Toward a Psycholinguistically-Motivated Model of Language Processing

William Schuler
Computer Science and Engineering
University of Minnesota
schuler@cs.umn.edu

Samir AbdelRahman
Department of Computer Science
Cairo University
s.abdelrahman@fci-cu.edu.eg

Tim Miller
Computer Science and Engineering
University of Minnesota
tmill@cs.umn.edu

Lane Schwartz
Computer Science and Engineering
University of Minnesota
lschwar@cs.umn.edu

Abstract

Psycholinguistic studies suggest a model of human language processing that 1) performs incremental interpretation of spoken utterances or written text, 2) preserves ambiguity by maintaining competing analyses in parallel, and 3) operates within a severely constrained short-term memory store — possibly constrained to as few as four distinct elements. This paper describes a relatively simple model of language as a factored statistical time-series process that meets all three of the above desiderata; and presents corpus evidence that this model is sufficient to parse naturally occurring sentences using human-like bounds on memory.

1 Introduction

Psycholinguistic studies suggest a model of human language processing with three important properties. First, eye-tracking studies (Tanenhaus et al., 1995; Brown-Schmidt et al., 2002) suggest that humans analyze sentences incrementally, assembling and interpreting referential expressions even while they are still being pronounced. Second, humans appear to maintain competing analyses in parallel, with eye gaze showing significant attention to competitors (referents of words with similar prefixes to the correct word), even relatively long after the end of the word has been encountered, when attention to other distractor referents has fallen off (Dahan and Gaskell, 2007). Preserving ambiguity in a parallel, non-deterministic search like this may account for human robustness to missing, unknown, mispronounced, or misspelled words. Finally, studies of short-term memory capacity suggest human language processing operates within a severely constrained short-term memory store — possibly restricted to as few as four distinct elements (Miller, 1956; Cowan, 2001).

The first two observations may be taken to endorse existing probabilistic beam-search models which maintain multiple competing analyses, pruned by contextual preferences and dead ends (e.g. Roark, 2001). But the last observation on memory bounds imposes a restriction that until now has not been evaluated in a corpus study. Can a simple, useful human-like processing model be defined using these constraints? This paper describes a relatively simple model of language as a factored statistical time-series process that meets all three of the above desiderata; and presents corpus evidence that this model is sufficient to parse naturally occurring sentences using human-like bounds on memory.

The remainder of this paper is organized as follows: Section 2 describes some current approaches to incremental parsing; Section 3 describes a statistical framework for parsing using a bounded stack of explicit constituents; Section 4 describes an experiment to estimate the level of coverage of the Penn Treebank corpus that can be achieved with various stack memory limits, using a set of reversible tree transforms, and gives accuracy results of a bounded-memory model trained on this corpus.

2 Background

Much work on cognitive modeling in psycholinguistics is centered on modeling the *concepts* to which utterances refer. Coarsely, these concepts may correspond to activation patterns among neurons in specific regions of the brain. In some theories, a short-term memory store of several unrelated concepts may be retained by organizing the activation of these concepts into compatible patterns, only a few of which can be reliably main-

Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008), pages 785–792
Manchester, August 2008

tained (Smolensky and Legendre, 2006). Activation is then theorized to spread through and among these groups of concepts in proportion to some learned probability that the concepts will be relevant (Anderson and Reder, 1999), with the most active concepts corresponding to the most likely linguistic analyses. Competition between rival activated groups of concepts (corresponding to incomplete linguistic analyses) has even been linked to reading delays (Hale, 2003).

This competition among mutually-exclusive variously-activated short term memory stores of concepts, essentially a weighted disjunction over conjunctions of concepts, can be modeled in language understanding as simple Viterbi decoding of a factored HMM-like time-series model (Schuler et al., in press). In this model, concepts (corresponding to vectors of individuals in a first-order world model) are introduced and composed (via set operations like intersection) in each hypothesized short-term memory store, using the elements of the memory store as a stack. These vectors of individuals can be considered a special case of vectors of concept elements proposed by Smolensky, with set intersection a special case of tensor product in the composition model. Referents in this kind of incremental model can be constrained by — but still distinguished from — higher-level referents while they are still being recognized.

