2

Mildly Context-Sensitive Grammars for Estimating Maximum Entropy Parsing Models

David Chiang

2.1 Introduction

The maximum-entropy framework provides great flexibility in specifying what features a model may take into account, making it effective for a wide range of natural language processing tasks. But because parameter estimation in this framework involves computations over the whole space of possible labelings, it is unwieldy for the parsing problem, where this space is very large.

Researchers have tried several strategies for efficiently training parsing models in the maximum-entropy framework. Ratnaparkhi's parser (1997) models the probabilities of actions of a pushdown automaton instead of the probabilities of entire parses, but for this reason is susceptible to the label-bias problem (Lafferty et al. 2001). Abney (1997) proposes random sampling of the parse space. Johnson et al. (1999) propose using conditional estimation instead of joint estimation. This reduces the space to the possible parses of a single sentence, which is much smaller but can still be unmanageably large for many grammars. Collins' discriminative models (2000) are cast as parse-reranking models instead of parsing models, and are therefore dependent on his earlier generative model (Collins 1999).

Geman and Johnson (2002) and Miyao and Tsujii (2002) propose compact representations of the parse space that make processing

Proceedings of FGVienna: The 8th Conference on Formal Grammar.
Gerald Penn (ed.).

tractable, at the cost of flexibility. Neither, however, details how these compact representations are obtained. Geman and Johnson refer to an algorithm from the LFG literature (Maxwell and Kaplan 1995) to fill this gap in the former method; the purpose of the present paper is to fill this gap in the latter method. We propose to do this by identifying feature forests with the derivation forests of linear context-free rewriting systems (LCFRSs, defined below).

Both of these methods bring tractability at the cost of some generality: features can no longer be completely arbitrary but only *locally* arbitrary. In Section 2.4 we discuss the nature of the locality restriction imposed by feature forests based on LCFRSs, arguing that these formalisms provide ample flexibility in specifying parsing models. We also include a comparison with Geman and Johnson's approach, finding that feature forests are a special case of the packed representations they use, but that feature forests are more efficient for this special case.

2.2 Background

2.2.1 Maximum entropy models

A maximum-entropy model defines a function $\mathbf{f} : \mathcal{T} \to \mathbb{R}^d$ from parses to feature vectors. Let $f_1, \ldots, f_d$ denote the component functions of $\mathbf{f}$. Then the probability of a parse T given a sentence S has the form

$$P(T \mid S) = \frac{1}{Z} \prod_i \theta_i^{f_i(T)} \tag{2.1}$$

where the θ_i are real-valued feature weights and Z is a normalization constant.

There are various methods for obtaining a conditional maximum-likelihood estimate of the θ_i for models of this form, including iterative-scaling methods and gradient-ascent methods (Berger et al. 1996, Malouf 2002), but all require the computation of

$$E(f_i \mid S) = \sum_T f_i(T) P(T \mid S) \tag{2.2}$$

where T ranges over parses of S. Since, for a parsing model, S can have exponentially many parses, it is not practical to perform this summation by brute force. For the same reason, computing the most probable parse

$$\arg\max_T P(T \mid S)$$

is also not practical to compute by brute force.

2.2.2 Miyao and Tsujii's feature forests

Miyao and Tsujii (2002) present a solution to this problem using the notion of a *feature forest*, a compact representation of the feature vectors of all the possible parses of S. Their formulation views forests as and/or graphs, but we provide here a reformulation in terms of context-free grammars (CFGs).

Definition 1 A *feature forest* $F = \langle G, \mathbf{f} \rangle$ is a CFG G together with a mapping $\mathbf{f}$ from productions of G to feature vectors. We can view F as a parameterized weighted CFG by defining the weight of each production $A \to \beta$ to be

$$\prod_i \theta_i^{f_i(A \to \beta)}$$

where, as in (2.1), the θ_i are the feature weights and the f_i are the component functions of $\mathbf{f}$.

Definition 2 The feature vector of a derivation of a feature forest is the sum of the feature vectors of the occurrences of productions in the derivation.

