. [61] take a different approach to clustering, where word clusters are based on linguistic similarity. They collect similar-word (SW) sets using hand-coded rules and used the frequencies of words in SW sets in order to differentiate the different analyses of an ambiguous word. They show that their statistical estimates improves performance on a number of tasks. Their method was also used to initiate the semi-supervised tagging by [1].

3 Empirical Results

3.1 Parsing Modern Standard Arabic

The Penn Arabic Treebank project (ATB), a set of annotated corpora for Modern Standard Arabic (MSA), has been developed by the Linguistic Data Consortium (LDC) since 2001.[15] The ATB annotation encodes the rich morphological structure of words and indicates the grammatical relation representation in the form of phrase-structure trees [62]. In addition, the ATB deals with phonemic transliteration of the Arabic script, bidirectionality, and the incorporation of diacritics [63, 66]. The PoS tags in the ATB are complex tags that specify the part-of-speech category, an English gloss, and a list of inflectional features for every space-delimited word. Independent clitics are segmented away and are assigned their own PoS tag. Syntactic trees are built on top of morphologically analyzed and segmented input.

The LDC team made a deliberate effort to rely on the English PTB annotation guidelines in order to speed up the development of the Arabic annotated corpora. The morphological analyzer of Buckwalter [15] was used to propose all possible analyses to surface space-delimited tokens, from which human annotators chose the correct one. Then, the parsing engine of Bikel [5] was used to automatically bootstrap syntactic parse-trees for the morphologically analyzed tokens [62]. This annotation pipeline has been challenged in [64, 65] due to mismatches between the PoS tags assigned by the morphological analyzer and the typical syntactic structure assigned by the parser. These mismatches are found to originate in the inherent complexity of the Semitic grammar, and the ATB guidelines have been revised to reflect such mismatches and treat them in the annotation scheme.

The LDC published different versions of the ATB. ATBp1 contains 166K words from the Agence France Press corpus, segmented, PoS tagged and syntactically analyzed. ATBp2 contains 144K words from *Al-Hayat* annotated similarly but with added case and mood endings indicated by diacritics. ATBp3 contains 350K words from the newswire text *An-Nahar*, in which the text is morphologically segmented, vocalized, tagged and syntactically analyzed. Because of the discrepancy between treebank words and space-delimited tokens, any morphologically analyzed and vocalized files contain mappings to the original surface forms.

The first constituency-based parser which was trained to parse Arabic using the ATB is the Head-Driven lexicalized parsing engine of Bikel [5], adapted in [7] to Arabic parsing. The rich morphological structure of Arabic words induces a large set of complex tags which the parser was not equipped to handle (Bikel [7, p. 79–80]), so the adaptation involved a reduction of the Arabic rich tag set to a set with comparable size to that of English. Bikel [7] only experimented with parsing off of gold input, assuming that morphological segmentation and tagging are known in advance. The results of applying Bikel's parsing Engine to parsing Arabic (75.68 F-Score) were lower than the results obtained previously for parsing English on a data set of a comparable size (87.54 F-Score) [63, 60].

[15] The project page is available at http://www.ircs.upenn.edu/arabic/

Parsing using Bikel's engine was mostly targeted at finding enhancements of the initial category set in the ATB. In a series of studies the ATB development team suggested fixes for punctuation, deverbal material, finer-grained distinctions between different types of nouns and demonstratives and other splits, based on these experiments [60, 63, 64, 66, 65]. Once the PoS tags have been enhanced, the ATB team changed the trees dominating certain nominals to reflect their verbal readings. The parsing results for all of these models leveled at about 79 F-Score for parsing gold segmented and tagged input [60].

Green and Manning [43] study in greater detail the factors affecting Arabic parsing performance using the ATB. They firstly compare the ATB to the Chinese and English treebanks, and report that they are comparable in gross statistical terms, while the ATB demonstrates a higher level of annotation inconsistencies. They then compare the performance of three different parsing models on the standard split of the ATB [23]. Specifically, they compare Bikel's parsing engine [6] and Petrov's PCFG-LA parser [81] with a human-interpretable state-split PCFG developed to address particular sources of ambiguity that they find to be pertinent in the ATB. They show that this PCFG, in conjunction with the Stanford parser (a factored model implementation based on [54]), achieves $F_1$80.67 on gold segmented and tagged input, outperforming the Bikel parser ($F_1$77.99) and outperformed by the PCFG-LA parser ($F_1$84.29) in the same settings. This performance trend persists when the corpus is pre-tagged or self-tagged by the parser (albeit obtaining lower scores). The trend was further confirmed using the Leaf Ancestor metric [91]. They finally compare the performance of the Stanford parser on the ATB for raw words, using a pipeline and a joint scenario. They show that a pipeline containing the MADA tagger followed by the Stanford parser ($F_1$79.17) outperforms joint lattice-based parsing using the stanford parser ($F_1$76.01), an overall loss of 2pt-5pt in F-Scores relative to the artificial gold-input parsing scenario.

