Broad-Coverage Parsing Using Human-Like Memory Constraints

William Schuler*
University of Minnesota

Samir AbdelRahman**
Cairo University

Tim Miller*
University of Minnesota

Lane Schwartz*
University of Minnesota

Human syntactic processing shows many signs of taking place within a general-purpose short-term memory. But this kind of memory is known to have a severely constrained storage capacity — possibly constrained to as few as three or four distinct elements. This article describes a model of syntactic processing that operates successfully within these severe constraints, by recognizing constituents in a right-corner transformed representation (a variant of left-corner parsing) and mapping this representation to random variables in a Hierarchical Hidden Markov Model, a factored time-series model which probabilistically models the contents of a bounded memory store over time. Evaluations of the coverage of this model on a large syntactically annotated corpus of English sentences, and the accuracy of a bounded-memory parsing strategy based on this model, suggest this model may be cognitively plausible.

1. Introduction

It is an interesting possibility that human syntactic processing may occur entirely within a general-purpose short-term memory. Like other short-term memory processes, syntactic processing is susceptible to degradation if short-term memory capacity is loaded, for example, when readers are asked to retain lists of words while reading (Just and Carpenter 1992); and memory of words and syntax degrades over time within and across sentences (Sachs 1967; Jarvella 1971), unlike semantics and discourse information about referents from other sentences (Ericsson and Kintsch 1995). But short-term memory is known to have severe capacity limitations of perhaps no more than three to four distinct elements (Miller 1956; Cowan 2001). These limits may seem

* Department of Computer Science and Engineering, 200 Union St. SE, Minneapolis, MN 55455. E-mail: schuler@cs.umn.edu; tmill@cs.umn.edu; lane@cs.umn.edu.
** Computer Science Department, Faculty of Computers and Information, 5 Dr. Ahmed Zewail Street, Postal Code: 12613, Orman, Giza, Egypt. E-mail: s.abdelrahman@fci-cu.edu.eg.

Submission received: 14 February 2008; revised submission received: 31 October 2008; accepted for publication: 27 May 2009.

too austere to process the rich tree-like phrase structure commonly invoked to explain word-order regularities in natural language.

This article aims to show that they are not. The article describes a comprehension model, based on a right-corner transform—a reversible tree transform related to the left-corner transform of Johnson (1998a)—that associates familiar phrase structure trees with the contents of a memory store of three to four partially completed constituents over time. Coverage results on the large syntactically annotated Penn Treebank corpus show a vast majority of naturally occurring sentences can be recognized using a memory store containing a maximum of only three incomplete constituents, and nearly all sentences can be recognized using four, consistent with estimates of human short-term memory capacity.

This transform reduces memory usage in incremental (left to right) processing by transforming right-branching constituent structures into left-branching structures, allowing child constituents to be composed with parent constituents before either have been completely recognized. But because this composition identifies an incomplete child as the awaited portion of an incomplete parent, it implicitly predicts that this child constituent will be the rightmost (i.e., last) child of the parent, before this child has been completely recognized. Parsing accuracy results on the Penn Treebank using a Hierarchical Hidden Markov Model (Murphy and Paskin 2001)—essentially a probabilistic pushdown automaton with a bounded pushdown store—show that this prediction can be reliably learned from training data.

The remainder of this article is organized as follows: Section 2 describes some related models of human syntactic processing using a bounded memory store; Section 3 describes a Hierarchical Hidden Markov Model (HHMM) framework for statistical parsing using this bounded store of incomplete constituents; Section 4 describes the right-corner transform and how it relates conventional phrase structure to incomplete constituents in a bounded memory store; Section 5 describes an experiment to estimate the level of coverage of the Penn Treebank corpus that can be achieved using this transform with various memory limits, given a linguistically motivated binarization of this corpus; and Section 6 gives accuracy results of this bounded-memory model trained on this corpus, given that some amount of incremental prediction (as described earlier) must be involved.

2. Bounded-Memory Parsing

One of the earliest bounded-memory parsing models is that of Marcus (1980). This model maintains a bounded store of complete but unattached constituents as a buffer, and operates on them using a variety of specialized memory manipulation operations, deferring certain attachment decisions until the contents of this buffer indicate it is safe to do so. (In contrast, the model described in this article maintains a store of *incomplete* constituents using ordinary stack-like push and pop operations, defined to allow constituents to be composed before being completely recognized.) The Marcus parser provides a bounded-memory explanation for human difficulties in processing garden path sentences: for example, *the horse raced past the barn fell*, with intended interpretation $[_{NP}$ *the horse* $[_{RC}$ *(which was) raced past the barn]] fell* (Bever 1970), in which *raced* seems like the main verb of the sentence until the word *fell* is encountered. But this explanation due to memory exhaustion is not compatible with observations of unproblematic parsing of sentences such as these when contextual information is provided in advance: for example, *two men on horseback had a race; one went by the meadow, and the other went by the barn* (Crain and Steedman 1985).

Ades and Steedman (1982) introduce the idea of composing incomplete constituents to reduce storage demands in incremental processing using Combinatorial Categorial Grammar (CCG), avoiding the need to maintain large buffers of complete but unattached constituents. The right-corner transform described in this article composes incomplete constituents in very much the same way, but CCG is essentially a competence model, in that it seeks to unify lexical category representations used in processing with learned generalizations about argument structure, whereas the model described herein is exclusively a performance model, allowing generalizations about lexical argument structures to be learned in some other representation, then combined with probabilistic information about parsing strategies to yield a set of derived incomplete constituents. As a result, the model described in this article has a freer hand to satisfy strict working memory bounds, which may not permit some of the alternative composition operations proposed in the CCG account, thought to be associated with available prosody and quantifier scope analyses.[1]

Johnson-Laird (1983) and Abney and Johnson (1991) propose a pure processing account of memory capacity limits in parsing ordinary phrase structure trees. The Johnson-Laird and Abney and Johnson models adopt a left-corner parsing strategy, of which the right-corner transform introduced in this article is a variant, in order to bring memory usage for most parsable sentences to within seven or so active or awaited phrase structure constituents. This account may be used to explain human processing difficulties in processing triply center-embedded sentences like *the rat that the cat that the dog chased killed ate the malt*, with intended interpretation $[_{NP}$ *the rat that* $[_{NP}$ *the cat that* $[_{NP}$ *the dog] chased] killed] ate the malt* (Chomsky and Miller 1963). But this explanation does not account for examples of triply center-embedded sentences that typically do not cause processing problems: $[_{NP}$ *that* $[_{NP}$ *the food that* $[_{NP}$ *John] ordered] tasted good] pleased him* (Gibson 1991). Moreover, the apparent competition between comprehension of center-embedded object relatives and retention of unrelated words in general-purpose memory (Just and Carpenter 1992) suggests that general-purpose memory is (or at least, can be) used to store incomplete constituents during comprehension. This would predict three or four elements of reliable storage, rather than seven (Cowan 2001). The transform-based model described in this article exploits a conception of chunking (Miller 1956) to combine pairs of active and awaited constituents from the Abney and Johnson analysis, connected by recognized structure, in order to operate within estimates of human short-term memory bounds.

Because of these counterexamples to the memory-exhaustion explanation of garden path and center-embedding difficulties, recent work has turned to explanations other than memory exhaustion for these phenomena. Lewis and Vasishth (2005) attribute processing errors to activation interference among stored constituents that have similar syntactic and semantic roles. Hale's surprisal (2001) and entropic model (2006) link human processing difficulties to significant changes in the relative probability of competing hypotheses in incremental parsing, such that if activation is taken to be a mechanism for probability estimation, processing difficulties may be ascribed to the relatively slow speed of activation change within the brain (or to collapsing activation when probabilities grow too small, as in the case of garden path sentences). These models explain many processing difficulties without invoking memory limits, and are

1 The lack of support for some of these available scope analyses may not necessarily be problematic for the present model. The complexity of interpreting nested raised quantifiers may place them beyond the capability of human interactive incremental interpretation, but not beyond the capability of post hoc interpretation (understood after the listener has had time to think about it).

compatible with brain imaging evidence of increased cortical activity and recruitment of auxiliary brain areas during periods of increased uncertainty in sentence processing (Just and Varma 2007). But if interference or changing activation is posited as the source of processing difficulty, and delays are not linked to memory exhaustion per se, then these theories do not explain how (or whether) syntactic processing operates within general-purpose short-term memory.

Toward this end, this article will specifically evaluate the claim that syntactic processing can be performed entirely within general-purpose short-term memory by using this memory to store unassimilated incomplete syntactic constituents, derived through a right-corner transform from basic properties of phrase structure trees. As a probabilistic incremental parser, the model described in this article is compatible with surprisal-based explanations of processing difficulties; it is, however, in some sense orthogonal, because it models a different dimension of resource allocation. The surprisal framework models allocation of processing resources (in this case, activation) among disjunctions of competing hypotheses, which are maintained for some amount of time in parallel, whereas the framework described here can be taken to model the allocation of processing resources (in this case, memory elements) among conjunctions of incompletely recognized constituents *within* each competing hypothesis.[2] Thus, in this view, there are two ways to simultaneously activate multiple concepts: disjunctively (sharing activation among competing hypotheses) and conjunctively (sharing activation among unassimilated constituents within a hypothesis). But only the inner conjunctive allocation corresponds to the familiar discretely bounded store of short-term memory as described by Miller (1956); the outer disjunctive allocation treats activation as a continuous resource in which like-valued pockets expand and contract as they are reinforced or contradicted by incoming observations. Indeed, it would be surprising if these two dimensions of resource allocation did not exist: the former, because it would contradict years of observations about the behavior of short-term memory; and the latter, because it would require neural activation spreading to be instantaneous and uniform, contradicting most neuropsychological evidence. Levy (2008) compares the allocation of activation in this kind of framework to the distributed allocation of resources in a particle filter (Gordon, Salmond, and Smith 1993), an approximate inference technique for probabilistic time-series models in which particles in a (typically fixed) reservoir are assigned randomly sampled hypotheses from learned transition probabilities, essentially functioning as units of activation. The model described in this paper qualifies this analogy by positing that each individual particle in this reservoir endorses a coherent hypothesis about the contents of a three- to four-element memory store at any given time, rather than about an entire unbounded phrase structure tree.[3]

2 Probability distributions in entropy-based models like Hale's are typically assumed to be defined over sets of hypotheses pursued in parallel, but other interpretations (for example, lookahead-based deterministic models) are possible. The model described in this article is also compatible with deterministic parsing frameworks, in which case it models allocation of processing resources among incompletely-recognized constituents within a single non-competing hypothesis.

3 Pure connectionist models of syntactic processing (Elman 1991; Berg 1992; Rohde 2002) attempt to unify storage of constituent structure with that of ambiguous alternative analyses, but the memory demands of systems based on this approach typically do not scale well to broad-coverage parsing. Recent results for using self-organizing maps as a unified memory resource are encouraging (Mayberry and Miikkulainen 2003), but are still limited to parsing relatively short travel planning queries with limited syntactic complexity. Hybrid systems that generate explicit alternative hypotheses with explicit stacked-up constituents, and use connectionist models for probability estimation over these hypotheses (Henderson 2004) typically achieve better performance in practice.

Previous memory-based explanations of problematic sentences (explaining garden path effects as exceeding a bound of four complete but unattached constituents, or explaining center embedding difficulties as exceeding a bound of seven active or awaited constituents) have been shown to underestimate human sentence processing capacity when equally complex but unproblematic sentences were examined. The hypothesis advanced in this article, that human sentence processing uses general-purpose short-term memory to store incomplete constituents as defined by a right-corner transform, leaves the explanation of several negative examples of unparsable garden path and center embedding sentences to orthogonal models of surprisal or interference. But in order to determine whether this right-corner memory hypothesis still underestimates human sentence processing capacity, a corpus study was performed on two complementary corpora of transcribed spontaneous speech and newspaper text, manually annotated with phrase structure trees (Marcus, Santorini, and Marcinkiewicz 1993). These spontaneous speech and newspaper text corpora contain only attested positive examples of parsable sentences, but they may be considered complementary for this purpose because the complexity of spontaneous speech may somewhat *understate* human recognition capacity (potentially limiting it to the cost of spontaneously generating sentences in an unusual social context), and the complexity of newspaper text may somewhat *overstate* human recognition capacity (though it is composed and edited to be readable, it is still composed and edited off-line), so results from these corpora may be taken together to suggest generous and conservative upper bounds on human processing capacity.