Definition 3 We say that a feature forest F *represents* a set $\mathcal{T}$ of parses if there is a one-to-one correspondence between derivations of F and parses in $\mathcal{T}$ such that each derivation of F has the same feature vector as the corresponding parse.

Note that when F represents the set of all parses of S, the derivations of F need not yield S; only the feature vectors need to match. Therefore the choice of terminal alphabet for F is immaterial.

Miyao and Tsujii's algorithm is essentially the same as the Inside-Outside algorithm (Baker 1979, Lari and Young 1990), which computes the expected number of times that each production will be used. If F is a feature forest representing all the possible parses of S, then

$$E(f_i \mid S) = \sum_{A \to \beta \in F} f_i(A \to \beta) E(A \to \beta) \tag{2.3}$$

where the expectation $E(A \to \beta)$ is computed by running the Inside-Outside algorithm using the weights of F (it makes no difference algorithmically that F is not a proper PCFG), and then dividing by the normalization constant Z from (2.1).

Similarly, though not noted by Miyao and Tsujii, it is simple to use dynamic programming to efficiently compute the feature vector of the most probable parse (Goodman 1999). Also, their algorithm only works on feature forests which generate a finite number of derivations. It can be extended to the general case (Goodman 1999), but we assume here

that all feature forests generate only a finite number of derivations, as this is not a severe limitation.

Miyao and Tsujii's method cannot be used on its own, because it does not provide any way of constructing a feature forest given a sentence. This is a problem for both training and decoding, since the methods described above for both tasks take a feature forest as their starting-point. The solution we propose below is to identify Miyao and Tsujii's feature forests with the derivation forests of linear context-free rewriting systems. Since they are polynomial-time parsable, and the resulting derivation forests have the same structure as feature forests, they tractably bridge the gap in Miyao and Tsujii's method between sentences and feature forests.

In Section 2.3 we describe the details of this solution, and in Section 2.4 we discuss its theoretical and practical possibilities.

2.3 Linear context-free rewriting systems

2.3.1 Definition

Linear context-free rewriting systems or LCFRSs (Vijay-Shanker et al. 1987, Weir 1988) are grammar formalisms with context-free derivation sets and whose elementary operations do not copy or erase arbitrary substrings. Examples of LCFRSs are CFG, tree-adjoining grammar (TAG), and combinatory categorial grammar (CCG).[1]

Definition 4 A *generalized CFG* G is a tuple $\langle V, S, F, P\rangle$, where

1. V is a finite set of nonterminal symbols,
2. $S \in V$ is a start symbol,
3. $\mathcal{F}$ is a finite set of function symbols, and
4. P is a finite set of productions of the form

$$A_0 \to f(A_1, \ldots, A_n) \tag{2.4}$$

where $n \geq 0$, $f \in \mathcal{F}$, and $A_i \in V$.

A generalized CFG rewrites nonterminal symbols just like a CFG, but the end result is a *term*, for example, $f(g(h()), h())$, rather than a string. This term must be evaluated with respect to some interpretation of the function symbols to produce a derived structure. In an LCFRS the interpretation must obey certain constraints, defined below.

Definition 5 We say that an interpretation function $[\![\cdot]\!]$ on terms is *compositional* if for each function symbol f, there is a unique *composi-*

[1] Some variants of CCG are not LCFRSs: for example, if the coordination schema $X \to X$ **and** X has an infinite number of instantiations.

tion operation $\bar{f}$ such that

$$[\![f(t_1, \ldots, t_n)]\!] = \bar{f}([\![t_1]\!], \ldots, [\![t_n]\!]).$$

Definition 6 A grammar of a *linear context-free rewriting system* is a generalized CFG together with a compositional interpretation function, called the *string yield function*, which maps each function symbol f to a composition operation $\bar{f}$ on string tuples definable as:

$$\bar{f}(\langle x_{11}, \ldots, x_{1m_1}\rangle, \ldots, \langle x_{n1}, \ldots, x_{nm_n}\rangle) = \langle w_1, \ldots, w_m\rangle \tag{2.5}$$

where the w_i are concatenations of the variables x_{ij} and symbols from a terminal alphabet Σ, and every variable which appears on the left-hand side appears exactly once on the right-hand side.