Green and Manning [43] provide strong baselines for Arabic parsing and establish an empirical advantage of the PCFG-LA parser [81]. Attia et al. [77] explore the use of simple or complex morphological cues for the handling of rare and unknown words in English, Arabic and French, in order to further improve the performance of the PCFG-LA model. They use their own PCFG-LA implementation and contrast two techniques of handling unknown words, a language-independent and a language-specific one. The language-independent technique simply learns a distribution for unknown words using rare words. In the language-specific case they map every word to its morphosyntactic signature, and divide the UNKNOWN probability mass across morphosyntactic signatures. Their experiments show that language specific cues were more helpful for Arabic than they were for French and English, ultimately increasing parser performance on the ATB dev set from $F_1$79.53 to $F_1$81.42, for gold segmented input. A study by Dehadri et al. [31] explored strategies for finding best features set and PoS tag split using the same parser. They experimented with a non-iterative best-first search algorithm, an iterative, greedy best-first search algorithm, an iterative, best-first with backtracking search algorithm and simulated annealing. They obtain $F_1$85.03 for vocalized ATB no-tag parsing and $F_1$83.34 for unvocalized ATB no-tag parsing, both of which assuming gold segmented input.

Arabic constituency parsing has also been explored in the context of parsing Arabic Dialects [23]. Arabic dialects are mainly spoken, so they do not have annotated corpora of their own. Chiang et al. [23] try to leverage the lack of data by exploiting the ATB and using explicit knowledge about the relation between MSA and its Levantine Arabic (LA) dialect. They test three approaches of interjecting such knowledge to the parsing architecture: sentence transduction, in which the LA sentence is turned into an MSA sentence and then parsed with an MSA parser, treebank transduction, in which the MSA treebank is turned into an LA treebank and used as training material for an LA parser; and grammar transduction, in which an MSA grammar is turned into an LA grammar which is then used for parsing LA. Grammar transduction obtained the best result, around $F_1$67 on gold-tagged input, more than 17% error reduction over their baseline.

Advances in dependency parsing [59] along with long-lasting claims for the adequacy of dependency grammars for analyzing free word-order phenomena [75, 97] have jointly led to increased interest in Arabic dependency parsing. Initial Arabic dependency parsing results were published in connection to data sets that were made available in the shared tasks on multilingual dependency parsing [14, 78].[16] The experimental setup in these experiments is artificial in the sense that it assumes that the gold segmentation, part-of-speech tags and morphological features are known prior to parsing. This is an unrealistic assumption, but even in artificial settings the task allowed the community to gain insights concerning the adequacy of general-purpose parsers for parsing Semitic phenomena. In an overall analysis of the shared-task based on the cross-linguistic results, Nivre [78] makes the typological observation that Arabic is one of the most challenging languages to parse, regardless of the parsing method used. The claim that completely language-independent parser may work sufficiently well on Semitic data has now been disconfirmed, and various specific studies on Arabic dependency parsing have emerged.

Marton et al. [69, 70, 71] focus on transition-based parsing and present a thorough study of the contribution of the lexical and inflectional features to Arabic dependency parsing. They experiment with two parsers, MaltParser [79] and EasyFirst [34], in three different experimental settings: gold scenarios (where morphological analysis is known in advance), and predicted scenarios (in which morphological analyses are automatically predicted). The data set they used is a converted version of the ATB into dependencies, using a narrowed down set of dependency labels.[17]. They report that more informative (fine-grained) tag sets are useful in gold scenarios, but may be detrimental in predicted scenarios. They identify a set of features (definiteness, person, number, gender, undiacritized lemma) that improve parsing performance, even in predicted scenarios. They finally show that some of the deficiency of parsing in predicted scenarios may be mitigated by training the parsers on both gold and predicted tags. Their findings are robust across parsers, with an advantage for EasyFirst (82.6 vs. 81.9 labeled attachment scores in the best settings).[18]

[16] The CoNLL-X shared task page for is available at http://ilk.uvt.nl/conll/

[17] The project homepage is available at http://www.ccls.columbia.edu/project/catib.