3. Bounded-Memory Parsing with a Time Series Model

The framework adopted in this article is a factored HMM-like time series model, which maintains a probability distribution over the contents of a bounded set of random variables over time, corresponding to hypothesized stores of memory elements. The random variables in this store may be understood as simultaneous activations in a cognitive model (similar to the superimposed **roles** described by Smolensky and Legendre [2006]), and the probability distribution over these stores may be thought of as competing pockets of activation, as described in the previous section. Some of these variables persist as elements of the short-term memory store, and some are transient as results of hypothesized compositions, which are estimated and immediately discarded or folded into the persistent store according to the dependencies in the model. The variables have values or contents (or **fillers**)—in this case incomplete constituent categories—that change over time, and although these values may be uncertain, the set of hypothesized contents of this memory store at any given point in time are collectively constrained to form a coherent (but possibly incomplete) syntactic analysis of a sentence.

The particular model used here is an HHMM (Murphy and Paskin 2001), which mimics a bounded-memory pushdown automaton (PDA), supporting simple push and pop operations on a bounded stack-like memory store. A time-series model is used here instead of an explicit stack machine, first because the probability model is well defined on a bounded memory store, and second because the plasticity of the random variables that mimic stack behavior in this model makes the model cross-linguistically attractive. By evoking additional random variables and dependencies, the model can be defined (or presumably, trained) to mimic other types of automata, such as extended pushdown automata (EPDAs) recognizing tree-adjoining languages with crossed and nested dependencies, as have been hypothesized for languages like Dutch (Shieber

1985). However, the remainder of this article will only discuss random variables and dependencies necessary to mimic a bounded stack pushdown automaton.

3.1 Hierarchical HMMs

Hierarchical Hidden Markov Models (Murphy and Paskin 2001) are essentially Hidden Markov Models factored into some fixed number of stack-like elements at each time step.

HMMs characterize speech or text as sequences of hidden states q_t (which may consist of phones, words, or other hypothesized syntactic or semantic information), and observed states o_t at corresponding time steps t (typically short, overlapping frames of an audio signal, or words or characters in a text processing application). A most likely sequence of hidden states $\hat{q}_{1..T}$ can then be hypothesized given any sequence of observed states $o_{1..T}$:

$$\hat{q}_{1..T} = \underset{q_{1..T}}{\operatorname{argmax}}\, \mathsf{P}(q_{1..T} \mid o_{1..T}) \tag{1}$$

$$= \underset{q_{1..T}}{\operatorname{argmax}}\, \mathsf{P}(q_{1..T}) \cdot \mathsf{P}(o_{1..T} \mid q_{1..T}) \tag{2}$$

$$\stackrel{\text{def}}{=} \underset{q_{1..T}}{\operatorname{argmax}} \prod_{t=1}^{T} \mathsf{P}_{\Theta_{\mathrm{A}}}(q_t \mid q_{t-1}) \cdot \mathsf{P}_{\Theta_{\mathrm{B}}}(o_t \mid q_t) \tag{3}$$

using Bayes' Law (Equation 2) and Markov independence assumptions (Equation 3) to define a full $\mathsf{P}(q_{1..T} \mid o_{1..T})$ probability as the product of a **Language Model (Θ_{A})** prior probability $\mathsf{P}(q_{1..T}) \stackrel{\text{def}}{=} \prod_t \mathsf{P}_{\Theta_{\mathrm{A}}}(q_t \mid q_{t-1})$ and an **Observation Model (Θ_{B})** likelihood probability $\mathsf{P}(o_{1..T} \mid q_{1..T}) \stackrel{\text{def}}{=} \prod_t \mathsf{P}_{\Theta_{\mathrm{B}}}(o_t \mid q_t)$ (Baker 1975; Jelinek, Bahl, and Mercer 1975).

Language model transitions $\mathsf{P}_{\Theta_{\mathrm{A}}}(q_t \mid q_{t-1})$ over complex hidden states q_t can be modeled using synchronized levels of stacked-up component HMMs in an HHMM, analogous to a shift-reduce parser or pushdown automaton with a bounded stack. HHMM transition probabilities are calculated in two phases: a "reduce" phase (resulting in an intermediate, transient final-state variable f_t), modeling whether component HMMs terminate; and a "shift" phase (resulting in a persistent modeled state q_t), in which unterminated HMMs transition and terminated HMMs are re-initialized from their parent HMMs. Variables over intermediate f_t and modeled q_t states are factored into sequences of depth-specific variables—one for each of D levels in the HHMM hierarchy:

$$f_t = \langle f_t^1 \ldots f_t^D \rangle \tag{4}$$

$$q_t = \langle q_t^1 \ldots q_t^D \rangle \tag{5}$$

Transition probabilities are then calculated as a product of transition probabilities at each level, using level-specific 'reduce' $\Theta_{\mathrm{F},d}$ and 'shift' $\Theta_{\mathrm{Q},d}$ models:

$$\mathsf{P}_{\Theta_{\mathrm{A}}}(q_t \mid q_{t-1}) = \sum_{f_t} \mathsf{P}(f_t \mid q_{t-1}) \cdot \mathsf{P}(q_t \mid f_t\, q_{t-1}) \tag{6}$$

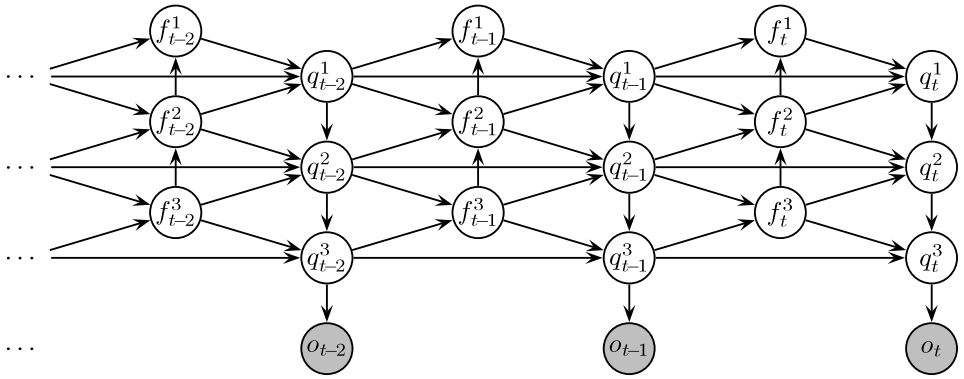


Figure 1
Graphical representation of a Hierarchical Hidden Markov Model. Circles denote random variables, and edges denote conditional dependencies. Shaded circles denote variables with observed values.

$$\stackrel{\text{def}}{=} \sum_{f_t^1 \ldots f_t^D} \prod_{d=1}^{D} \mathsf{P}_{\Theta_{\mathrm{F},d}}(f_t^d \mid f_t^{d+1} q_{t-1}^d q_{t-1}^{d-1}) \cdot \prod_{d=1}^{D} \mathsf{P}_{\Theta_{\mathrm{Q},d}}(q_t^d \mid f_t^{d+1} f_t^d \; q_{t-1}^d q_t^{d-1}) \tag{7}$$

with f_t^{D+1} and q_t^0 defined as constants. In these equations, probabilities are marginalized or summed over all combinations of intermediate variables $f_t^1 \ldots f_t^D$, so only the memory store contents $q_t^1 \ldots q_t^D$ persist across time steps.[4] A graphical representation of an HHMM with three depth levels is shown in Figure 1.

The independence assumptions in this model can be psycholinguistically motivated. Independence across time points t (Equation 3) arise naturally from causality: Any change to a memory store configuration to generate a configuration at time step $t+1$ should depend only on the current memory store configuration at time step t; memory operations should not be able to peek backward or forward in time to consult past or future memory stores. Independence across depth levels d (Equation 7) arise naturally from uncertainty about the structure between incomplete constituent chunks (this property of right-corner transform categories is elaborated in Section 4).[5]

Shift and reduce probabilities can now be defined in terms of finitely recursive HMMs with probability distributions over recursive expansion, transition, and reduction of states at each depth level. In the version of HHMMs used in this paper, each modeled variable is a syntactic state $q_t^d \in G \times G$ (describing an incomplete constituent consisting of an active grammatical category from domain G and an awaited grammatical category from domain G—for example, an incomplete constituent S/NP consisting of an active sentence S awaiting a noun phrase constituent NP); and each intermediate

4 In Viterbi decoding, probabilities over intermediate variables may be maximized rather than marginalized, but in any case the intermediate variables do not persist.

5 Also, the fact that this is a generative model, in which observations are conditioned on hypotheses, then flipped using Bayes' Law (Equation 2)—as opposed to a discriminative or conditional model, in which hypotheses are conditioned directly on observations—is also appealing as a human model, in that it allows the same architecture to be used for both recognition and generation. This is a desirable property for modeling split utterances, in which interlocutors complete one another's sentences (Lerner 1991; Helasvuo 2004).

variable is a reduction or non-reduction state $f_t^d \in G \cup \{\mathbf{1}, \mathbf{0}\}$ (indicating, respectively, a reduction of incomplete constituent q_{t-1}^d to a complete right child constituent of some grammatical category from domain G, or a non-reduction of q_{t-1}^d as a unary or left child, as defined in Section 4). An intermediate variable f_t^d at depth d may indicate reduction or non-reduction according to $\Theta_{\text{F-Reduction},d}$ if there is a reduction at the depth level immediately below d, but must indicate non-reduction ($f_t^d = \mathbf{0}$) with probability 1 if there is no reduction below:[6]

$$\mathsf{P}_{\Theta_{\text{F},d}}(f_t^d \mid f_t^{d+1} q_{t-1}^d q_{t-1}^{d-1}) \stackrel{\text{def}}{=} \begin{cases} \text{if } f_t^{d+1} \notin G : [f_t^d = \mathbf{0}] \\ \text{if } f_t^{d+1} \in G : \mathsf{P}_{\Theta_{\text{F-Reduction},d}}(f_t^d \mid q_{t-1}^d, q_{t-1}^{d-1}) \end{cases} \tag{8}$$

where $f_t^{D+1} = \mathbf{1}$ and $q_t^0 = \mathbf{ROOT}$.

Shift probabilities over the modeled variable q_t^d at each level are defined using level-specific transition $\Theta_{\text{Q-Transition},d}$ and expansion $\Theta_{\text{Q-Expansion},d}$ models:

$$\mathsf{P}_{\Theta_{\text{Q},d}}(q_t^d \mid f_t^{d+1} f_t^d \; q_{t-1}^d q_t^{d-1}) \stackrel{\text{def}}{=} \begin{cases} \text{if } f_t^{d+1} \notin G, f_t^d \notin G : [q_t^d = q_{t-1}^d] \\ \text{if } f_t^{d+1} \in G, f_t^d \notin G : \mathsf{P}_{\Theta_{\text{Q-Transition},d}}(q_t^d \mid f_t^{d+1} f_t^d \; q_{t-1}^d q_t^{d-1}) \\ \text{if } f_t^{d+1} \in G, f_t^d \in G : \mathsf{P}_{\Theta_{\text{Q-Expansion},d}}(q_t^d \mid q_t^{d-1}) \end{cases} \tag{9}$$

where $f_t^{D+1} = \mathbf{1}$ and $q_t^0 = \mathbf{ROOT}$. This model is conditioned on final-state intermediate variables f_t^d and f_t^{d+1} at and immediately below each HHMM level. If there is no reduction immediately below a given level (the first case provided), it deterministically copies the current HHMM state forward to the next time step. If there is a reduction immediately below the current level but no reduction at the current level (the second case provided), it transitions the HHMM state at the current level, according to the distribution $\Theta_{\text{Q-Transition},d}$. And if there is a reduction at the current level (the third case above), it re-initializes this state given the state at the level above, according to the distribution $\Theta_{\text{Q-Expansion},d}$.