It is often assumed that this semantic concept composition proceeds isomorphically with the composition of syntactic constituents (Frege, 1892). This parallel semantic and syntactic composition is considered likely to be performed in short-term memory because it has many of the characteristics of short-term memory processes, including nesting limits (Miller and Chomsky, 1963) and susceptibility to degradation due to interruption. Ericsson and Kintch (1995) propose a theory of long-term working memory that extends short-term memory, but only for inter-sentential references, which *do* seem to be retained across interruptions in reading. But while the relationship between competing probability distributions in such a model and experimental reading times has been evaluated (e.g. by Hale), the relationship between the syntactic demands on a short-term memory store and observations of human short-term memory limits is still largely untested. Several models have been proposed to perform syntactic analysis using a bounded memory store.

For example, Marcus (1980) proposed a deterministic parser with an explicit four-element working memory store in order to model human parsing limitations. But this model only stores *complete* constituents (whereas the model proposed in this paper stores *incompletely recognized* constituents, in keeping with the Tanenhaus et al. findings). As a result, the Marcus model relies on a suite of specialized memory operations to compose complete constituents out of complete constituents, which are not independently cognitively motivated.

Cascaded finite-state automata, as in FASTUS (Hobbs et al., 1996), also make use of a bounded stack, but stack levels in such systems are typically dedicated to particular syntactic operations: e.g. a word group level, a phrasal level, and a clausal level. As a result, some amount of constituent structure may overflow its dedicated level, and be sacrificed (for example, prepositional phrase attachment may be left underspecified).

Finite-state equivalent parsers (and thus, bounded-stack parsers) have asymptotically linear run time. Other parsers (Sagae and Lavie, 2005) have achieved linear runtime complexity with unbounded stacks in incremental parsing by using a greedy strategy, pursuing locally most probable shift or reduce operations, conditioned on multiple surrounding words. But without an explicit bounded stack it is difficult to connect these models to concepts in a psycholinguistic model.

Abney and Johnson (1991) explore left-corner parsing as a memory model, but again only in terms of (complete) syntactic constituents. The approach explored here is similar, but the transform is reversed to allow the recognizer to store recognized structure rather than structures being sought, and the transform is somewhat simplified to allow more structure to be introduced into syntactic constituents, primarily motivated by a need to keep track of disconnected semantic concepts rather than syntactic categories. Without this link to disconnected semantic concepts, the syntax model would be susceptible to criticism that the separate memory levels could be simply chunked together through repeated use (Miller, 1956).

Roark's (2001) top-down parser generates trees incrementally in a transformed representation related to that used in this paper, but requires distributions to be maintained over entire trees rather than stack configurations. This increases the beam

width necessary to avoid parse failure. Moreover, although the system is conducting a beam search, the objects in this beam are growing, so the recognition complexity is not linear, and the connection to a bounded short-term memory store of unconnected concepts becomes somewhat complicated.

The model described in this paper is arguably simpler than many of the models described above in that it has no constituent-specific mechanisms, yet it is able to recognize the rich syntactic structures found in the Penn Treebank, and is still compatible with the psycholinguistic notion of a bounded short-term memory store of conceptual referents.

3 Bounded-Memory Parsing with a Time Series Model

This section describes a basic statistical framework — a factored time-series model — for recognizing hierarchic structures using a bounded store of memory elements, each with a finite number of states, at each time step. Unlike simple FSA compilation, this model maintains an explicit representation of active, incomplete phrase structure constituents on a bounded stack, so it can be readily extended with additional variables that depend on syntax (e.g. to track hypothesized entities or relations). These incomplete constituents are related to ordinary phrase structure annotations through a series of bidirectional tree transforms. These transforms:

1. binarize phrase structure trees into linguistically motivated head-modifier branches (described in Section 3.1);
2. transform right-branching sequences to left-branching sequences (described in Section 3.2); and
3. align transformed trees to an array of random variable values at each depth and time step of a probabilistic time-series model (described in Section 3.3).

Following these transforms, a model can be trained from example trees, then run as a parser on unseen sentences. The transforms can then be reversed to evaluate the output of the parser. This representation will ultimately be used to evaluate the coverage of a bounded-memory model on a large corpus of tree-annotated sentences, and to evaluate the accuracy of a basic (unsmoothed, unlexicalized) implementation of this model in Section 4.

It is important to note that these transformations are not postulated to be part of the human recognition process. In this model, sentences can be recognized and interpreted entirely in right-corner form. The transforms only serve to connect this process to familiar representations of phrase structure.

3.1 Binary branching structure

This paper will attempt to draw conclusions about the syntactic complexity of natural language, in terms of stack memory requirements in incremental (left-to-right) recognition. These requirements will be minimized by recognizing trees in a *right-corner* form, which accounts partially recognized phrases and clauses as *incomplete constituents*, lacking one instance of another constituent yet to come.