[18] An additional resource for Arabic dependency parsing is the Prague dependency treebank http://ufal.mff.cuni.cz/padt/PADT_1.0/docs/index.html which we omit from the present discussion.

3.2 Parsing Modern Hebrew

The main resource for the development of statistical parsers for Hebrew is the Modern Hebrew Treebank [95], which was developed at the MILA (The knowledge center for processing Hebrew).[19] The treebank contains 6220 unvocalized sentences from the daily newspaper *Ha'aretz*. The sentences were morphologically segmented and syntactically annotated in a semi-automatic process. Each node in the syntactic parse tree contains different sorts of information: a phrase-structure label (S, NP, VP, etc.) grammatical function labels (from a small set containing SBJ, OBJ, COM and CNJ) and a set of morphological features decorating part-of-speech tags. In the second version of the Hebrew treebank, morphological features have been percolated to higher level syntactic categories, and (partial) head information has been added to phrase-level nodes, based on hand coded syntactic rules [45].

Tsarfaty [99] reported the use of the Hebrew treebank for creating the first Hebrew NLP pipeline, using a combination of morphological analyzer, a part-of-speech tagger, treebank-induced PCFG and a chart parser. The best reported result on parsing raw input word in this study was $F_1$61 with about 91.5 segmentation accuracy. Since the results involved unmatched yields, the scores were reported using character-based ParseEval. Cohen and Smith [26] followed up with an implementation of a joint inference for morphological and syntactic disambiguation using a product of experts. Their syntactic model is a treebank PCFG with first-order vertical Markovization. For the morphological model they contrast a unigram-based model and a discriminative model based on Conditional Random Fields (CRF). For their best model they report 64.4 parsing accuracy and 90.9 segmentation accuracy for the joint model on raw input words. The numbers are not comparable to other studies, however, because Cohen and Smith used their own evaluation metrics based on string edit distance correction to the tree.

Goldberg and Tsarfaty [39] present a fully generative solution of the joint morphological and syntactic disambiguation task based on lattice parsing. Their best model included PCFG with 1st-order vertical Markovization, manually selected category splits and unknown words handling using a Hebrew spell-checker. The baseline for Hebrew parsing was now pushed further to $F_1$66.60 on parsing raw input (68.79 on the metrics of Cohen and Smith) with $F_1$94.7 segmentation accuracy. These results made it clear that for a Hebrew treebank so small, additional resources for unknown words treatment may have crucial effects on parsing accuracy. Goldberg et al. [40] experimented with a wide-coverage lexicon, fuzzy tag-set mapping and EM-HMM-based lexical probabilities acquired using a large unannotated corpus, and reported 73.40/73.99 precision/recall on parsing raw words (compared with 76.81/76.49 precision/recall for parsing off of gold segmentation). The components of these different solutions, including a lattice-based decoder, a wide-coverage lexicon and HMM probabilities were incorporated by Goldberg [35] into the PCFG-LA parser of Petrov [81], along with a rule-based filter to the output, obtaining $F_1$76.95 on parsing raw words (and F_1 85.7 for parsing gold segments).

[19] The MILA center homepage is at http://www.mila.cs.technion.ac.il/.

In an orthogonal line of work, Tsarfaty and Sima'an [103] explored the contribution of different parameterization decisions to probabilistic grammars induced from the Hebrew treebank. They experimented with horizontal Markovization, vertical Markovization and morphological state-splits applied to a PCFG-based parsing model. They have shown that when each dimension of parameterization is considered in isolation, the *depth* annotation contributed more to parsing accuracy than parent annotation, which in turn contributed more than horizontal Markovization. They further show that the contribution of the different dimensions is cumulative, and that using 1st-order vertical Markovization, 1st-order horizontal Markovization and one level of morphological depth, parsing results improve from $F_1$69.24 to $F_1$74.41 on gold segmented input.

The emerging insight is that parameterization decisions which work well for English parsing may miss important generalizations that are relevant for parsing other languages, e.g., languages with rich morphology. Tsarfaty et al. [104, 105, 106] followed up on this insight with the development of the Relational-Realizational (RR) model and its application to parsing Hebrew. In the application to Hebrew data, described in detail in Tsarfaty [100], the RR framework is applied to explicitly modeling and directly estimating morphosyntactic patterns such as differential case marking, agreement and clitics. The model obtained $F_1$76.6 score on parsing gold segmented input, and $F_1$84.4 when parsing gold segmented and tagged input.