Models $\Theta_{\text{F-Reduction},d}$, $\Theta_{\text{Q-Transition},d}$, and $\Theta_{\text{Q-Expansion},d}$ are defined directly from training examples, for example (in the experiments described in this article), using relative frequency estimation. The overall effect is that higher-level HMMs are allowed to transition only when lower-level HMMs terminate. An HHMM therefore behaves like a probabilistic implementation of a shift–reduce parser or pushdown automaton with a bounded stack, where the maximum stack depth is equal to the number of depth levels in the HHMM hierarchy.

4. Right-Corner Transform and Incomplete Constituents

The model described in this article recognizes trees in a **right-corner transformed** representation to minimize usage of a bounded short-term memory store. This right-corner transform is a variant of the left-corner transform described by Johnson (1998a), but whereas the left-corner transform changes left-branching structure into right-branching structure, the right-corner transform changes right-branching structure into

6 Here $[\cdot]$ is an indicator function: $[\phi] = 1$ if ϕ is true, 0 otherwise.

left-branching structure. Recognition using this transformed grammar, extracted from a transformed corpus, is similar to recognition using a left-corner parsing strategy (Aho and Ullman 1972). This kind of strategy was shown to reduce memory requirements for parsing sentences with mainly left- or right-recursive phrase structure to fewer than seven active or awaited constituent categories (Abney and Johnson 1991). This is within Miller's (1956) estimate of human short-term memory capacity (if memory elements store individual categories), whereas parsing heavily center-embedded sentences (known to be difficult for human readers) would require seven or more elements at the frontier of this capacity.

But recent research suggests that human memory capacity may be limited to as few as three or four distinct items (Cowan 2001), with longer estimates of seven or more possibly due to the human capacity to chunk remembered items into associated groups (Miller 1956). The right-corner strategy described in this paper therefore assumes constituent categories can similarly be chunked into *incomplete constituents* A/B formed by pairing an active category A with an awaited category B somewhere along the active category's right progeny (so, for example, a transitive verb may become an incomplete constituent VP/NP consisting of an active verb phrase lacking an awaited noun phrase yet to come).[7] These chunked incomplete constituent categories A and B are naturally related through fixed contiguous phrase structure between them, established during the course of parsing prior to the beginning of B, and these incomplete constituents can be composed with other incomplete constituents B/C to form similarly related category pairs A/C.

These chunks are not only contiguous sections of phrase structure trees, they also have contiguous string yields, so they correspond to the familiar notion of text chunks used in shallow parsing approaches (Hobbs et al. 1996). For example, a hypothesized memory store may contain incomplete constituents S/NP (a sentence without a noun phrase), followed by NP/NN (a noun phrase lacking a common noun), with corresponding string yields *demand for bonds propped up* and *the municipal*, respectively, forming a complete contiguous segmentation of a sentence at any point in processing. Although these two chunks could be composed into an incomplete constituent S/NN, doing so at this point would close off the possibility of introducing another constituent between these two, containing the recognized noun phrase as a left child (e.g., *demand for bonds propped up* $[_{NP}$ $[_{NP}$ *the municipal bonds]'s prices]*).

This conception of chunking applied to right-branching progeny in phrase structure trees does not have the power to eliminate the bounds of a memory store, however. In a larger cognitive model, syntactic processing is assumed to occur as part of an interactive semantic interpretation process, in which referents of constituents are calculated as these constituents are recognized, and are used to constrain subsequent processing decisions (Tanenhaus et al. 1995; Brown-Schmidt, Campana, and Tanenhaus 2002).[8] The chunked category pairs A and B in these incomplete constituents A/B result from successful compositions of other such constituents earlier in the recognition process, which means that the relationship between the referents of A and B is known and fixed

7 Incomplete constituents may also be defined through a left-corner transform, but left-corner transformed categories are incomplete in the other direction—a goal category yet to come lacking an already-recognized constituent—so stored incomplete constituent categories resulting from a left-corner transform would have the character of future goal events, rather than remembered past events. This is discussed in greater detail in Section 4.4.

8 This can be implemented in a time-series model by factoring the model to include additional random variables over referents, as described in Schuler, Wu, and Schwartz (2009).

in any hypothesized incomplete constituent. But syntactic and semantic relationships *between* chunks in a hypothesized memory store are unspecified. Chunking beyond the level of incomplete constituents would therefore involve grouping referents whose interrelations have not necessarily been established by the parser. Because the set of referents is presumably much larger than the set of syntactic categories, one may assume there are real barriers to reliably chunking them in the absence of these fixed relationships.

There certainly may be cases where syntactically unconnected referents (belonging to different incomplete constituents) could be grouped together as chunks. But for simplicity, this article will assume a very strict condition that only a single incomplete constituent can be stored in each short-term memory element. Experimental results described in Section 5 suggest that a vast majority of English sentences can be recognized within these human-like memory bounds, even with this strict condition on chunking. If parsing can be performed in bounded memory under such strict conditions, it can reasonably be assumed to operate at least as well under more permissive circumstances, where some amount of syntactically-unrelated referential chunking is allowed.

Several existing incremental systems are organized around a left-corner parsing strategy (Roark 2001; Henderson 2004). But these systems generally keep large numbers of constituents open for modifier attachment in each hypothesis. This allows modifiers to be attached as right children of any such open constituent. But if any number of open constituents are allowed, then either the assumption that stored elements have fixed syntactic (and semantic) internal structure will be violated, or the assumption that syntax operates within a bounded memory store will be violated, both of which are psycholinguistically attractive as simplifying assumptions. The HHMM model described in this article upholds both the fixed-element and bounded-memory assumptions by hypothesizing fixed reductions of right child constituents into incomplete parents in the same memory element, to make room for new constituents that may be introduced at a later time. These in-element reductions are defined naturally on phrase structure trees as the result of aligning right-corner transformed constituent structures to sequences of random variables in a factored time-series model. The success of this predictive strategy in corpus-based coverage and accuracy results described in Sections 5 and 6 suggests that it may be plausible as a cognitive model.

Other accounts may model reductions in bounded memory as occurring as soon as possible, by maintaining the option of undoing them when necessary (Stevenson 1998). This seems unattractive in the context of an interactive semantic model, however, where syntactic constituents and semantic referents are composed in tandem, because potentially very rich referential constraints introduced by composing a child constituent into a parent would have to be systematically undone. An interesting possibility might be that the appearance of syntactic restructuring may arise from a collection of hypothesized stores of syntactically fixed incomplete constituents, pursued in parallel. The results presented in this article suggest that this mechanism is possible, but these two possibilities might be very difficult to distinguish empirically.

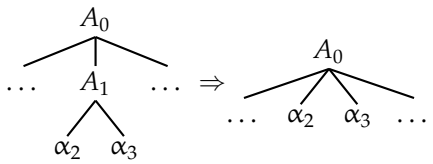
There is also a tradition of defining incomplete constituents as head-driven—introduced in parsing only at the point in incremental recognition at which they can be associated with a head word (Gibson 1991: Pritcher 1991: Gorrell 1995). In typically head-initial languages such as English, incomplete constituents derived from these head-driven models resemble those derived from a right-corner transform. But head-driven incomplete constituents do not appear to obey general-purpose memory bounds in head-final languages such as Japanese, and do not appear to obey attachment prefer-

ences predicted by a head-driven account (Kamide and Mitchell 1999), favoring a pre-head attachment account, as a right-corner transform would predict.

4.1 Tree Transforms Using Rewrite Rules

The incomplete constituents used in the present model are defined in terms of tree transforms, which consist of recursive operations that change tree structures into other tree structures. These transforms are not cognitive processes—syntax in this model is learned and used entirely as time-series probabilities over random variable values in the memory store. The role of these transforms is as a means to associate sequences of configurations of incomplete constituents in a memory store with linguistically familiar phrase structure representations, such as those studied in competence models or found in annotated corpora.

The transforms presented in this article will be defined in terms of *destructive rewrite rules* applied iteratively to each constituent of a source tree, from leaves to root, and from left to right among siblings, to derive a target tree. These rewrites are ordered; when multiple rewrite rules apply to the same constituent, the later rewrites are applied to the results of the earlier ones.[9] For example, the rewrite



could be used to iteratively eliminate all binary-branching nonterminal nodes in a tree, except the root.

In the notation used in this article,

- Roman uppercase letters (A_i) are variables matching constituent labels,
- Roman lowercase letters (a_i) are variables matching terminal symbols,
- Greek lowercase letters (α_i) are variables matching entire subtree structure,
- Roman letters followed by colons, followed by Greek letters (A_i:α_i) are variables matching the label and structure, respectively, of the same subtree, and
- ellipses (. . .) are taken to match zero or more subtree structures, preserving the order of ellipses in cases where there are more than one (as in the rewrite shown herein).

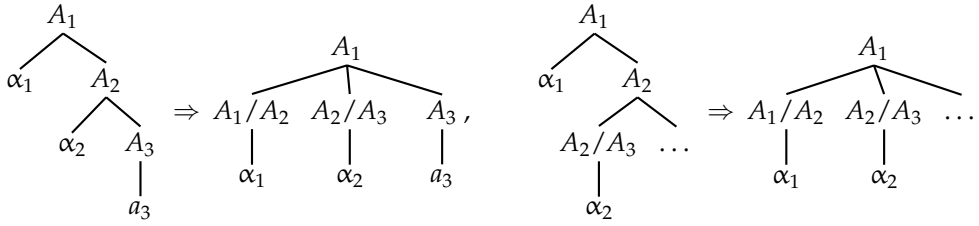
Many of the transforms used in this article are reversible, meaning that the result of applying a transform to a tree, then applying the reverse of that transform to the resulting tree, will be the original tree itself. In general, a transform can be reversed if the direction of its rewrite rules is reversed, and if each constituent in a target tree

9 The appropriate analogy here is to a Unix sed script, made sensitive to the beginning and end brackets of a constituent and those of its children.

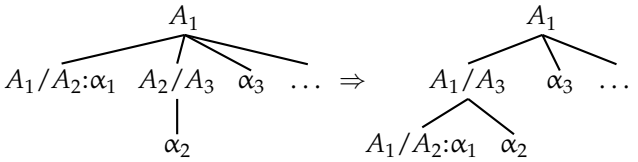
matches a unique rewrite rule in the reversed transform. The fact that not all rewrites can be unambiguously matched to HHMM output means that parse accuracy must be evaluated on partially-binarized gold-standard trees, in order to remove the effect of this ambiguous matching from the evaluation. This will be discussed in greater detail in Section 6.