We also define a function ϕ from nonterminal symbols to integers as follows: for each nonterminal A, it must be the case that m in equation (2.5) is the same for all productions with left-hand side A; let $\phi(A)$ be this value.

For example, the composition operations of a CFG are defined over single strings (tuples of length 1) and simply concatenate their arguments. In a TAG, it is convenient to define a *tree* yield function first: each tree composition operation corresponds to an elementary tree γ, taking a number of derived auxiliary trees as arguments and evaluating to γ with the arguments adjoined into it. Define the *yield* of a derived auxiliary tree be $\langle w_l, w_r\rangle$, where w_l and w_r are the yield of the tree to the left and right of the foot, respectively. Then each tree composition operation $\bar{f}$ induces a string composition operation, taking yields of derived auxiliary trees as arguments and evaluating to the yield of the value of $\bar{f}$ on those trees.

Note, in fact, that any grammar which is to produce parse trees, for example, as found in the Penn Treebank, must have a tree yield function, even though the above definition of LCFRS only defines the string yield function. For our purposes, the tree yield function can be defined in any way which is consistent with the string yield function.

2.3.2 Parsing

Several parsing algorithms for equivalents to LCFRS have been described in the literature (Seki et al. 1991, Bertsch and Nederhof 2001). We present here, for completeness, a parsing algorithm as a procedure for intersecting a grammar with a finite-state automaton (Lang 1994, Bar-Hillel et al. 1964). The parsing problem is to compute, given an input string w, the derivation forest for w, that is, a compact representation of all derivations of w under a grammar G. Just as we defined feature forests as CFGs above, we can also define the parse forest

as a CFG: a CFG generating all and only the possible derivations of w. Suppose we have a construction which, given a grammar G and a finite-state automaton M, computes a new grammar G' that generates just those derivations of G (up to nonterminal relabeling) which yield strings accepted by M—call this the intersection of G with M. If we then let M be an automaton accepting the singleton language $\{w\}$, the intersection of G with M is a parse forest of w. Thus an algorithm for intersecting G with a finite-state automaton constitutes a parser for G.

We present below an algorithm for computing the intersection of any LCFRS grammar $G = \langle V, S, \mathcal{F}, P\rangle$ with a finite-state automaton $M = \langle Q, \Sigma, \delta, q_0, Q_f\rangle$. Define a new start symbol S' and a new nonterminal alphabet

$$V' = \{S'\} \cup \bigcup_{A \in V} \{A\} \times Q^{2\phi(A)}$$

Intuitively, each nonterminal $A \in V$ is accompanied by $2\phi(A)$ states, indicating the start and stop states of M when recognizing the substrings yielded by A. Next define a new set of productions P': for each production in P, if the production has the form (2.4) and $\bar{f}$ is defined as in (2.5), we replace A_i with

$$\langle A_i, p_{i1}, q_{i1}, \ldots p_{i,\phi(A_i)}, q_{i,\phi(A_i)}\rangle,$$

for all possible values of $p_{ij}, q_{ij} \in Q$, subject to the following constraints (in all the following $v \in \Sigma^*$):

- If vx_{ij} is a prefix of w_k, then $\delta(p_{0k}, v) = p_{ij}$;
- if $x_{ij}v$ is a suffix of w_k, then $\delta(q_{ij}, v) = q_{0k}$;
- if $x_{ij}vx_{kl}$ is a substring of w_m, then $\delta(q_{ij}, v) = p_{kl}$.

Each occurrence of A_i in a derivation yields $\phi(A_i)$ substrings; p_{ij} and q_{ij} are the start and stop states of the jth substring. The purpose of the constraints is to ensure that the substrings are only combined in ways that are licensed by M.