A recent study by Tsarfaty et al. [101] compares a combination of the RR model with lattice-based PCFG-LA parsing [36] with bare-bone phrase structure (PS) lattice-based PCFG-LA parsing. The different models are evaluated using TedEval, distance-based scores which are tailored for evaluating morphological syntactic disambiguation jointly [101]. In all parsing scenarios (gold, predicted tags, raw), the bare-bone PS model outperform the trees extracted from the RR model: TedEval yields 93.39, 86.26 80.67 on trees given by the PS model and 92.45, 85.83,79.46 on trees extracted from the RR model. An advantage of the RR trees over is that they provide a direct representation of predicate-argument structure in their relational networks. Evaluation on dependency labels using TedEval shows 88.01, 82.64 and 76.10 accuracy on RR trees, whereas bare-bone PS trees provide no information concerning dependency labels.

An additional way to obtain a direct representation of the predicate argument structure of sentences by training a statistical parser directly on dependency trees. Goldberg and Elhadad [37] converted the Hebrew treebank into dependency treebank and compared the results of several general-purpose dependency parsers on the treebank in gold vs. predicted settings.[20] They tested the graph-based MST parser [74] and the transition-based MaltParser [80]. In all scenarios, the 2nd-order MST model outperformed the 1st-order MST, and both outperformed MaltParser. The best result obtained for the Hebrew treebank, a second order MST with limited lexical features, was 84.77 unlabeled attachment scores on the gold input scenario and 76.10 on unlabeled attachment scores on automatically predicted segmentation and tagging.

[20] The treebank data is available at http://www.cs.bgu.ac.il/ yoavg/data/hebdeptb/.

Goldberg and Elhadad [34] moved on to propose EasyFirst, a non-directional transition-based dependency parser that is aimed at making easier attachments earlier. The feature model of the EasyFirst parser can make reference to partial parses, and thus allow incorporating features that encode rich morphological interactions. The ability of the EasyFirst parser to prefer different ordering of attachments, along with the ability to incorporate complex features makes it particularly suited for Semitic phenomena. Goldberg and Elhadad [38] accordingly report 84.2 attachment scores on the gold input and 76.2 on automatically predicted segmentation and tagging. It is still an open empirical question how well the EasyFirst performs on labeled dependencies.

4 Conclusion and Future Work

This chapter surveyed significant morphosyntactic phenomena in Semitic languages and presented the challenges they pose to general-purpose statistical parsers. We have seen that the structure of Semitic languages complicates the NLP pipeline, by providing morphologically ambiguous signal as input to the parser. We have further shown how parameterization decisions in generative parsing models may affect the performance of a parser relative to the properties of the language, and proposed various methods for adding linguistic constraints and for dealing with sparse data.

We reported parsing results only on Modern Hebrew and Modern Standard Arabic. A pre-condition for applying any statistical parsing model to the other Semitic languages is the availability of morphologically and syntactically annotated corpora. As of yet there are no broad-coverage treebanks for these languages. An important direction for further research is therefore the development of such annotated corpora for other Semitic languages. Given the similarity of Semitic languages to one another, it may well be possible to speed the work on such treebanks using transfer learning. For those languages with annotated corpora, the time is ripe to explore more sophisticated modeling methods, possibly incorporating discriminative estimation, semantic resources, or external constraints.

We have seen that different models addressed different sorts of challenges. It would be interesting to try to combine the strengths of these orthogonal strategies. For instance, while the constituency parser of Goldberg [35] shows gains from improving the lexical model using an external lexicon and unannotated data, the modeling method of Tsarfaty [100] is targeted at improving the syntactic model via parameterization choices. Their combination may yield a more accurate parser.

Beyond that, a careful treatment of evaluation across languages, parsers and scenarios is critical. We have seen in the previous section that while many of the studies deal with the same challenges, they report numbers on different metrics and on different scales. Simple numerical comparison across models and parsing scenarios are thus uninformative. Only upon developing better methods to evaluate the quality of parses across frameworks, possibly within downstream applications, we will be able to identify which models make better predictions of complex Semitic structures.

Acknowledgements

The author wishes to thank Joakim Nivre, Imed Zitouni, Ro'i Gibori and two anonymous reviewers for useful comments on earlier drafts. The author further thanks Yoav Goldberg, Spence Green and Shay Cohen for discussion and relevant resources, and Reshef Shilon for discussion of Arabic data. The writing of this chapter was partially funded by the Swedish research council, for which we are grateful.