4.2 Right-Corner Transform Using Rewrite Rules

Rewrite rules for the right-corner transform are shown here, first to flatten out right-recursive structure,



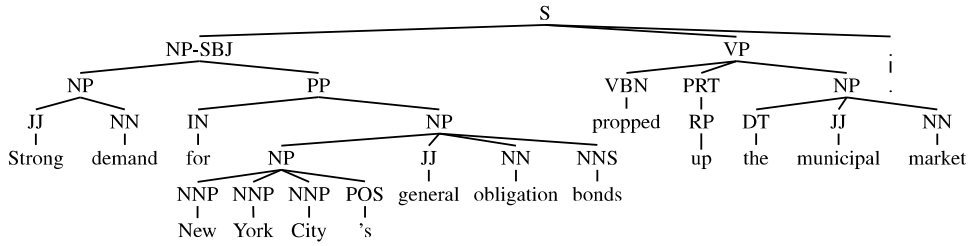
then to replace it with left-recursive structure,



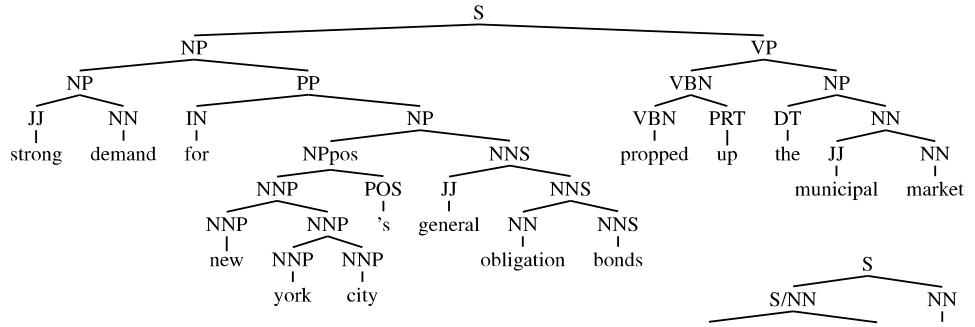
Here, the first two rewrite rules are applied iteratively (bottom-up on the tree) to flatten all right recursion, using incomplete constituents to record the original nonterminal ordering. The second rule is then applied to generate left-recursive structure, preserving this ordering. Note that the last rewrite leaves a unary branch at the leftmost child of each flattened node. This preserves the simple category labels of nodes that correspond directly to nodes in the original tree, so the original tree can be reconstructed when the right-corner transform concatenates multiple right-recursive sequences into a single left-recursive sequence.

An example of a right-corner transformed tree is shown in Figure 2(c). An important property of this transform is that it is reversible. Rewrite rules for reversing a right-corner transform are simply the converse of those shown here. The correctness of this can be demonstrated by dividing a tree into maximal sequences of right-recursive branches (that is, maximal sequences of adjacent right children). The first two "flattening" rewrites of the right-corner transform, applied to any such sequence, will replace the right-branching nonterminal nodes with a flat sequence of nodes labeled with slash categories, which preserves the order of the nonterminal category symbols in the original nodes. Reversing this rewrite will therefore generate the original sequence of nonterminal nodes. The final rewrite similarly preserves the order of these nonterminal symbols while grouping them from the left to the right, so reversing this rewrite will reproduce the flattened tree.

a) original Treebank phrase structure tree:



b) binarized phrase structure tree:



c) result of right-corner transform:

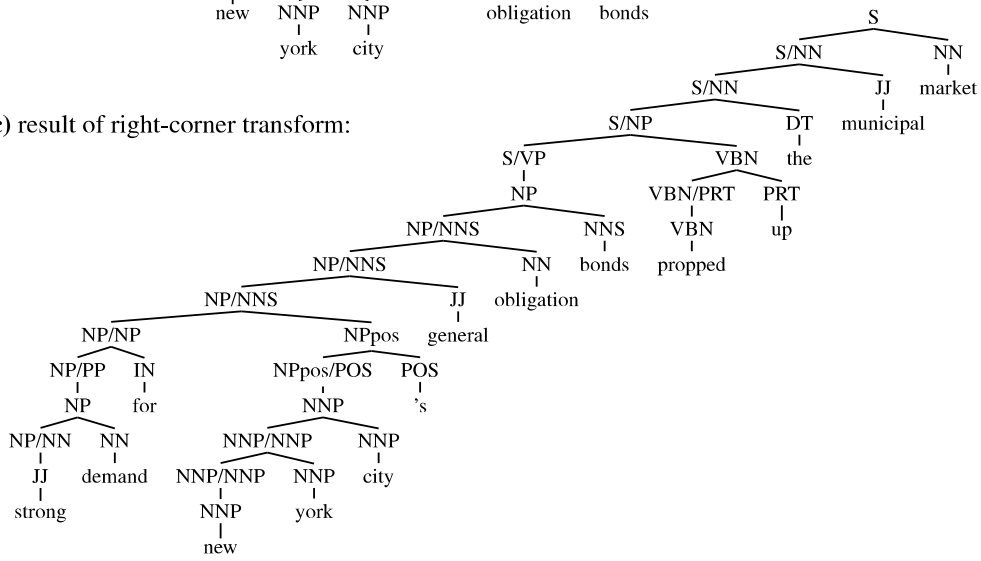


Figure 2
A sample phrase structure tree (a) as it appears in the Penn Treebank, (b) after it has been binarized, and (c) after it has been right-corner transformed.

4.3 Mapping Trees to HHMM Derivations

Any tree can be mapped to an HHMM derivation by aligning the nonterminals with q_t^d categories. First, it is necessary to define rightward depth d, right index position t, and final (rightmost) child status f_{t+1}^d, for every nonterminal node A in a tree, where

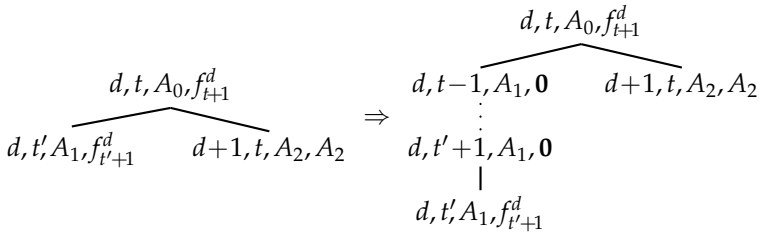
- d is defined to be the number of right branches between node A and the root,

- t is defined to be the number of words beneath or to the left of node A, and
- f^d_{t+1} is defined to be **0** if A is a left child, **1** if A is a unary child, and A if A is right.

Any right-corner transformed tree can then be annotated with these values and rewritten to define labels and final-state values for every combination of d and t covered by the tree. This is done using the rewrite rule

$$\begin{array}{c} d,t,A_0,\mathbf{0} \\ | \\ d,t,A_1,\mathbf{1} \end{array} \Rightarrow\ d,t,A_1,\mathbf{1}$$

to replace unary branches with f^d_{t+1} flags, and



to copy stacked-up left child constituents over multiple time steps, while lower-level (right child) constituents are being recognized. The dashed line on the right side of the rewrite rule represents the variable number of time steps for a stacked-up higher-level constituent (as seen, for example, in time steps 4–7 at depth 1 in Figure 3). Coordinates $d,t \leq D$, and T that have $f^d_{t+1}=\mathbf{1}$ are assigned label '$-$', and coordinates not covered by the tree are assigned label '$-$' and $f^d_{t+1}=\mathbf{1}$.

The resulting label and final-state values at each node now define a value of q^d_t and f^d_t for each depth d and time step t of the HHMM (see Figure 3). Probabilities for HHMM models $\Theta_{\text{Q-Expansion},d}$, $\Theta_{\text{Q-Transition},d}$, and $\Theta_{\text{F-Reduction},d}$ can then be estimated from these values directly. Like the right-corner transform, this mapping is reversible, so q^d_t and f^d_t values can be taken from a hypothesized most likely sequence and mapped back

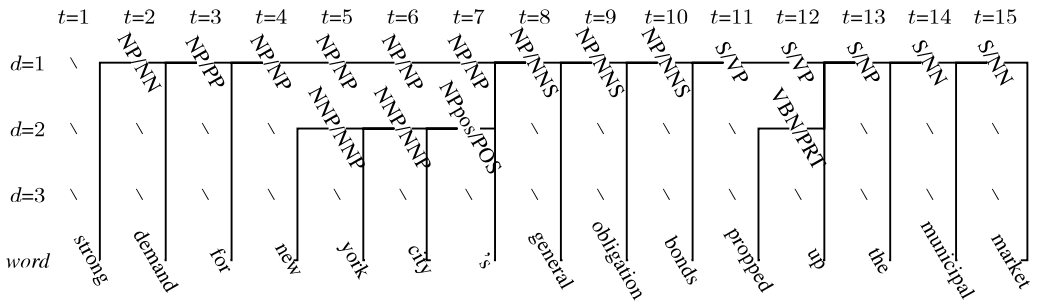


Figure 3
Sample tree from Figure 2 mapped to q^d_t variable positions of an HHMM at each stack depth d (vertical) and time step t (horizontal). This tree uses only two levels of stack memory. Values for final-state variables f^d_t are not shown. Note that the mapping transform omits some nonterminal labels; labels for these nodes can be reconstructed from their children.

to trees (which can then undergo the reverse of the right-corner transform to become ordinary phrase structure trees). Inspection of this rewrite rule will reveal the reverse of this transform simply involves deleting unary-branching sequences that differ only in the value of t and restoring unary branches when $f^d_{t+1} = \mathbf{1}$.

This alignment of right-corner transformed trees also has the interesting property that the categories on the stack at any given time step represent a segmentation of the input up to that time step. For example, in Figure 3 at t = 12 the stack contains a sentence lacking a verb phrase: S/VP (*strong demand for . . . bonds*), followed by a verb projection lacking a particle: VBN/PRT (*propped*).

4.4 Comparison with Left-Corner Transform

A right-corner transform is used in this study, rather than a left-corner transform, mainly because the right-corner version coincides with an intuition about how incomplete constituents might be stored in human memory. Stacked-up constituents in the right-corner form correspond to chunks of words that have been encountered, rather than hypothesized goal constituents. Intuitively, in the right-corner view, after a sentence has been recognized, the stack memory contains a complete sentential constituent (and some associated referent). In the left corner view, on the other hand, the stack memory after a sentence has been recognized contains only the lower-rightmost constituent in the corresponding phrase structure tree (see Figure 4). This is because a time-order

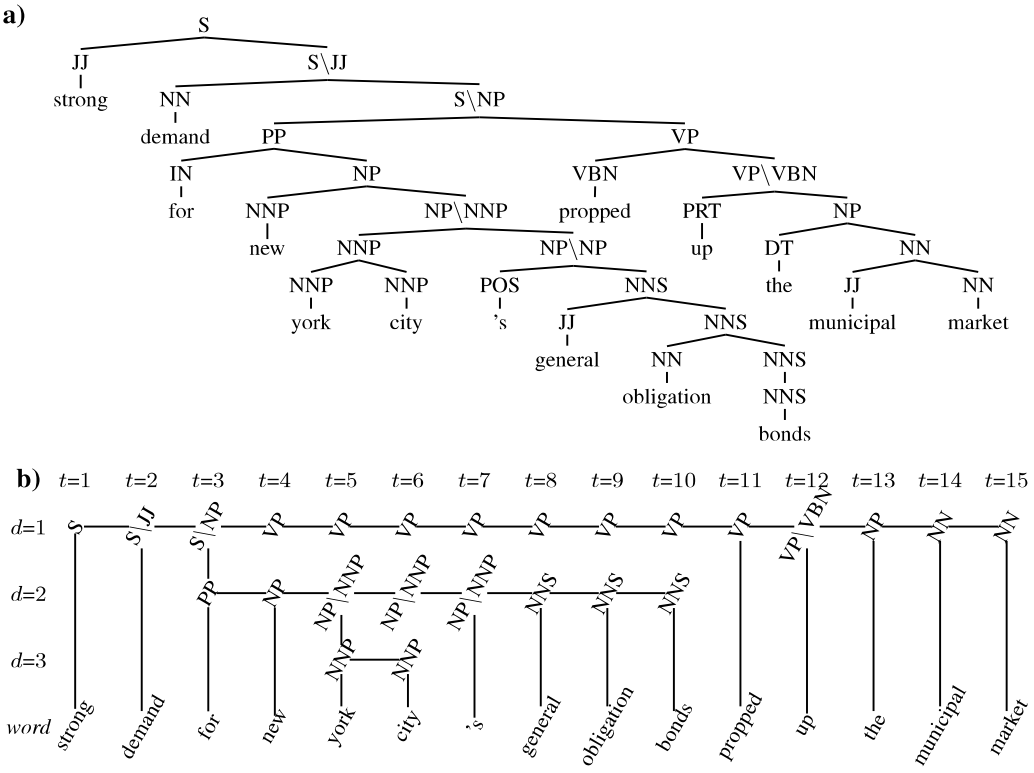


Figure 4
A left-corner transformed version of the tree (a) and memory store (b) from Figures 2 and 3.

alignment of a left-corner tree to elements in a bounded memory store corresponds to a top-down traversal of the tree, whereas a time-order alignment of a right-corner tree to elements in a bounded memory store corresponds to a bottom-up traversal of the tree. If referential semantics are assumed to be calculated in tandem (as suggested by the Tanenhaus et al. [1995] results), a top-down traversal through time requires some effort to reconcile with the traditional compositional semantic notion that the meanings of constituents are composed from the meanings of their parts (Frege 1892).