Finally, for every $q \in Q_f$, add to P' the production

$$S' \to \langle S, q_0, q\rangle$$

Then the final result is the generalized CFG

$$G' = \langle V', S', \mathcal{F}, P'\rangle$$

In the case where M is a straight-line automaton accepting a single string, P' has $\mathcal{O}(|G|\, n^{(r+1)f})$ productions, where n is the number of states in Q, f is the maximum value of $\phi(A)$, $A \in V$, and r is the maximum number of nonterminals on the right-hand side of a production. Therefore the running time of this construction is also $\mathcal{O}(|G|\, n^{(r+1)f})$.

This algorithm always runs in polynomial time, but is not necessarily optimal. The grammar may need to be transformed first to achieve the optimal running time. For example, for CFG, we have $f = 1, r \geq 1$, but if we first convert the grammar into Chomsky normal form, we always have $r \leq 2$, giving a running time of $\mathcal{O}(n^3)$. Similarly, for TAG, we have $f = 2, r \geq 1$, but any TAG can be converted into an equivalent TAG with $r \leq 2$ (Lang 1994), giving $\mathcal{O}(n^6)$.

2.3.3 Feature forests

All that remains is to add features to the LCFRSs:

Definition 7 A *generalized CFG (or LCFRS) with features* is a generalized CFG (resp., LCFRS) together with an interpretation function that maps each function symbol f to a composition operation $\bar{f}$ on feature vectors definable as:

$$\bar{f}(\mathbf{x}_1, \ldots \mathbf{x}_n) = \mathbf{x}_f + \sum \mathbf{x}_i \tag{2.6}$$

where $\mathbf{x}_f$ is a feature vector depending on f.

The parsing algorithm for LCFRSs applies equally well to LCFRSs with features, and the resulting generalized CFG with features can be viewed as a feature forest: just treat each production of the form (2.4) as a production $A_0 \rightarrow A_1 \cdots A_n$ with feature vector $\mathbf{x}_f$. This gives, as desired, a procedure for computing feature forests from strings. Together with Miyao and Tsujii's algorithm, this provides a complete method for training a maximum-entropy parsing model.

2.4 Discussion

Thus an LCFRS becomes a formalism for describing maximum-entropy models: the features are defined not on entire parses, but on the elementary structures of the grammar. Therefore features are no longer completely arbitrary, but only *locally* arbitrary, i.e., arbitrary within the local domains defined by the elementary structures of the grammar. The question is, does this provide enough flexibility to be useful for natural language parsing?

2.4.1 Large domains

First of all, there is no fixed limit on the size of local domains. We can extend an existing parsing model by increasing the size of its local domains, widening the range of possible features. Crucially, this move does not require us to eliminate the original features, as would be the case in a generative model; it will make more features available, while retain the original ones if desired.

For example, when we modify a generative TAG model (Chiang 2000) to use elementary TAG trees with multiple lexical anchors instead of only one, accuracy decreases (Chiang 2003), probably because the smoothing method used does not cope well with the larger elementary trees. But a model that allows locally arbitrary features could circumvent this difficulty. For example, if the elementary trees for PPs were fused with their objects, there could be a feature for the dependency between the preposition and the modified headword and a feature for the dependency between the preposition and the object headword, just as in the original model, but also a feature involving the prepositional object headword and the modified headword, or all three words.

One practical issue that would arise from enlarging the elementary structures would be *grammar* sparseness: if the parser's grammar is obtained by fragmenting trees from a corpus (e.g., Chiang 2000, Hockenmaier and Steedman 2002), then larger structures will be less likely to recur in new data. This problem already occurs with lexicalized grammars, whose coverage can be improved by recombining lexical anchors and structures which have the same part-of-speech tag. An analogous solution for even larger elementary structures would be to define additional recombination points (for example, every S or NP node). Of course, the increase in the effective size of the grammar will also increase parsing time.