References

1. Meni Adler. *Hebrew Morphological Disambiguation: An Unsupervised Stochastic Word-based Approach.* PhD thesis, Ben-Gurion University, 2007.
2. Meni Adler and Michael Elhadad. An unsupervised morpheme-based hmm for hebrew morphological disambiguation. In *Proceedings of COLING-ACL*, 2006.
3. Stephen R. Anderson. *A-Morphous Morphology*. Cambridge University Press, 1992.
4. Adam L. berger, Stephen A. Della Pietra, and Vincent A. Della Pietra. A Maximum-Entropy approach to natural language processing. *Computational Linguistics*, 1996.
5. Daniel M. Bikel. Design of multi-lingual parallel processing statistical parsing engine. In *Proceedings of HLT*, San Diego, CA, 2002.
6. Daniel M. Bikel. Intricacies of collins' parsing model. *Computational Linguistics*, 4(30), 2004.
7. Daniel M. Bikel. *On the Parameter Space of Generative Lexicalized Statistical Parsing Models*. PhD thesis, University of Pennsylvanya, 2004.
8. Ezra Black, John Lafferty, and Salim Roukos. Development and evaluation of a broad-coverage probabilistic grammar of english language computer manuals. In *Proceedings of ACL*, 1992.
9. Leonard Bloomfield. *Language*. Holt, Rinehart and Winston Inc., 1933.
10. Rens Bod. *Enriching Linguistics With Statistics*. PhD thesis, University of Amsterdam, 1995.
11. Rens Bod. An efficient implementation of a new dop model. In *Proceedings of EACL*, 2003.
12. Joan Bresnan. *Lexical-Functional Syntax*. Blackwell, 2000.
13. Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. Class-based n-gram models of natural language. *Comput. Linguist.*, 18(4):467–479, December 1992.
14. Sabine Buchholz and Erwin Marsi. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of CoNLL-X*, pages 149–164, 2006.
15. Tim Buckwalter. Arabic morphological analyzer version 1.0. Linguistic Data Consortium, 2002.
16. Aoife Cahill, Michael Burke, Ruth O'Donovan, Stefan Riezler, Josef van Genabith, and Andy Way. Wide-coverage deep statistical parsing using automatic dependency structure annotation. *Computational Linguistics*, 34(1):81–124, 2008.
17. Marie Candito and Benoit Crabbé. Improving generative statistical parsing with semi-supervised word clustering. In *Proceedings of IWPT*, 2009.
18. Marie Candito and Djame Seddah. Parsing word clusters. In *Proceedings of NAACL/HLT workshop on Statistical Parsing of Morphologically Rich Languages*, 2010.
19. Eugene Charniak. Tree-Bank Grammars. In *AAAI/IAAI, Vol. 2*, pages 1031–1036, 1996.
20. Eugene Charniak. Statistical parsing with a context-free grammar and word statistics. In *AAAI/IAAI*, pages 598–603, 1997.
21. Eugene Charniak and Mark Johnson. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of ACL*, 2005.