4.5 Comparison with CCG

The incomplete constituent categories generated in the right-corner transform have the same form and much of the same meaning as non-constituent categories in a CCG (Steedman 2000).[10] Both CCG operations of forward function application:

$$A_1 \rightarrow A_1/A_2 \;\; A_2$$

and forward function composition:

$$A_1/A_3 \rightarrow A_1/A_2 \;\; A_2/A_3$$

appear in the branching structure of right-corner transformed trees. Nested operations can also occur in CCG derivations:

$$A_1/A_2 \rightarrow A_1/A_2/A_3 \;\; A_3$$

as well as in right-corner transformed trees (using underscore delimiters to denote sequences of constituent categories, described in Section 5.1):

$$A_1/A_2 \rightarrow A_1/A_3\_A_2 \;\; A_3$$

There are also correlates of type-raising (unary branches introduced by the right-corner transform operations described in Section 4):

$$A_1/A_2 \rightarrow A_3$$

But, importantly, the right-corner transform generates no correlates to the CCG operations of backward function application or composition:

$$A_1 \rightarrow A_2 \;\; A_1\backslash A_2$$
$$A_1\backslash A_3 \rightarrow A_2\backslash A_3 \;\; A_1\backslash A_2$$

This has two consequences. First, right-corner transform models do not introduce ambiguity between type-raised forward and backward operations, as CCG derivations do. Second, because leftward dependencies (as between a verb and its subject in English) cannot be incorporated into lexical categories, right-corner transform models cannot be taken to explicitly encode argument structure, as CCGs are. The right-corner transform model described in this article is therefore perhaps better regarded as a performance model of processing, given subcategorizations specified in some other grammar (such as in this case the Treebank grammar), rather than a constraint on grammar itself.

10 In fact, one of the original motivations for CCG as a model of language was to minimize stack usage in incremental processing (Ades and Steedman 1982).

4.6 Comparison with Cascaded FSAs in Information Extraction

Hierarchies of weighted finite-state automata (FSA)–equivalent HMMs may also be viewed as probabilistic implementations of cascaded FSAs, used for modeling syntax in information extraction systems such as FASTUS (Hobbs et al. 1996). Indeed, the left-branching sequences of transformed constituents recognized by this model (as shown in Figure 3) bear a strong resemblance to the flattened phrase structure representations recognized by cascaded FSA systems, in that most phrases are consolidated to flat sequences at one hierarchy level. This flat structure is desirable in cascaded FSA systems because it allows information to be extracted from noun or verb phrases using straightforward pattern matching rules, implemented as FSA-equivalent regular expressions.

Like FASTUS, this system produces layers of flat phrases that can be searched using regular expression pattern-matching rules. It also has a fixed number of levels and linear-time recognition complexity. But unlike FASTUS, the model described here can produce—and can be trained on—complete phrase structure trees (accessible by reversing the transforms described previously).

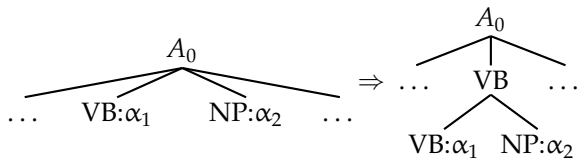
5. Coverage

The coverage of this model was evaluated on the large Penn Treebank corpus of syntactically annotated sentences from the Switchboard corpus of transcribed speech (Godfrey, Holliman, and McDaniel 1992) and the *Wall Street Journal* (Marcus, Santorini, and Marcinkiewicz 1993). These sentences were right-corner transformed and mapped to a time-aligned bounded memory store as described in Section 4 to determine the amount of memory each sentence would require.

5.1 Binary Branching Structure

In order to obtain a linguistically plausible right-corner transform representation of incomplete constituents, the corpus is subjected to another pre-process transform to introduce binary-branching nonterminal projections, and fold empty categories into nonterminal symbols in a manner similar to that proposed by Johnson (1998b) and Klein and Manning (2003). This binarization is done in such a way as to preserve linguistic intuitions of head projection, so that the depth requirements of right-corner transformed trees will be reasonable approximations to the working memory requirements of a human reader or listener.

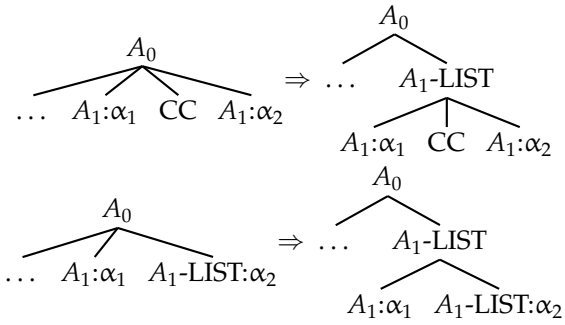
First, ordinary phrases and clauses are decomposed into **head projections**, each consisting of one subordinate head projection and one argument or modifier, for example:



The selection of head constituents is done using rewrite rules similar to the Magerman-Black head rules (Magerman 1995). Any new constituent created by this process is

assigned the label of the subordinate head projection. The subordinate projection may be the left or complete list of head-projection rewrite rules is provided in Appendix A.[11]

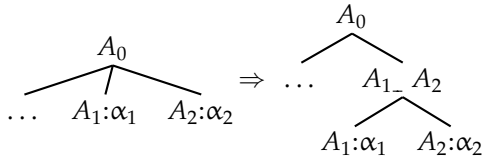
Conjunctions are decomposed into purely right-branching structures using nonterminals appended with a "-LIST" suffix:



This right-branching decomposition of conjoined lists is motivated by the general preference in English toward right branching structure, and the distinction of right children as "-LIST" categories is motivated by the asymmetry of conjunctions such as *and* and *or* generally occurring only between constituents at the end of a list, not at the beginning. (Thus, in decomposing *coffee, tea or milk,* the words *tea or milk* form an NP-LIST constituent, whereas the words *coffee, tea* do not.)

Empty constituents are removed outright, along with any unary projections that may arise from this removal. In the case of empty constituents representing traces, the extracted category label is annotated onto the lowest nonterminal dominating the trace using the suffix "-extr$X$," where "X" is the category of the extracted constituent. To preserve grammaticality, this annotation is then passed up the tree and eliminated when a *wh-*, topicalized, or other moved constituent is encountered, in a manner similar to that used in Head-driven Phrase Structure Grammar (Pollard and Sag 1994), but this does not affect branching structure.

Together these rewrites remove about 65% of super-binary branches from the unprocessed Treebank. All remaining super-binary branches are "nominally" decomposed into right-branching structures by introducing intermediate nodes, each with a label concatenated from the labels of its children, delimited by underscores:



This decomposition is "nominal" in that the concatenated labels leave the resulting binary branches just as complex as the original n-ary branches prior to this decomposition. It is equivalent to leaving super-binary branches intact and using dot rules in parsing

11 Although it is possible that in some cases these rules may generate counterintuitive branching patterns, inspection of transformed trees during this experiment showed no unusual branching structure, except in the case of noun sequences in base noun phrases (e.g. *[general obligation] bonds* or *general [obligation bonds]*), which were left flat in the Treebank. Correct binarization of these structures would require extensive annotator effort. However, because base noun phrases are often very small, and seldom contain any sub-structure, it seems safe to assume that structural errors in these base noun phrases would not drastically alter coverage results reported in this section.

(Earley 1970). This decomposition therefore does nothing to reduce sparse data effects in statistical parsing.

5.2 Coverage Results

Sections 2 and 3 (the standard training set) of the Penn Treebank Switchboard corpus were binarized as described in Section 5.1, then right-corner transformed and mapped to elements in a bounded memory store as described in Section 4. Punctuation added by transcribers was removed. Coverage of this corpus, in sentences, for a recognizer using right-corner transform chunking with one to five levels of stack memory, is shown in Table 1. These results show that a simple syntax-based chunking into incomplete constituents, using the right-corner transform defined in Section 4 of this article, allows a vast majority of Switchboard sentences (over 99%) to be recognized using three or fewer elements of memory, with no sentences requiring more than five elements, essentially as predicted by studies of human short-term memory.

Although spontaneous speech is arguably more natural test data than prepared speech or edited text, it is possible that coverage results on these data may underestimate processing requirements, due to the preponderance of very short sentences and sentence fragments in spontaneous speech (for example, nearly 30% of sentences in the Switchboard corpus are only one word long). It may also be argued that coverage results on this corpus more accurately reflect the complexity of speech planning under somewhat awkward social circumstances (being asked to start a conversation with a stranger), which may be more cognitively demanding than recognition. For these reasons, the right-corner transform chunking was also evaluated on Sections 2–21 (the standard training set) of the Penn Treebank *Wall Street Journal* (WSJ) text corpus (see Table 2, column 1).

The WSJ text corpus results appear to show substantially higher memory requirements than Switchboard, with only 93% of sentences recognizable using three or fewer memory elements. But much of this increase is due to arguably arbitrary treebank conventions in annotating punctuation (for example, commas between phrases are attached to the leftmost phrase: (*[Pierre Vinken . . . [61 years old] ,] joined . . .*) which can lead to psycholinguistically implausible analyses in which phrases (in this case *61 years old*) are center-embedded by lone punctuation marks on one side or the other. In general, branching structure for punctuation can be difficult to motivate on linguistic grounds, because punctuation marks do not have lexical projections or argument structure in most linguistic theories. In spoken language, punctuation corresponds to

Table 1
Percent coverage of right-corner transformed Switchboard Treebank Sections 2–3.

memory capacity (right-corner, no punct)	sentences	coverage
no stack memory	26,201	28.38%
1 stack element	53,740	58.21%
2 stack elements	85,068	92.14%
3 stack elements	91,890	99.53%
4 stack elements	92,315	99.99%
5 stack elements	92,328	100.00%
TOTAL	92,328	100.00%

Table 2
Percent coverage of left- and right-corner transformed WSJ Treebank Sections 2–21.

memory capacity	right-corner, with punct		right-corner, no punct		left-corner, no punct	
	sentences	coverage	sentences	coverage	sentences	coverage
no stack elements	35	0.09%	127	0.32%	127	0.32%
1 stack elements	3,021	7.57%	3,550	8.90%	4,284	10.74%
2 stack elements	21,916	54.95%	25,948	65.06%	26,750	67.07%
3 stack elements	37,203	93.28%	38,948	97.66%	38,853	97.42%
4 stack elements	39,703	99.54%	39,866	99.96%	39,854	99.93%
5 stack elements	39,873	99.97%	39,883	100.00%	39,883	100.00%
6 stack elements	39,883	100.00%	-	-	-	-
TOTAL	39,883	100.00%	39,883	100.00%	39,883	100.00%

pauses or patterns of inflection, distributed throughout an utterance. It therefore seems questionable to account for punctuation marks in a psycholinguistic model as explicit composable concepts in a memory store. In order to counter possible undesirable effects of an arbitrary branching analysis of punctuation, a second evaluation of the model was performed on a version of the WSJ corpus with punctuation removed.