2.4.2 Overlapping domains

Local domains may also overlap by a finite amount. For example, in a CFG two adjacent local domains have a nonterminal symbol in common. By means of grammar transformations, one can use this overlap as a conduit through which information can be propagated between local domains, effectively extending the domains of features. This is the technique lexicalized PCFG parsers use, for example: they propagate headwords up through the tree to capture bilexical dependencies. We note two consequences of this observation.

First, it is possible to "squeeze" strong generative capacity out of a grammar formalism without increasing its weak generative capacity (Joshi 2003), and this extra power can be demonstrated formally in certain respects (Chiang 2002); for example, regular form TAG (Rogers 1994) is weakly equivalent to CFG, but its synchronous version generates more string relations than synchronous CFG. It might be thought that something similar could be shown for feature forests, that a feature forest based, for example, on regular form TAG allows more definable features than one based on CFG. But this is not true in general. Often a grammar G of a "squeezed" formalism can be converted into a gram-

mar G' of the "unsqueezed" formalism such that the derivations by G can be recovered from the derivations by G'.

Definition 8 We say that a grammar G' of an LCFRS *simulates*[2] a grammar G if there is a bijection $[\![\cdot]\!]_s$ from the terms of G' to the terms of G such that $[\![[\![t]\!]_s]\!]_G = [\![t]\!]_{G'}$ for all terms t of G'.

If $[\![\cdot]\!]_s$ is compositional and "linear" by analogy with Definition 6, that is, each of its composition operations introduces a fixed set of function symbols and only rearranges existing symbols without copying or erasing, then it is easy to show that G' can define all the features that G can. This is because as G' simulates G, it can emit the same function symbols (and therefore the same feature vectors) that G' would in its derivation, though possibly in a very different order; but here order does not matter because addition of feature vectors is commutative. Therefore G' can define any feature that G can.

This is the case with regular form TAG (as defined by Chiang (2001)) relative to CFG, and tree-local multicomponent TAG (Weir 1988) relative to TAG. (Tree insertion grammar (TIG, Schabes and Waters 1995) can be simulated by CFG, but the simulation only preserves strings, not trees.) We expect this relationship to obtain whenever a grammar formalism can be parsed using an algorithm similar to that of a simpler formalism, because such parsers typically work by computing $[\![\cdot]\!]_s$.[3] So although this type of "squeezing" may allow more felicitous descriptions of parsing models, it does not actually allow any new models to be described.

On the other hand, this also means that one can model a wide range of features without even going beyond CFG. Any decision that can be made by a finite-state tree automaton can be made into a feature that is local with respect to CFG. Nearly all the features of Collins' discriminative reranker, for example, fall into this class.[4] What's more, since finite-state tree automata are closed under intersection, they can be freely combined. The blow-up in the size of the grammar, however, may be considerable.

[2]This definition is somewhat broader than the original definition (Chiang 2001).

[3]The Lambek calculus would be an example of a weakly context-free formalism with no CFG-like parsing algorithm (Pentus 1993, 2003)

[4]The distance features cannot be encoded by a tree automaton. However, one of the features, which fires when the distance between a headword and its modifier is not greater than some threshold, could be so encoded, because the threshold is limited to 9.

2.4.3 Interleaving domains

Local domains also need not be contiguous regions of the parse tree, because parts of a local domain may end up in very distant parts of the parse tree. (For example, nodes of a TAG or TIG elementary tree get stretched apart when an auxiliary tree adjoins between them.) The parameter f above (Section 2.3.2) controls how many discontinuous substrings the yield of a subderivation may have. Since parsing time depends on this parameter, there is a tradeoff between interleaving of local domains and parsing efficiency.

But the existence of wide-coverage TAGs for English and French suggests that many linguistically-motivated features do not require $f > 2$. Moving from CFG to TAG (or just to TIG) would make it possible to define features capturing nonlocal dependencies, like the dependency between the subject and embedded verb in a raising or control construction ("The *cat* wants to *sit* on the mat"), or between the verb in a relative clause and the noun modified by the relative clause ("the *cat* which *sat* on the mat"). Moving from a less powerful formalism to a more powerful one would make more features available, without affecting the original ones.