22. Eugene Charniak, Mark Johnson, Micha Eisner, David Ellis Joseph Austerweil, Isaac Haxton, Catherine Hill, R. Shrivaths, Jeremy Moore, Michael Pozar, and Theresa Vu. Multilevel coarse-to-fine pcfg parsing. In *Proceedings of HLT-NAACL*, 2006.
23. David Chiang, Mona Diab, Nizar Habash, Owen Rambow, and Safiullah Shareef. Parsing arabic dialects. In *Proceedings of EACL*, 2006.
24. Noam Chomsky. Three models for the description of language. *IRE Transactions on Information Theory*, 2(2):113–123, 1956.
25. Noam Chomsky. *Syntactic Structures*. Mouton: The Hague, 1957.
26. Shay B. Cohen and Noah A. Smith. Joint morphological and syntactic disambiguation. In *Proceedings of EMNLP-CoNLL*, pages 208–217, 2007.
27. Michael Collins. Three generative lexicalized models for statistical parsing. In *Proceedings of ACL*, 1997.
28. Michael Collins. Discriminative reranking for natural language parsing. In *Proceedings of ICML*, 2000.
29. Michael Collins. Head-driven statistical models for natural language parsing. *Computational Linguistics*, 2003.
30. Gabi Danon. Syntactic definiteness in the grammar of modern hebrew. *Linguistics*, 6(39), 2001.
31. Jon Dehdari, Lamia Tounsi, and Josef van Genabith. Morphological features for parsing morphologically-rich languages: A case of arabic. In *Proceedings of the second Workshop on Parsing Morphologically Rich Languages*, 2011.
32. Edit Doron. Agency and voice: The semantics of the semitic templates. *Natural Language Semantics*, 11:1–67, 2003.
33. Jenny Rose Finkel, Alex Kleeman, and Christopher D. Manning. Efficient, feature-based, conditional random field parsing. In *Proceedings of ACL*, 2008.
34. Yoav Goldberg. An efficient algorithm for easy-first non-directional dependency parsing. In *Proceedings of NAACL/HLT*, 2010.
35. Yoav Goldberg. Automatic syntactic analysis of modern hebrew. Technical report, Ben Gurion University, 2011.
36. Yoav Goldberg. Joint morphological segmentation and syntactic parsing. In *Proceedings of ACL*, 2011.
37. Yoav Goldberg and Michael Elhadad. Hebrew dependency parsing: Initial results. In *Proceedings of IWPT*, 2009.
38. Yoav Goldberg and Michael Elhadad. Easy-first dependency parsing of modern hebrew. In *Proceedings of NAACL/HLT workshop on Statistical Parsing of Morphologically Rich Languages*, 2010.
39. Yoav Goldberg and Reut Tsarfaty. A single framework for joint morphological segmentation and syntactic parsing. In *Proceedings of ACL*, 2008.
40. Yoav Goldberg, Reut Tsarfaty, Meni Adler, and Michael Elhadad. Enhancing unlexicalized parsing performance using a wide coverage lexicon, fuzzy tag-set mapping, and em-hmm-based lexical probabilities. In *Proceedings of EACL*, 2009.
41. Joshua Goodman. Probabilistic feature grammars. In *Proceedings of IWPT*, 1997.
42. Spence Green, Marie-Catherine de Marneffe, and Christopher D. Manning. Parsing models for identifying multiword expressions. *Computational Linguistics*, Special Issue on Parsing Morphologically Rich Languages, to appear.
43. Spence Green and Christopher D. Manning. Better arabic parsing: Baselines, evaluations, and analysis. In *Proceedings of COLING*, 2010.
44. Joseph H. Greenberg. Some universals of grammar with particular reference to the order of meaningful elements. In Joseph H. Greenberg, editor, *Universals of Language*, pages 73–113. MIT Press, 1963.
45. Nomie Guthmann, Yuval Krymolowski, Adi Milea, and Yoad Winter. Automatic annotation of morpho-syntactic dependencies in a Modern Hebrew treebank. In Frank Van Eynde, Anette Frank, Koenraad De Smedt, and Gertjan van Noord, editors, *Proceedings of TLT*, 2009.

46. Nizar Habash, Abdelhadi Soudi, and Timothy Buckwalter. On arabic transliteration. *Text, Speech and Language Technology*, 38:15–22, 2007.
47. Julia Hockenmaier and Mark Steedman. Generative models for statistical parsing with combinatory categorial grammar. In *Proceedings of ACL*, 2002.
48. Liang Huang. Forest reranking: Discriminative parsing with non-local features. In *Proceedings of ACL*, 2008.
49. Mark Johnson. PCFG Models of Linguistic Tree Representations. *Computational Linguistics*, 24(4):613–632, 1998.
50. Mark Johnson. Joint and conditional estimation of tagging and parsing models. In *Proceedings of ACL*, 2001.
51. Mark Johnson, Stuart Geman, Stephan Canon, Zhiyi Chi, and Stefan Riezler. Estimators for stochastic "unification-based" grammars. In *Proceedings of ACL*, 1999.
52. Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition Second Edition*. 2009.
53. T. Kasami. An efficient recognition and syntax-analysis algorithm for context-free languages. Technical report, Scientific report AFCRL-65-758, Air Force Cambridge Research Lab, 1965.
54. Dan Klein and Christopher D. Manning. Fast exact inference with a factored model for natural language parsing. In *Advances in Neural Information Processing Systems 15 (NIPS 2002)*, 2002.
55. Dan Klein and Christopher D. Manning. A* parsing: Fast exact viterbi parse selection. In *Proceedings of NAACL*, 2003.
56. Dan Klein and Christopher D. Manning. Accurate Unlexicalized Parsing. In *Proceedings of ACL*, pages 423–430, 2003.
57. Dan Klein and Christopher D. Manning. Factored a* search for models over sequences and trees. In *Proceedings of IJCAI*, 2003.
58. Terry Koo, Xavier Carreras, and Michael Collins. Simple semi-supervised dependency parsing. In *IProceedings of ACL/HLT*, 2008.
59. Sandra Kübler, Ryan McDonald, and Joakim Nivre. *Dependency Parsing*. Number 2 in Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers, 2009.
60. Seth Kulick, Ryan Gabbard, and Mitchell Marcus. Parsing the arabic treebank: Analysis and improvements. In *Proceedings of TLT*, 2006.
61. Moshe Levinger, Uzzi Ornan, and Alon Itai. Learning morpho-lexical probabilisties from an untagged corpus with an application to hebrew. *Computational Linguistics*, 1995.
62. Mohamed Maamouri, Ann Bies, Tim Buckwalter, and Wigdan Mekki. The penn arabic treebank: Building a large-scale annotated arabic corpus. In *Proceedings of NEMLAR International Conference on Arabic Language Resources and Tools*, 2004.
63. Mohamed Maamouri, Ann Bies, and Seth Kulick. Diacritization: A challenge to arabic treebank annotation and parsing. In *Proceeding of the British Computer Society Arabic NLP/MT Conference*, 2006.
64. Mohamed Maamouri, Ann Bies, and Seth Kulick. Enhanced annotation and parsing of the arabic treebank. In *Proceedings of INFOS*, 2008.
65. Mohamed Maamouri, Ann Bies, and Seth Kulick. Enhancing the arabic treebank: a collaborative effort toward new annotation guidelines. In *Proceedings of LREC*, 2008.
66. Mohamed Maamouri, Seth Kulick, and Ann Bies. Diacritic annotation in the arabic treebank and its impact on parser evaluation. In *Proceedings of LREC*, 2008.
67. David M. Magerman. Statistical decision-tree models for parsing. In *Proceedings of ACL*, pages 276–283, 1995.
68. Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, Massachusetts, June 1999.
69. Yuval Marton, Nizar Habash, , and Owen Rambow. Improving arabic dependency parsing with inflectional and lexical morphological features. In *Proceedings of the first workshop on Statistical Parsing of Morphologically Rich Languages (SPMRL) at NA-ACL*, 2010.