An analysis (Table 2, column 2) of the Penn Treebank WSJ corpus Sections 2–21 without punctuation, using the right-corner transformed trees just described, shows that 97.66% of trees can be recognized using three hidden levels, and 99.96% can be recognized using four, and again (similar to the Switchboard results), no sentences require more than five remembered incomplete constituents. Table 2, column 3, shows similar results for a left-corner transformed corpus, using left-right reflections of the rewrite rules presented in Section 4.

Cowan (2001) documents empirically observed short-term memory limits of about four elements across a wide variety of tasks. It is therefore not surprising to find a similar limit in the memory required to parse the Treebank, assuming elements corresponding to right-corner-transformed incomplete constituents.

As the table shows, some quintuply center-embedded constituents were found in both corpora, suggesting that a three- to four-element limit may be soft, and can be relaxed for short durations. Indeed, all quintuply embedded constituents were only a few words long. Interestingly, many of the most heavily embedded words seemed to strongly co-occur, which may suggest that these words arise from fixed expressions and are not compositional. For example, Figure 5 shows one of the 13 phrase structure trees in the Switchboard corpus which require five stack elements in right-corner parsing. The complete sentence is:

So if there's no one else around and I have a chance to listen to something I'll turn that on.

If the construction *there 's NP AP* in this sentence is parsed non-compositionally as a single expression (and thus is rendered left-branching by the right-corner transform as defined in Section 4), the sentence could be parsed using only four memory elements.

Even constrained to only four center embeddings, the existence of such sentences confounds explanations of the center-embedding difficulties as directly arising from stack limits in a left-corner (or right-corner) parser (Abney and Johnson 1991). It is also interesting to note that three of the incomplete constituents in this example are recursively nested or **self-embedded** instances of sentential projections, essentially with

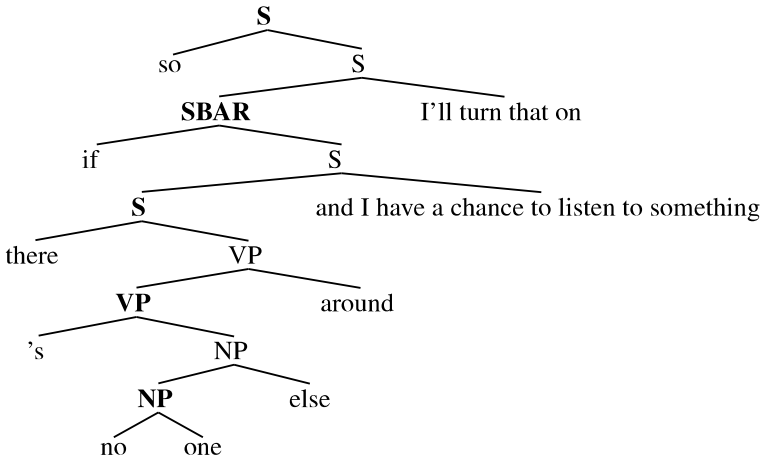


Figure 5
A phrase structure tree requiring five stack elements. Categories in bold will be incomplete at a point after recognizing *so if there's no . . .*

the same category, similar to the center-embedded constructions which human readers found difficult to process. This suggests that restrictions on self-embedding of identical constituent categories would also fail to predict readability.

Instead, these data seem to argue in favor of an explanation due to probability: Although the five-element sentences found in the Treebank use mostly common phrase structure rules, problematic center-embedded sentences like *the salmon the man the dog chased smoked fell* may cause difficulty simply because they are examples of an unusual construction: a nested object relative clause. The fact that this is an unusual construction may in turn be a result of the fact that speakers tend to avoid nesting object relative clauses because they can lead to memory exhaustion, though such constructions may become readable with practice.

6. In-Element Composition Ambiguity and Parsing Accuracy

The right-corner transform described in Section 4 saves memory because it transforms any right-branching sequence with left-child subtrees into a left-branching sequence of incomplete constituents, with the same sequence of subtrees as right children. The left-branching sequences of siblings resulting from this transform can then be composed bottom-up through time by replacing each left child category with the category of the resulting parent, within the same memory element (or depth level). For example, in Figure 6(a) a left-child category NP/NP at time $t = 4$ is composed with a noun *new* of category NP/NNP (a noun phrase lacking a proper noun yet to come), resulting in a new parent category NP/NNP at time $t = 5$ replacing the left child category NP/NP in the topmost $d = 1$ memory element.

This in-element composition preserves elements of the bounded memory store for use in processing descendants of this composed constituent, yielding the human-like memory demands reported in Section 5. But whenever an in-element composition like this is hypothesized, it isolates an intermediate constituent (in this example, the noun phrase *new york city*) from subsequent composition. Allowing access to this intermediate constituent—for example, to allow *new york city* to become a modifier of *bonds*, which itself becomes an argument of *for*—requires an analysis in which the intermediate

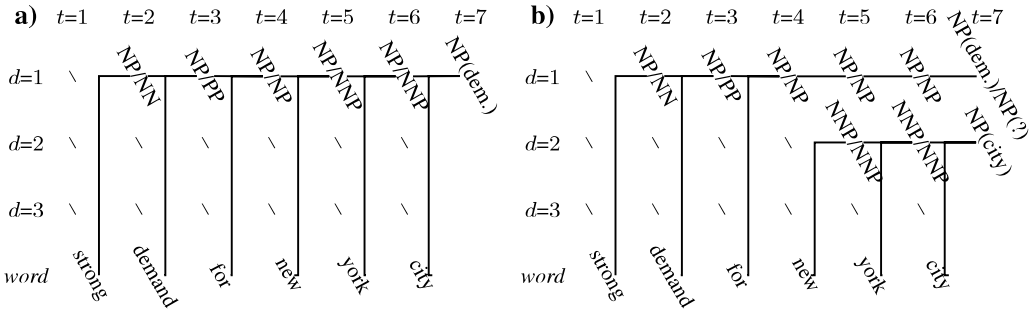


Figure 6
Alternative analyses of *strong demand for new york city ...*: (a) using in-element composition, compatible with *strong demand for new york city is ...* (in which the demand is for the city); and (b) using cross-element (or delayed) composition, compatible with either *strong demand for new york city is ...* (in which the demand is for the city) or *strong demand for new york city bonds is ...* (in which a forthcoming referent—in this case, bonds—is associated with the city, and is in demand). In-element composition (a) saves memory but closes off access to the noun phrase headed by *city*, and so is not incompatible with the *...bonds* completion. Cross-element composition (b) requires more memory, but allows access to the noun phrase headed by *city*, so is compatible with either completion. This ambiguity is introduced at $t = 4$ and propagated until at least $t = 7$. An ordinary, non-right-corner stack machine would exclusively use analysis (b), avoiding ambiguity.

constituent is stored in a separate memory element, shown in Figure 6(b). This creates a local ambiguity in the parser (in this case, from time step $t = 4$) that may have to be propagated across several words before it can be resolved (in this case, at time step $t = 7$). This is essentially an ambiguity between arc-eager (in-element) and arc-standard (cross-element) composition strategies, as described by Abney and Johnson (1991). In contrast, an ordinary (purely arc-standard) parser with an unbounded stack would only hypothesize analysis (b), avoiding this ambiguity.[12]

The right-corner HHMM approach described in this article relies on a learned statistical model to predict when in-element (arc-eager) compositions will occur, in addition to hypothesizing parse trees. The model encodes a mixed strategy: with some probability arc-eager or arc-standard for each possible expansion. Accuracy results on a right-corner HHMM model trained on the Penn *Wall Street Journal* Treebank suggest that this kind of optionally arc-eager strategy can be reliably statistically learned.

6.1 Evaluation

In order to determine whether a memory-preserving parsing strategy, like the optionally arc-eager strategy, can be reliably learned, a baseline Cocke-Kasami-Younger (CKY) parser and bounded-memory right-corner HHMM parser were evaluated on the standard Penn Treebank WSJ Section 23 parsing task, using the binarized tree set described in Section 5.2 (WSJ Sections 2–21) as training data. Training examples requiring more

12 It is important to note that neither the right-corner nor left-corner parsing strategy by itself creates this ambiguity. The ambiguity arises from the decision to use this optionally arc-eager strategy to reduce memory store allocation in a bounded memory parser. Implementations of left-corner parsers such as that of Henderson (2004) adopt an arc-standard strategy, essentially always choosing analysis (b), and thus do not introduce this kind of local ambiguity. But in adopting this strategy, such parsers must maintain a stack memory of unbounded size, and thus are not attractive as models of human parsing in short-term memory (Resnik 1992).

than four stack elements were excluded from training, in order to avoid generating inconsistent model probabilities (e.g., from expansions that could not be re-composed within the bounded memory store).

Most likely sequences of HHMM stack configurations are evaluated by reversing the binarization, right-corner, and time-series mapping transforms described in Sections 4 and 5. But some of the binarization rewrites cannot be completely reversed, because they cannot be unambiguously matched to output trees. Automatically derived lexical projections below the annotated phrase level (e.g., binarizations of base noun phrases) can be completely reversed, because the derived categories are characteristically labeled with terminal symbols. So, too, can the conjunction and "nominal" binarizations described in Section 5.1, because they can be identified by characteristic "-LIST" and underscore delimiters. But automatically derived projections above the annotated phrase level cannot be reliably identified in parser output (for example, an intermediate projection "S → PP S" may or may not be annotated in the corpus). In order to isolate the evaluation from the effects of these ambiguous matchings, the evaluation was performed using trees in a partially binarized format, obtained by reversing only those rewrites that result in unambiguous matches. Evaluating on this partially binarized data does not seem to unfairly increase parsing performance compared to other published results—quite the contrary: an evaluation using the state-of-the-art Charniak (2000) parser scores about half a point *worse* on labeled F-score (89.3% vs. 89.9%) when its hypotheses and gold standard trees are converted into this format.[13]

Both CKY baseline and HHMM test systems were run with a simple part of speech (POS) model using relative frequency estimates from the training set, backed off to a discriminative (decision tree) model conditioned on the last five letters of each word, normalized over unigram POS probabilities. The CKY baseline and HHMM results were obtained by training and evaluating on binarized trees, which is a necessary condition for the right-corner transform. The CKY baseline results appear to be better than those for a baseline probabilistic context-free grammar (PCFG) system reported by Klein and Manning (2003) using no modifications to the corpus, and no parent or sibling conditioning (see Table 3, top) because the binarization process allows the parser to avoid some sparse data effects due to large flat branching structures in the Treebank, resulting in improved parsing accuracy. Klein and Manning note that applying linguistically motivated binarization transforms can yield substantial improvements in accuracy—as much as nine points, in their study (in comparison, binarization only seems to improve accuracy by about seven points above an unmodified baseline in the present study). But the Klein and Manning results for binarization are provided only for models already augmented with Markov dependencies (that is, conditioning on parent and sibling categories, analogous to HHMM dependencies), so it was not possible to compare to a binarized and un-Markovized benchmark.

The results for HHMM parsing, training, and evaluating on these same binarized trees (modulo right-corner and variable-mapping transforms) were substantially better than binarized CKY, most likely due to the expanded HHMM dependencies on previous (q^d_{t-1}) and parent (q^{d-1}_t) variables at each q^d_t. For example, binarized PCFG probabilities may be defined in terms of three category symbols A, B, and C: $\mathsf{P}(A \rightarrow B\ C \mid A)$; whereas some of the HHMM probabilities are defined in terms of five category

13 This is presumably because the probability that a human annotator will annotate phrase structure brackets at a particular projection or not is something existing parsers learn and exploit to improve their accuracy. But it is not clear that this distinction is linguistically motivated.