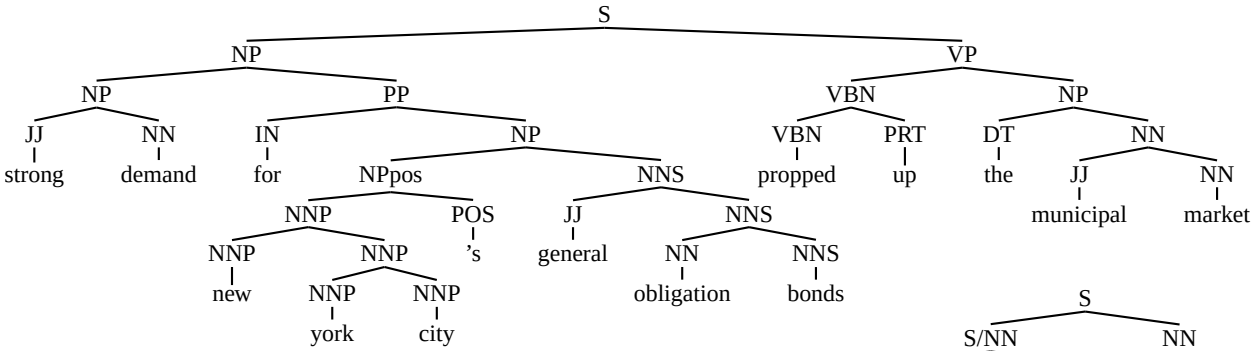
In particular, this study will use the trees in the Penn Treebank Wall Street Journal (WSJ) corpus (Marcus et al., 1994) as a data set. In order to obtain a linguistically plausible right-corner transform representation of incomplete constituents, the corpus is subjected to another, pre-process transform 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.

3.2 Right-Corner Transform

Phrase structure trees are recognized in this framework in a *right-corner* form that can be mapped to and from ordinary phrase structure via reversible transform rules, similar to those described by Johnson (1998a). This transformed grammar constrains memory usage in left-to-right traversal to a bound consistent with the psycholinguistic results described above.

This right-corner transform is simply the left-right dual of a left-corner transform (Johnson, 1998a). It transforms all right branching sequences in a phrase structure tree into left branching sequences of symbols of the form A_1/A_2, denoting an incomplete instance of category A_1 lacking an instance of category A_2 to the right. These incomplete constituent categories have the same form

a) binarized phrase structure tree:



b) result of right-corner transform:

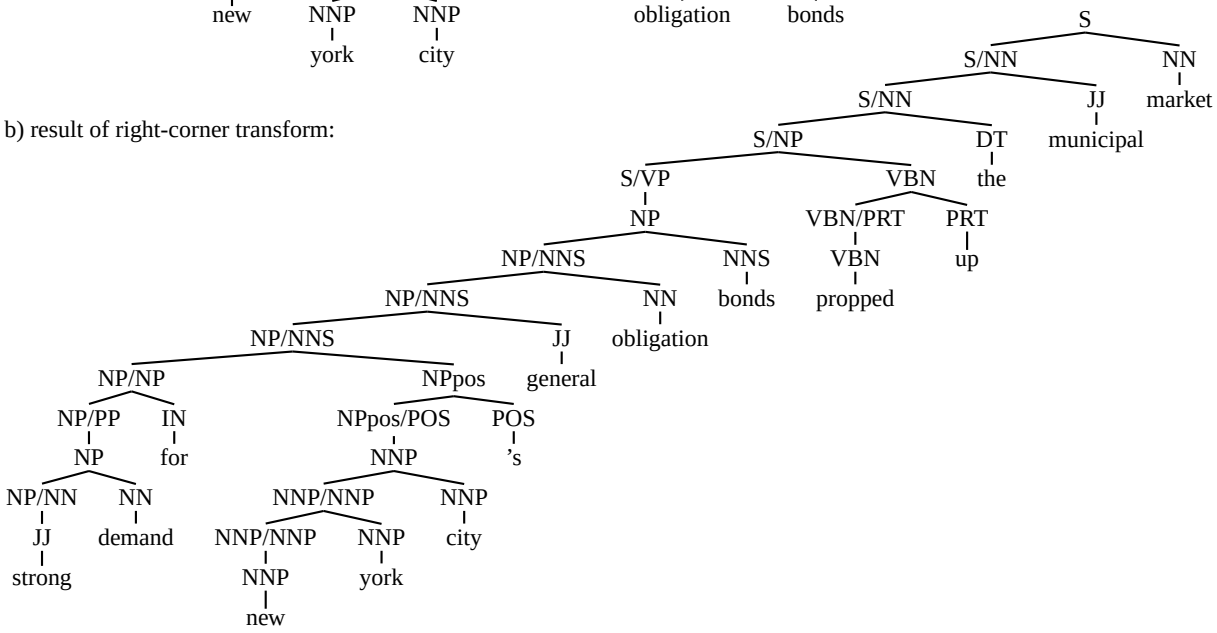
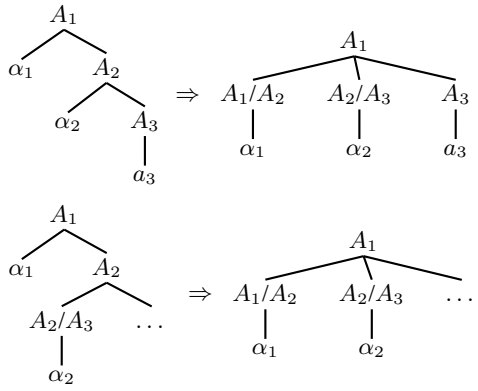