70. Yuval Marton, Nizar Habash, and Owen Rambow. Improving arabic dependency parsing with lexical and inflectional surface and functional features. In *Proceedings of ACL*, 2011.
71. Yuval Marton, Nizar Habash, and Owen Rambow. Improving arabic dependency parsing with surface and functional morphology features. *Computational Linguistics Special Issue on Parsing Morphologically Rich Languages*, to appear.
72. Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. Probabilistic cfg with latent annotations. In *Proceedings of ACL*, 2005.
73. David McClosky, Eugene Charniak, and Mark Johnson. Reranking and self-training for parser adaptation. In *Proceedings of ACL*, 2006.
74. Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. Non-projective dependency parsing using spanning tree algorithms. In *HLT '05: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 523–530, Morristown, NJ, USA, 2005. Association for Computational Linguistics.
75. Igor Mel'čuk. *Dependency Syntax: Theory and Practice*. State University of New York Press, 1988.
76. Yusuke Miyao, Ninomiya Takashi, and Tsujii Jun'ichi. Corpus-oriented grammar development for acquiring a head-driven phrase structure grammar from the penn treebank. In *Proceedings of IJCNLP*, 2004.
77. Deirdre Hogan Joseph Le Roux Lamia Tounsi Mohammed Attia, Jennifer Foster and Josef van Genabith. Handling unknown words in statistical latent-variable parsing models for arabic, english and french. In *Workshop on Parsing Morphologically Rich Languages*, 2010.
78. Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 915–932, 2007.
79. Joakim Nivre, Johan Hall, and Jens Nilsson. Maltparser: A data-driven parser-generator for dependency parsing. In *Proceedings of LREC*, pages 2216–2219, 2006.
80. Joakim Nivre, Jens Nilsson, Johan Hall, Atanas Chanev, Gülsen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(1):1–41, 2007.
81. Slav Petrov. *Coarse-to-Fine Natural Language Processing*. PhD thesis, University of Califiornia at Berkeley, 2009.
82. Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of ACL*, 2006.
83. Slav Petrov and Dan Klein. Improved inference for unlexicalized parsing. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 404–411, Rochester, New York, April 2007. Association for Computational Linguistics.
84. Detlef Prescher. A tutorial on the expectation-maximization algorithm including maximum-likelihood estimation and em training of probabilistic context-free grammars. Presented at the 15th European Summer School in Logic, Language, and Information (ESSLLI), 2003.
85. Detlef Prescher. Head-driven pcfgs with latent-head statistics. In *Proceedings of ACL*, 2005.
86. Alan Prince and Paul Smolensky. Optimality theory: Constraint interaction in generative grammar. Technical report, utgers University Center for Cognitive Science and Computer Science Department, University of Colorado at Boulder, 1993.
87. Geoffrey K. Pullum and Barbara C. Scholz. On the distinction between model-theoretic and generative-enumerative syntactic frameworks. In *Proceedings of Logical Aspects of Computational Linguistics (LACL)*, 2001.
88. Adwait Ratnaparkhi. A linear observed time statistical parser based on maximum entropy models. In *Proceedings of EMNLP*, 1997.
89. Brian Roark, Mary Harper, Eugene Charniak, Bonnie Dorr C, Mark Johnson D, Jeremy G. Kahn E, Yang Liu F, Mari Ostendorf E, John Hale H, Anna Krasnyanskaya I, Matthew Lease D, Izhak Shafran J, Matthew Snover C, Robin Stewart K, and Lisa Yung J. Sparseval: Evaluation metrics for parsing speech. In *Proceesings of LREC*, 2006.
90. Ivan A. Sag and Thomas Wasow. *Syntactic Theory: A Formal Introduction*. CSLI Publications, Stanford, 1999.