Table 3
Labeled recall (LR), labeled precision (LP), weighted average (F-score), and parse failure (% of sentences yielding no tree output) results for basic CKY parser and HHMM parser on unmodified and binarized WSJ Sections 22 (sentences 1–393: "devset") and 23–24 (all sentences). Results are shown with and without punctuation, compared to Klein and Manning 2003 (KM'03) using baseline and parent+sibling (par+sib) conditioning, and Roark 2001 (R'01) using parent+sibling conditioning. Baseline CKY and test (parent+sibling) cases for the HHMM system start out at a higher accuracy than for the Klein-Manning system because the HHMM system requires binarization of trees, which removes some data sparsity in the raw Treebank annotation, whereas the Klein-Manning results are computed prior to binarization. Because it is incremental, the parser occasionally eliminates all continuable analyses from the beam, and therefore fails to find a parse. HHMM parse failures are accounted as zeros in the recall statistics, but are also listed separately, because in principle it might be possible to recover useful syntactic structure from partial sequences.

with punctuation: ($\leq$ 40 wds)	LR	LP	F-score	sentence failure	error reduction
KM'03: unmodified, devset	–	–	72.6	0	
KM'03: par+sib, devset	–	–	77.4	0	17.5%
CKY: binarized, devset	80.3	79.9	80.1	0.8	
HHMM: par+sib, devset	84.1	83.5	83.8	0.5	18.6%
CKY: binarized, sect 23	78.8	79.4	79.1	0.1	
HHMM: par+sib, sect 23	83.4	83.7	83.5	0.1	21.1%
no punctuation: ($\leq$ 120 wds)	LR	LP	F	fail	
R'01: par+sib, sect 23–24	75.2	77.4	–	0.1	
HHMM: par+sib, sect 23–24	77.2	78.3	77.7	0.0	

labels: $\mathsf{P}(A/B \mid C/D, E)$ (transitioning from incomplete constituent C/D to incomplete constituent A/B in the context of an expanding category E). This increases the number of free parameters (estimated conditional probabilities) in the model,[14] but apparently not to the point of sparsity; this is similar to the effect of horizontal Markovization (conditioning on the sibling category immediately previous to an expanded category) and vertical Markovization (conditioning on the parent of an expanded category) commonly used in PCFG parsing models (Collins 1999).

The improvement due to HHMM parsing over the PCFG baseline (18.6% reduction in error) is comparable to that reported by Klein and Manning for parent and sibling dependencies (first-order vertical and horizontal Markovization) over a baseline PCFG without binarization (17.5% reduction in error). However, because it is not possible to run the HHMM parser without binarization, and because Klein and Manning do not report results for binarization transforms in the absence of parent and sibling Markovization, it is potentially misleading to compare the results directly. For example, it is possible that the binarization transforms described here may have performance-optimizing effects that are latent in the binarized PCFG, but are brought out in HHMM parsing.

Results on Section 23 of this corpus show close to 84% recall and precision, comparable to that reported for state-of-the-art cubic-time parsers (with no constant bounds

14 Without punctuation, the HHMM model has 50,429 free parameters (including both Q and F models), whereas the binarized PCFG has 12,373.

on processing storage) using similar configurations of conditioning information, that is, without lexicalization or smoothing.

Roark (2001) describes a similar incremental parser based on left-corner transformed grammars, and also reports results for parsing with and without parent and sibling Markovization. Again the performance is comparable under similar conditions (Table 3, bottom).

This system was run with a beam width of 2,000 hypotheses. This beam width was selected in order to compare the performance of the bounded-memory model, which predicts in-element or cross-element composition, with that of conventional broad-coverage parsers, which also maintain large beams. With better modeling and vastly more data from which to learn, it is possible that the human processor may need to maintain far fewer alternative analyses, or perhaps only one, conditioned on a lookahead window of observations (Henderson 2004).[15]

These experiments used a maximum stack depth of four, and conditioned expansion and transition probabilities for each q_t^d on only the portion of the parent category following the slash (that is, only A_2 of A_1/A_2), in order to avoid sparse data effects. Examples requiring more than four stack elements were excluded from training. This is because in the basic relative frequency estimation used here, training examples are depth-specific. Because the (unpunctuated) training set contains only about a dozen sentences requiring more than four depth levels, each occupying that level for only a few words, the data on which the fifth level of this model would be trained are very sparse. Models at greater stack depths, and models depending on complete parent categories (or grandparent categories, etc., as in state-of-the-art parsers) could be developed using smoothing and backoff techniques or feature-based log-linear models, but this is left for later work (see Section 7).

7. Conclusion

This article has described a model of human syntactic processing that recognizes complete phrase structure trees using only a small store of memory elements of limited complexity. Sequences of hypothesized contents of this memory store can be mapped to and from conventional phrase structure trees using a reversible right-corner transform. If this syntactic processing model is combined with a bounded-memory interpreter (Schuler, Wu, and Schwartz 2009), however, allowing the contents of this store to be incrementally interpreted within the same bounded memory, it stands to reason that complete, explicit phrase structure trees would not need to be constructed at any time in processing, in keeping with experimental results showing similar lack of retention of words and syntactic structure during human processing (Sachs 1967; Jarvella 1971).

Initial results show the use of a memory store consisting of only three to four memory elements within this framework provides nearly complete coverage of the Penn Treebank Switchboard and WSJ corpora, consistent with recent estimates of general-purpose short-term memory capacity. This suggests that, unlike some earlier models, the hypothesis that human sentence processing uses general-purpose short-term

15 Although, if most competing analyses are unconscious, they would be difficult to detect. Formally, the competing pockets of activation hypothesized in a parallel-processing version of this model could be arbitrarily small and numerous, but it seems unlikely that very small pockets of activation would persist for very long (just as low probability analyses would be unlikely to remain on the HHMM beam). This possibility is discussed in the particle filter account of Levy (2008).

memory to store incomplete constituents, as defined by a right-corner transform, does not seem to substantially underestimate human processing capacity. Moreover, despite additional predictions that must take place within this model to manage parsing in such close quarters, preliminary accuracy results for an unlexicalized, un-smoothed version of this model, using only a four-element memory store, show close to 84% recall and precision on the standard parsing evaluation. This result is comparable to that reported for state-of-the-art cubic-time parsers (with no constant bounds on processing storage) using similar configurations of conditioning information, namely, without lexicalization or smoothing.

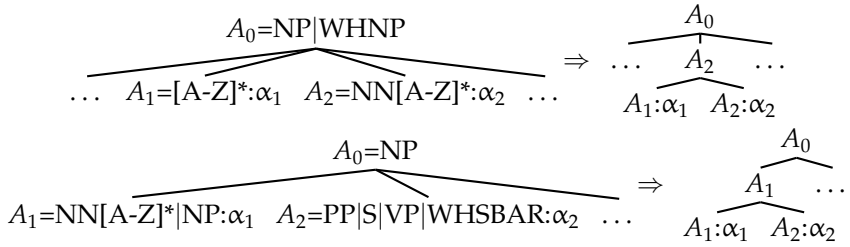
This model does not attempt to derive processing difficulties from memory bounds, following evidence that garden path and center-embedding processing difficulties are caused by interference or local probability estimates rather than encounters with memory capacity limits. But this does not mean that memory store capacity and probabilistic explanations of processing difficulty are completely independent. Probability estimation seems likely to be dependent on structural information from the memory store (for example, incomplete object relative clauses seem to be very improbable in the context of other incomplete object relative clauses). As hypotheses use more elements in the memory store, the distribution over these hypotheses will tend to become broader, taxing the reservoir of activation capacity, and making it more likely for low probability hypotheses to disappear, increasing the incidence of garden path errors. Further investigations into how the memory store elements are allocated in various syntactic contexts may allow these apparently disparate dimensions of processing capacity to be unified.

The model described here may be promising as an engineering tool as well. But to achieve competitive performance with unconstrained state-of-the-art parsers will require the development of additional approximation algorithms beyond the scope of this article. This is because most modern parsers are lexicalized, incorporating head-word dependencies into parsing decisions, and employing finely tuned smoothing and backoff techniques to integrate these potentially sparse head-word dependencies with denser unlexicalized models. The bounded-memory right-corner HHMM described in this article can also be lexicalized in this way, but because head word dependencies are most straightforwardly defined in terms of top-down PCFG-like dependency structures, this lexicalization requires the introduction of additional formal machinery to transform PCFG probabilities into right-corner form (Schuler 2009). In other words, rather than transforming a training set of trees and mapping them to a time series model, it is necessary to transform a consistent probabilistically weighted grammar (in some sense, an infinite set of trees) into appropriately weighted and consistent right-corner PCFG and HHMM models. This requires the introduction of an approximate inference algorithm, similar to that used in value iteration (Bellman 1957), which estimates probabilities of infinite left-recursive or right-recursive chains by exploiting the fact that increasingly longer chains of events contribute exponentially decreasing probability mass. On top of this, preserving head-word dependencies in incremental processing also requires the introduction of a framework for storing head words of modifier constituents that precede the head word of a parent constituent; including some mechanism to ensure that probability assignments are fairly distributed among competing hypotheses (e.g., by marginalizing over possible head words) in cases where the calculation of accurate dependency probabilities must be deferred until the head word of the parent constituent is encountered. For these reasons, a complete lexicalized model is considered beyond the scope of this article, and is left for future work.

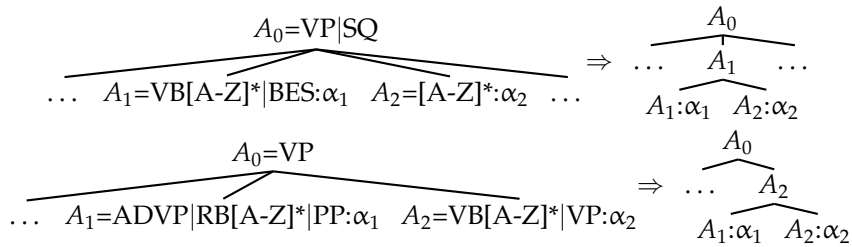
Appendix A: Head Transform Rules

The experiments described in this article used a binarization process that included the following rewrite rules, designed to binarize flat Treebank constituents into linguistically motivated head projections:

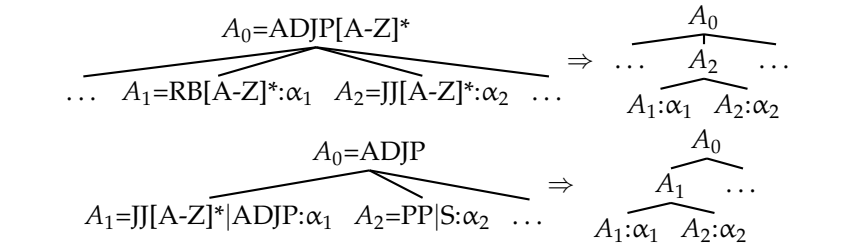
1. NP: right-binarize basal NPs as much as possible; then left-binarize NPs after left context reduced to nil:



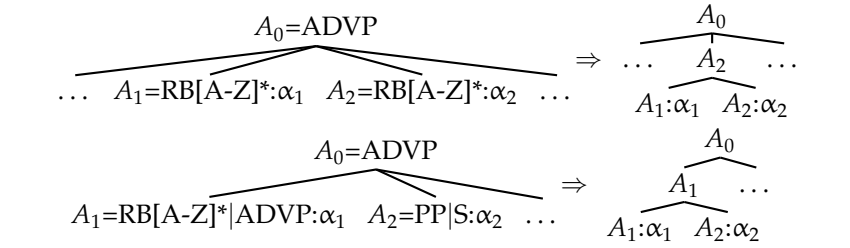
2. VP: left-binarize basal VPs as much as possible; then right-binarize VPs after right context reduced to nil:



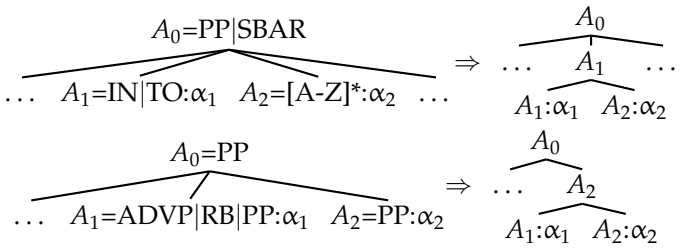
3. ADJP: right-binarize basal ADJPs as much as possible; then left-binarize ADJPs after left context reduced to nil:



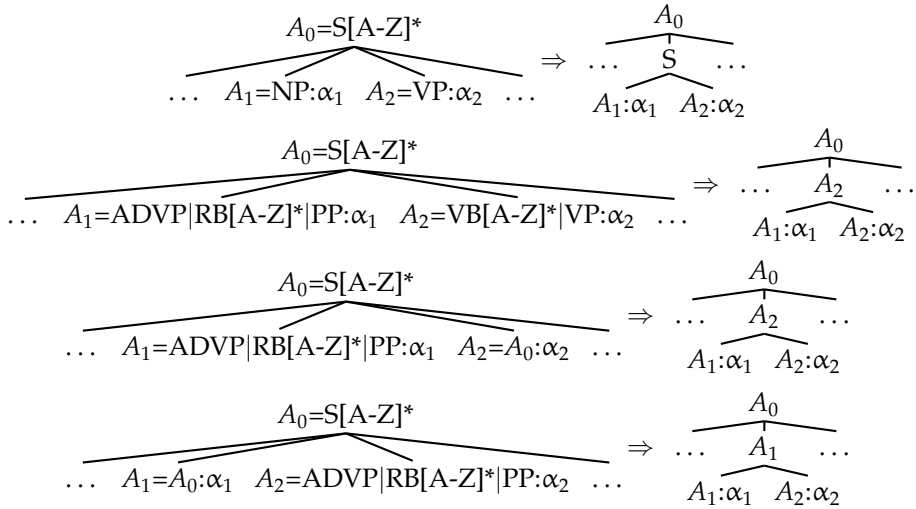
4. ADVP: right-binarize basal ADVPs as much as possible; then left-binarize ADVPs after left context reduced to nil:



5. PP: left-binarize PPs as much as possible; then right-binarize PPs after right context reduced to nil:



6. S: group subject NP and predicate VP of a sentence; then group modifiers to right and left:



Acknowledgments

The authors would like to thank the anonymous reviewers for their input. This research was supported by National Science Foundation CAREER/PECASE award 0447685 and by NASA award NNX08AC36A. The views expressed are not necessarily endorsed by the sponsors.

References

Abney, Steven P. and Mark Johnson. 1991. Memory requirements and local ambiguities of parsing strategies. *J. Psycholinguistic Research*, 20(3):233–250.

Ades, Anthony E. and Mark Steedman. 1982. On the order of words. *Linguistics and Philosophy*, 4:517–558.

Aho, Alfred V. and Jeffery D. Ullman. 1972. *The Theory of Parsing, Translation and Compiling; Volume. I: Parsing*. Prentice-Hall, Englewood Cliffs, NJ.

Baker, James. 1975. The Dragon system: an overview. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 23(1):24–29.

Bellman, Richard. 1957. *Dynamic Programming*. Princeton University Press, Princeton, NJ.

Berg, George. 1992. A connectionist parser with recursive sentence structure and lexical disambiguation. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 32–37, San Jose, CA.

Bever, Thomas G. 1970. The cognitive basis for linguistic structure. In J. R. Hayes, editor, *Cognition and the Development of Language*. Wiley, New York, pages 279–362.

Brown-Schmidt, Sarah, Ellen Campana, and Michael K. Tanenhaus. 2002. Reference resolution in the wild: Online circumscription of referential domains in a

natural interactive problem-solving task. In *Proceedings of the 24th Annual Meeting of the Cognitive Science Society*, pages 148–153, Fairfax, VA.

Charniak, Eugene. 2000. A maximum-entropy inspired parser. In *Proceedings of the First Meeting of the North American Chapter of the Association for Computational Linguistics (ANLP-NAACL'00)*, pages 132–139, Seattle, WA.

Chomsky, Noam and George A. Miller. 1963. Introduction to the formal analysis of natural languages. In *Handbook of Mathematical Psychology*. Wiley, New York, pages 269–321.

Collins, Michael. 1999. *Head-driven statistical models for natural language parsing*. Ph.D. thesis, University of Pennsylvania.

Cowan, Nelson. 2001. The magical number 4 in short-term memory: A reconsideration of mental storage capacity. *Behavioral and Brain Sciences*, 24:87–185.

Crain, Stephen and Mark Steedman. 1985. On not being led up the garden path: The use of context by the psychological syntax processor. In D. R. Dowty, L. Karttunen, and A. M. Zwicky, editors, *Natural Language Parsing: Psychological, Computational, and Theoretical Perspectives*, number 1 in Studies in Natural Language Processing. Cambridge University Press, Cambridge, pages 320–358.

Earley, Jay. 1970. An efficient context-free parsing algorithm. *CACM*, 13(2):94–102.

Elman, Jeffrey L. 1991. Distributed representations, simple recurrent networks, and grammatical structure. *Machine Learning*, 7:195–225.

Ericsson, K. Anders and Walter Kintsch. 1995. Long-term working memory. *Psychological Review*, 102:211–245.

Frege, Gottlob. 1892. Uber sinn und bedeutung. *Zeitschrift fur Philosophie und Philosophischekritik*, 100:25–50.

Gibson, Edward. 1991. *A computational theory of human linguistic processing: Memory limitations and processing breakdown*. Ph.D. thesis, Carnegie Mellon University.

Godfrey, John J., Edward C. Holliman, and Jane McDaniel. 1992. Switchboard: Telephone speech corpus for research and development. In *Proceedings of ICASSP*, pages 517–520, San Francisco, CA.

Gordon, N. J., D. J. Salmond, and A. F. M. Smith. 1993. Novel approach to nonlinear/non-gaussian bayesian state estimation. *IEE Proceedings F (Radar and Signal Processing)*, 140(2):107–113.

Gorrell, Paul. 1995. *Syntax and Parsing*. Cambridge University Press, Cambridge.

Hale, John. 2001. A probabilistic earley parser as a psycholinguistic model. In *Proceedings of the Second Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 159–166, Pittsburgh, PA.

Hale, John. 2006. Uncertainty about the rest of the sentence. *Cognitive Science*, 30(4):609–642.

Helasvuo, Marja-Liisa. 2004. Shared syntax: the grammar of co-constructions. *Journal of Pragmatics*, 36:1315–1336.

Henderson, James. 2004. Lookahead in deterministic left-corner parsing. In *Proceedings Workshop on Incremental Parsing: Bringing Engineering and Cognition Together*, pages 26–33, Barcelona.

Hobbs, Jerry R., Douglas E. Appelt, John Bear, David Israel, Megumi Kameyama, Mark Stickel, and Mabry Tyson. 1996. Fastus: A cascaded finite-state transducer for extracting information from natural-language text. In Yves Schabes, editor, *Finite State Devices for Natural Language Processing*. MIT Press, Cambridge, MA, pages 383–406.

Jarvella, Robert J. 1971. Syntactic processing of connected speech. *Journal of Verbal Learning and Verbal Behavior*, 10:409–416.

Jelinek, Frederick, Lalit R. Bahl, and Robert L. Mercer. 1975. Design of a linguistic statistical decoder for the recognition of continuous speech. *IEEE Transactions on Information Theory*, 21:250–256.

Johnson, Mark. 1998a. Finite state approximation of constraint-based grammars using left-corner grammar transforms. In *Proceedings of COLING/ACL*, pages 619–623, Montreal.

Johnson, Mark. 1998b. PCFG models of linguistic tree representation. *Computational Linguistics*, 24:613–632.

Johnson-Laird, P. N. 1983. *Mental Models: Towards a Cognitive Science of Language, Inference and Consciousness*. Harvard University Press, Cambridge, MA.

Just, Marcel Adam and Patricia A. Carpenter. 1992. A capacity theory of comprehension: Individual differences in working memory. *Psychological Review*, 99:122–149.

Just, Marcel Adam and Sashank Varma. 2007. The organization of thinking: What functional brain imaging reveals about the neuroarchitecture of complex cognition. *Cognitive, Affective, & Behavioral Neuroscience*, 7:153–191.

Kamide, Yuki and Don C. Mitchell. 1999. Incremental pre-head attachment in Japanese parsing. *Language and Cognitive Processes*, 14:631–662.

Klein, Dan and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 423–430, Sapporo.

Lerner, Gene H. 1991. On the syntax of sentences in progress. *Language in Society*, 20:441–458.

Levy, Roger. 2008. Modeling the effects of memory on human online sentence processing with particle filters. In *Proceedings of NIPS*, pages 937–944, Vancouver.

Lewis, Richard L. and Shravan Vasishth. 2005. An activation-based model of sentence processing as skilled memory retrieval. *Cognitive Science*, 29(3):375–419.

Magerman, David. 1995. Statistical decision-tree models for parsing. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics (ACL'95)*, pages 276–283, Cambridge, MA.

Marcus, Mitch. 1980. *Theory of Syntactic Recognition for Natural Language*. MIT Press, Cambridge, MA.

Marcus, Mitchell P., Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Mayberry, III, Marshall R. and Risto Miikkulainen. 2003. Incremental nonmonotonic parsing through semantic self-organization. In *Proceedings of the* 25th *Annual Conference of the Cognitive Science Society*, pages 798–803, Boston, MA.

Miller, George A. 1956. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review*, 63:81–97.

Murphy, Kevin P. and Mark A. Paskin. 2001. Linear time inference in hierarchical HMMs. In *Proceedings of NIPS*, pages 833–840, Vancouver.

Pollard, Carl and Ivan A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. Chicago: University of Chicago Press and Stanford: CSLI Publications.

Pritchett, Bradley L. 1991. Head position and parsing ambiguity. *Journal of Psycholinguistic Research*, 20:251–270.

Resnik, Philip. 1992. Left-corner parsing and psychological plausibility. In *Proceedings of COLING*, pages 191–197, Nantes.

Roark, Brian. 2001. Probabilistic top-down parsing and language modeling. *Computational Linguistics*, 27(2):249–276.

Rohde, Douglas L. T. 2002. *A connectionist model of sentence comprehension and production*. Ph.D. thesis, Computer Science Department, Carnegie Mellon University.

Sachs, Jacqueline. 1967. Recognition memory for syntactic and semantic aspects of connected discourse. *Perception and Psychophysics*, 2:437–442.

Schuler, William. 2009. Parsing with a bounded stack using a model-based right-corner transform. In *Proceedings of the North American Association for Computational Linguistics (NAACL '09)*, pages 344–352, Boulder, CO.

Schuler, William, Stephen Wu, and Lane Schwartz. 2009. A framework for fast incremental interpretation during speech decoding. *Computational Linguistics*, 35(3):313–343.

Shieber, Stuart. 1985. Evidence against the context-freeness of natural language. *Linguistics and Philosophy*, 8:333–343.

Smolensky, P. and G. Legendre. 2006. *The Harmonic Mind: From Neural Computation to Optimality-Theoretic Grammar*. MIT Press, Cambridge, MA.

Steedman, Mark. 2000. *The Syntactic Process*. MIT Press/Bradford Books, Cambridge, MA.

Stevenson, Suzanne. 1998. Parsing as incremental restructuring. In J. D. Fodor and F. Ferreira, editors, *Reanalysis in Sentence Processing*. Kluwer Academic, Boston, MA, pages 327–363.

Tanenhaus, Michael K., Michael J. Spivey-Knowlton, Kathy M. Eberhard, and Julie E. Sedivy. 1995. Integration of visual and linguistic information in spoken language comprehension. *Science*, 268:1632–1634.