One might also consider extending further to multicomponent TAG. The standard TAG extraction method uses head rules to propagate headwords through each training example, then defines local domains such that two nodes are in the same local domain if and only if they have the same headword. This method extends straightforwardly to noncontiguous local domains. For example, one could preserve the empty elements in the Penn Treebank and assign them to the same local domains as the nodes with which they are coindexed. The extracted multicomponent TAG would localize filler-gap dependencies and could be used to reconstruct them.

2.4.4 Comparison with Maxwell-Kaplan packed representations

Geman and Johnson (2002) present a similar method for parsing and estimation of stochastic unification-based grammars. Like the present method, it uses packed representations of sets of parses. The representation they use comes from earlier work by Maxwell and Kaplan (1995), which we define briefly here. In this view, a parse $T \in \mathcal{T}$ is thought of as a set of *parse-features* (called simply "features" in their terminology).

Definition 9 A *Maxwell-Kaplan packed representation* (or MKPR) is a tuple $\langle \mathcal{P}, \mathcal{X}, N, \alpha \rangle$, where

- $\mathcal{P}$ is a finite set of possible parse-features,

- $\mathbf{X}$ is a finite vector of variables, each variable X_i ranging over a finite set $\mathcal{X}_i$,
- N is a finite set of conditions on $\mathbf{X}$, and
- α is a function mapping each parse-feature $p \in \mathcal{P}$ to a condition (not necessarily in N) on $\mathbf{X}$.

Definition 10 We say that an assignment of values to $\mathbf{X}$ *identifies* a parse T if it satisfies the conditions (given by α) of the parse-features in T, and no others.

Definition 11 We say that an MKPR R *represents* a set $\mathcal{T}$ of parses if there is a one-to-one correspondence between assignments of $\mathbf{X}$ satisfying N and parses in $\mathcal{T}$ such that the parse identified by an assignment is equal to the corresponding parse.

As with feature forests, Geman and Johnson's method also imposes a notion of locality on features: the feature vector ("property vector," in their terminology) of a parse is the sum of the feature vectors of its parse-features.

Definition 12 Let $\mathbf{f}$ be a mapping from parse-features to feature vectors. Then the feature vector of a parse T is $\sum_{p \in T} \mathbf{f}(p)$.

Any feature forest F can be converted into an MKPR. Let V be the nonterminal alphabet of F, and P the productions of F. First we must assume that no nonterminal symbol can be used twice in a single derivation. If this is not the case, then the offending nonterminals must first be renamed apart, duplicating productions as necessary (see below for analysis).

The parse-features are just the productions in P, and each nonterminal symbol $A \in V$ corresponds to a variable X_A, ranging over the set $\{\beta \mid A \to \beta \in P\} \cup \{*\}$. Each X_A gives the right-hand side with which A gets rewritten ($*$ means that A does not get used at all). Next we define the conditions on $\mathbf{X}$. Let α assign to the parse-feature $A \to \beta$ the condition that $X_A = \beta$. The conditions in N ensure that $X_S \neq *$, and that for each production $A \to \beta$ and each nonterminal B occurring in β, $X_B \neq *$ if and only if $X_A = \beta$.

How does the size of the resulting MKPR compare with the original feature forest? Ordinarily it should be the same, but we noted above that if a nonterminal symbol of the feature forest can occur twice in a single derivation, the two occurrences must be renamed apart, duplicating productions as necessary. This can only happen in two cases. Let $A' = \langle A, p_1, q_1, \ldots, p_{\phi(A)}, q_{\phi(A)} \rangle$ stand for a nonterminal of the feature forest (as produced by the construction of Section 2.3.2). Then the two cases are when $A' \stackrel{*}{\Rightarrow} \cdots A' \cdots$, which is already ruled out by the

assumption that the feature forest has a finite number of derivations, or when $S' \stackrel{*}{\Rightarrow} \cdots A' \cdots A' \cdots$, which is possible in a parse forest only if $p_i = q_i$ for all i, that is, the two occurrences of A have the same empty span. So in most cases the MKPR will be the same size as the feature forest, though in the worst case, the blowup could be arbitrarily large.