Figure 1: Trees resulting from a) a binarization of a sample phrase structure tree for the sentence *Strong demand for New York City's general obligations bonds propped up the municipal market,* and b) a right-corner transform of this binarized tree.

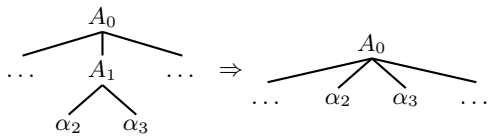
and much of the same meaning as non-constituent categories in a Combinatorial Categorial Grammar (Steedman, 2000).

Rewrite rules for the right-corner transform are shown below, first to flatten out right-branching structure:[1]

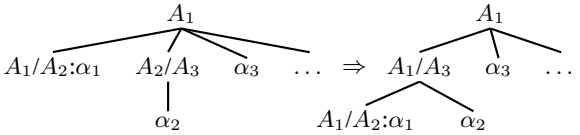


then to replace it with left-branching structure:

[1]The tree transforms presented in this paper 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. 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 paper, 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 ($\ldots$) 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 above).



Here, the first two rewrite rules are applied iteratively (bottom-up on the tree) to flatten all right branching structure, using incomplete constituents to record the original nonterminal ordering. The third rule is then applied to generate left-branching structure, preserving this ordering. Note that the last rewrite above leaves a unary branch at the leftmost child of each flattened node. This preserves the nodes at which the original tree was not right-branching, so the original tree can be reconstructed when the right-corner transform concatenates multiple right-branching sequences into a single left-branching sequence.

An example of a right-corner transformed tree is shown in Figure 1(b). 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 above. The correctness of this can be demonstrated by dividing a tree into maximal sequences of right 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 original version of the flattened tree.

3.3 Hierarchic Hidden Markov Models

Right-corner transformed phrase structure trees can then be mapped to random variable positions in a Hierarchic Hidden Markov Model (Murphy and Paskin, 2001), essentially a Hidden Markov Model (HMM) factored into some fixed number of stack levels at each time step.

HMMs characterize speech or text as a sequence of hidden states q_t (in this case, stacked-up syntactic categories) and observed states o_t (in this case, words) at corresponding time steps t. A most likely sequence of hidden states $\hat{q}_{1..T}$ can then be hypothesized given any sequence of observed states $o_{1..T}$, 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 *Transition 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 $\mathsf{P}(o_{1..T} \mid q_{1..T}) \stackrel{\text{def}}{=} \prod_t \mathsf{P}_{\Theta_{\mathrm{B}}}(o_t \mid q_t)$:

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

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

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

Transition probabilities $\mathsf{P}_{\Theta_{\mathrm{A}}}(q_t \mid q_{t-1})$ over complex hidden states q_t can be modeled using synchronized levels of stacked-up component HMMs in a Hierarchic Hidden Markov Model (HHMM) (Murphy and Paskin, 2001). HHMM transition probabilities are calculated in two phases: a *reduce* phase (resulting in an intermediate, marginalized state f_t), in which component HMMs may terminate; and a *shift* phase (resulting in a 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 HMM 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* Θ_{R} and *shift* Θ_{S} models:

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

$$\stackrel{\text{def}}{=} \sum_{f_t^{1..D}} \prod_{d=1}^{D} \mathsf{P}_{\Theta_{\mathrm{R}}}(f_t^d \mid f_t^{d+1} q_{t-1}^d q_{t-1}^{d-1}) \cdot \mathsf{P}_{\Theta_{\mathrm{S}}}(q_t^d \mid f_t^{d+1} f_t^d q_{t-1}^d q_t^{d-1}) \tag{7}$$

with f_t^{D+1} and q_t^0 defined as constants. In Viterbi decoding, the sums are replaced with argmax operators. This decoding process preserves ambiguity by maintaining competing analyses of the entire memory store. A graphical representation of an HHMM with three levels is shown in Figure 3.