91. Geoffrey Sampson and Anna Babarczy. A test of the leaf-ancestor metric for parse accuracy. In *Proceedings of "Beyond Parseval" LREC Workshop*, 2002.
92. Stuart M. Shieber. Evidence against the context-freeness of natural language. *Linguistics and Philosophy*, 8:333–343, 1985.
93. Reshef Shilon, Nizar Habash, Alon Lavie, and Shuly Wintner. Machine translation between hebrew and arabic. *Machine Translation*, 26(1-2):177–195, 2012.
94. Ur Shlonsky. *Clause Structure and Word Order in Hebrew and Arabic*. Oxfrod Studies in Comparative Syntax. Oxford University Press, 1997.
95. Khalil Sima'an, Alon Itai, Yoad Winter, Alon Altman, and Noa Nativ. Building a Tree-Bank for Modern Hebrew Text. In *Traitement Automatique des Langues*, 2001.
96. Mark Steedman. *Surface Structures and Interpretation*. Number 30 in Linguistic Inquiry Monograph. The MIT Press, Cambridge, MA., 1996.
97. Lucian Tesnière. *Èlèmenets de Syntaxe Structurale*. Editions Klincksieck, 1959.
98. Reut Tsarfaty. Participants in action: Aspectual meanings and thematic relations interplay in the semantics of semitic morphology. In Henk Zeevat and Balder ten Cate, editors, *Proceedings of the Sixth International Tbilisi Symposium on Language, Logic and Computation*, 2005.
99. Reut Tsarfaty. Integrated morphological and syntactic disambiguation for modern hebrew. In *Proceeding of ACL-SRW*, 2006.
100. Reut Tsarfaty. *Relational-Realizational Parsing*. PhD thesis, University of Amsterdam, 2010.
101. Reut Tsarfaty, Joakim Nivre, and Evelina Andersson. Joint evaluation for morphological segmentation and syntactic parsing. In *Proceedings of ACL*, 2012.
102. Reut Tsarfaty, Djame Seddah, Yoav Goldberg, Sandra Kuebler, Marie Candito, Jennifer Foster, Yannick Versley, Ines Rehbein, and Lamia Tounsi. Statistical parsing for morphologically rich language (spmrl): What, how and whither. In *Proceedings of the first workshop on Statistical Parsing of Morphologically Rich Languages (SPMRL) at NA-ACL*, 2010.
103. Reut Tsarfaty and Khalil Sima'an. Three-dimensional parametrization for parsing morphologically rich languages. In *Proceedings of IWPT*, 2007.
104. Reut Tsarfaty and Khalil Sima'an. Relational-realizational parsing. In *Proceedings of CoLing*, 2008.
105. Reut Tsarfaty and Khalil Sima'an. Modeling morphosyntactic agreement for constituency-based parsing of modern hebrew. In *Proceedings of NAACL/HLT workshop on Statistical Parsing of Morphologically Rich Languages*, 2010.
106. Reut Tsarfaty, Khalil Sima'an, and Remko Scha. An alternative to head-driven approaches for parsing a (relatively) free word order language. In *Proceedings of EMNLP*, 2009.
107. Arnold M. Zwicky. Heads, bases, and functors. In G.G. Corbett, N. Fraser, and S. McGlashan, editors, *Heads in Grammatical Theory*, pages 292–315. Cambridge University Press, 1993.