How does the time complexity of processing the resulting MKPR compare with processing the original feature forest? Geman and Johnson use graphical techniques to perform computations on MKPRs, the complexity of which is a function of the pattern of dependencies between the X_A variables in the conditions. For MKPRs produced by the above construction, for each production $A \to \beta$ there is a dependency between X_A and X_B, where B occurs in β. If every nonterminal appears only once among the right-hand sides (i.e., there is no subderivation sharing), then the dependency graph will be a tree, the most efficient case. But subderivation sharing is to be expected; indeed, dynamic programming would not provide any benefit if there were none. So Geman and Johnson's algorithm will not, in general, achieve its optimal case. By contrast, Miyao and Tsujii's algorithm, because it takes advantage of the particular structure of feature forests, always matches the optimal case of Geman and Johnson's algorithm.

2.4.5 Extension to range-concatenation grammars

Most of the ideas presented here extend straightforwardly to range-concatenation grammars, or RCGs (Boullier 2000),[5] which are more powerful still than LCFRSs, generating exactly the languages which are recognizable in polynomial time (Bertsch and Nederhof 2001). Because the derivations forests of RCGs can be thought of as context-free grammars, just as with LCFRSs, the method described above of attaching feature vectors to productions and estimating parameters applies equally well to RCGs as to LCFRSs.

The class of range-concatenation languages has the convenient property of being closed under intersection. A negative consequence of this is that it does not seem possible in general to define generative statistical models for RCGs. The method described here circumvents this problem, providing a novel way of defining statistical RCGs. Moreover, the ability of RCGs to superimpose different analyses via intersection would seem to be ideal for defining models with overlapping features. However, the fact that this intersection ability does not carry over straightforwardly to trees may reduce its usefulness for defining maximum-entropy parsing models. Further investigation into such models is necessary.

[5] Thanks to an anonymous reviewer for this suggestion.

2.5 Conclusion

We have described how to integrate Miyao and Tsujii's algorithm into a full system for training on treebank data and parsing new sentences, and discussed the advantages and limitations of this approach relative to history-based models and relative to Geman and Johnson's approach. We have shown how the descriptive power of a formalism correlates with its flexibility for specifying parsing models; it remains to be seen whether more powerful formalisms can give rise to more accurate parsing models in practice.

Acknowledgements

This research was supported in part by NSF grants SBR-89-20230 and ITR EIA-02-05456. I would like to thank Fernando Pereira, William Schuler, Dan Bikel, Aravind Joshi, and the anonymous reviewers for their valuable help. *S. D. G.*

References

Abney, S. P. 1997. Stochastic attribute-value grammars. *Computational Linguistics* 23(4):597–618.

Baker, J. K. 1979. Trainable grammars for speech recognition. In *Proceedings of the Spring Conference of the Acoustical Society of America*, pages 547–550.

Bar-Hillel, Y., M. Perles, and E. Shamir. 1964. On formal properties of simple phrase structure grammars. In Y. Bar-Hillel, ed., *Language and Information: Selected Essays on their Theory and Application*, pages 116–150. Addison-Wesley.

Berger, A. L., S. A. Della Pietra, and V. J. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics* 22(1):39–71.

Bertsch, E. and M.-J. Nederhof. 2001. On the complexity of some extensions of RCG parsing. In *Proceedings of the 7th International Workshop on Parsing Technologies*, pages 66–77. Beijing.

Boullier, P. 2000. Range concatenation grammars. In *Proceedings of the 6th International Workshop on Parsing Technologies*, pages 53–64. Trento.

Chiang, D. 2000. Statistical parsing with an automatically-extracted tree adjoining grammar. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 456–463. Hong Kong.