Shift and reduce probabilities can then be defined in terms of finitely recursive Finite State Automata (FSAs) with probability distributions over

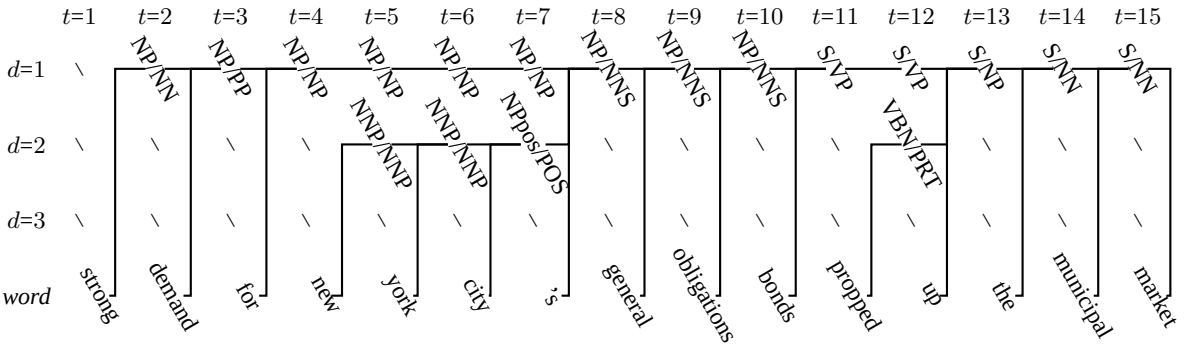


Figure 2: Sample tree from Figure 1 mapped to q_t^d variable positions of an HHMM at each stack depth d (vertical) and time step t (horizontal). This tree uses only two levels of stack memory. Values for final-state variables f_t^d are not shown. Note that some nonterminal labels have been omitted; labels for these nodes can be reconstructed from their children.

transition, recursive expansion, and final-state status of states at each hierarchy level. In simple HHMMs, each intermediate variable is a boolean variable over final-state status $f_t^d \in \{\mathbf{0}, \mathbf{1}\}$ and each modeled state variable is a syntactic, lexical, or phonetic state q_t^d. The intermediate variable f_t^d is true or false (equal to $\mathbf{1}$ or $\mathbf{0}$ respectively) according to $\Theta_{\text{F-Reduce}}$ if there is a transition at the level immediately below d, and false (equal to $\mathbf{0}$) with probability 1 otherwise:[2]

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

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 $\Theta_{\text{Q-Trans}}$ and expansion $\Theta_{\text{Q-Expand}}$ models:

$$\mathsf{P}_{\Theta_{\text{S}}}(q_t^d \mid f_t^{d+1} f_t^d \, q_{t-1}^d q_t^{d-1}) \stackrel{\text{def}}{=} \begin{cases} \text{if } f_t^{d+1} = \mathbf{0},\ f_t^d = \mathbf{0} : [q_t^d = q_{t-1}^d] \\ \text{if } f_t^{d+1} = \mathbf{1},\ f_t^d = \mathbf{0} : \mathsf{P}_{\Theta_{\text{Q-Trans}}}(q_t^d \mid q_{t-1}^d q_t^{d-1}) \\ \text{if } f_t^{d+1} = \mathbf{1},\ f_t^d = \mathbf{1} : \mathsf{P}_{\Theta_{\text{Q-Expand}}}(q_t^d \mid q_t^{d-1}) \end{cases} \quad (9)$$

where $f_t^{D+1} = \mathbf{1}$ and $q_t^0 = \mathbf{ROOT}$. This model is conditioned on final-state switching variables at and immediately below the current FSA level. If there is no final state immediately below the current level (the first case above), it deterministically copies the current FSA state forward to the next time step. If there is a final state immediately below the current level (the second case above), it transitions the FSA state at the current level, according to the distribution $\Theta_{\text{Q-Trans}}$. And if the state at the current level is final (the third case above), it re-initializes this state given the state at the level above, according to the distribution $\Theta_{\text{Q-Expand}}$. The overall effect is that higher-level FSAs are allowed to transition only when lower-level FSAs terminate. An HHMM therefore behaves like a probabilistic implementation of a pushdown automaton (or shift–reduce parser) with a finite stack, where the maximum stack depth is equal to the number of levels in the HHMM hierarchy.