Chiang, D. 2001. Constraints on strong generative power. In *Proceedings of the 39th Meeting of the Association for Computational Linguistics*, pages 124–131. Toulouse.

Chiang, D. 2002. Putting some weakly context-free formalisms in order. In *Proceedings of the 6th International Workshop on TAG and Related Formalisms*, pages 11–18.

Chiang, D. 2003. Statistical parsing with an automatically extracted tree adjoining grammar. In R. Bod, R. Scha, and K. Sima'an, eds., *Data-Oriented Parsing*, pages 299–316. Palo Alto: CSLI Publications.

Collins, M. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Collins, M. 2000. Discriminative reranking for natural language parsing. In *Proceedings of the 17th International Conference on Machine Learning*, pages 175–182. San Mateo: Morgan Kaufmann.

Geman, S. and M. Johnson. 2002. Dynamic programming for parsing and estimation of stochastic unification-based grammars. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 279–286. Philadelphia, PA.

Goodman, J. 1999. Semiring parsing. *Computational Linguistics* 25:573–605.

Hockenmaier, J. and M. Steedman. 2002. Acquiring compact lexicalized grammars from a cleaner treebank. In *Proceedings of the Third International Conference on Language Resources and Evaluation*, pages 1974–1981.

Johnson, M., S. Geman, S. Canon, Z. Chi, and S. Riezler. 1999. Estimators for stochastic "unification-based" grammars. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 535–541. College Park, MD.

Joshi, A. K. 2003. A note on the strong and weak generative powers of formal systems. *Theoretical Computer Science* 293:243–259. Originally presented at TAG+5 in 2000.

Lafferty, J., A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*. San Mateo: Morgan Kaufmann.

Lang, B. 1994. Recognition can be harder than parsing. *Computational Intelligence* 10(4):484–494. Special Issue on Tree Adjoining Grammars.

Lari, K. and S. J. Young. 1990. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech and Language* 4:35–56.

Malouf, R. 2002. A comparison of algorithms for maximum entropy parameter estimation. In *Proceedings of the 6th Conference on Natural Language Learning*, pages 49–55.

Maxwell, J. T., III and R. M. Kaplan. 1995. A method for disjunctive constraint satisfaction. In M. Dalrymple, R. M. Kaplan, J. T. Maxwell, III, and A. Zaenen, eds., *Formal Issues in Lexical-Functional Grammar*. Palo Alto: CSLI Publications.

Miyao, Y. and J. Tsujii. 2002. Maximum entropy estimation for feature forests. In *Proceedings of the Second International Conference on Human Language Technology Research*, pages 292–297. San Diego.

Pentus, M. 1993. Lambek grammars are context-free. In *Proceedings of the 8th Annual IEEE Symposium on Logic in Computer Science*, pages 429–433. Los Alamitos.

Pentus, M. 2003. Lambek calculus is NP-complete. Technical Report TR-2003005, CUNY Ph.D. Program in Computer Science.

Ratnaparkhi, A. 1997. A linear observed time statistical parser based on maximum entropy models. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing (EMNLP-2)*. Providence, RI.

Rogers, J. 1994. Capturing CFLs with tree adjoining grammars. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, pages 155–162. Las Cruces, NM.

Schabes, Y. and R. C. Waters. 1995. Tree insertion grammar: a cubic-time parsable formalism that lexicalizes context-free grammar without changing the trees produced. *Computational Linguistics* 21:479–513.

Seki, H., T. Matsumura, M. Fujii, and T. Kasami. 1991. On multiple context-free grammars. *Theoretical Computer Science* 88:191–229.

Vijay-Shanker, K., D. Weir, and A. Joshi. 1987. Characterizing structural descriptions produced by various grammatical formalisms. In *Proceedings of the 25th Annual Meeting of the Association for Computational Linguistics*, pages 104–111. Stanford.

Weir, D. 1988. *Characterizing Mildly Context-Sensitive Grammar Formalisms*. Ph.D. thesis, University of Pennsylvania.