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

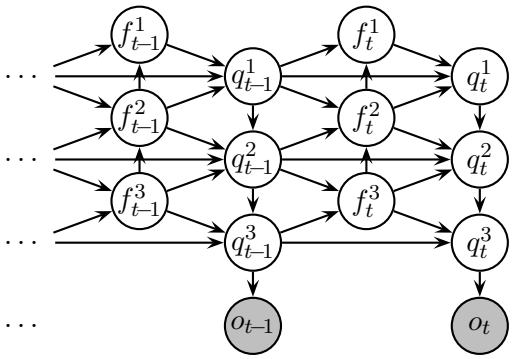


Figure 3: Graphical representation of a Hierarchic Hidden Markov Model. Circles denote random variables, and edges denote conditional dependencies. Shaded circles are observations.

Figure 2 shows the transformed tree from Figure 1 aligned to HHMM depth levels and time steps. Because it uses a bounded stack, recognition in this model is asymptotically linear (Murphy and Paskin, 2001).

This model recognizes right-corner transformed trees constrained to a stack depth corresponding to observed human short term memory limits. This

HHMM depth limit	sentences	coverage
no memory	127	0.32%
1 memory element	3,496	8.78%
2 memory elements	25,909	65.05%
3 memory elements	38,902	97.67%
4 memory elements	39,816	99.96%
5 memory elements	39,832	100.00%
TOTAL	39,832	100.00%

Table 1: Percent coverage of right-corner transformed treebank sections 2–21 with punctuation omitted, using HHMMs with depth limits D from zero to five.

is an attractive model of human language processing because the incomplete syntactic constituents it stores at each stack depth can be directly associated with (incomplete) semantic referents, e.g. by adding random variables over environment or discourse referents at each depth and time step. If these referents are calculated incrementally, recognition decisions can be informed by the values of these variables in an interactive model of language, following Tanenhaus et al. (1995). The corpus results described in the next section suggest that a large majority of naturally occurring sentences can be recognized using only three or four stack memory elements via this transform.

4 Empirical Results

In order to evaluate the coverage of this bounded-memory model, Sections 2–21 of the Penn Treebank WSJ corpus were transformed and mapped to HHMM variables as described in Section 3.3. In order to counter possible undesirable effects of an arbitrary branching analysis of punctuation, punctuation was removed. Coverage results on this corpus are shown in Table 1.

Experiments training on transformed trees from Sections 2–21 of the WSJ Treebank, evaluating reversed-transformed output sequences from Section 22 (development set) and Section 23 (test set), show an accuracy (F score) of 82.1% and 80.1% respectively.[3] Although they are lower than those for state-of-the-art parsers, these results suggest that the bounded-memory parser described here is doing a reasonably good job of modeling syntactic dependencies, and therefore may have some promise as a psycholinguistic model.

Although recognition in this system is linear, it essentially works top-down, so it has larger runtime constants than a bottom-up CKY-style parser. The experimental system described above runs at a rate of about 1 sentence per second on a 64-bit 2.6GHz dual core desktop with a beam width of 2000. In comparison, the Klein and Manning (2003) CKY-style parser runs at about 5 sentences per second on the same machine. On sentences longer than 40 words, the HHMM and CKY-style parsers are roughly equivalent, parsing at the rate of .21 sentences per second, versus .24 for the Klein and Manning CKY.

But since it is linear, the HHMM parser can be directly integrated with end-of-sentence detection (e.g. deciding whether '.' is a sentence delimiter based on whether the words preceding it can be reduced as a sentence), or with n-gram language models (if words are observations, this is simply an autoregressive HMM topology). The use of an explicit constituent structure in a time series model also allows integration with models of dynamic phenomena such as semantics and coreference which may depend on constituency. Finally, as a linear model, it can be directly applied to speech recognition (essentially replacing the hidden layer of a conventional word-based HMM language model).

5 Conclusion

This paper has described a basic incremental parsing model that achieves worst-case linear time complexity by enforcing fixed limits on a stack of explicit (albeit incomplete) constituents. Initial results show a use of only three to four levels of stack memory within this framework provides nearly complete coverage of the large Penn Treebank corpus.

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. The views expressed are not necessarily endorsed by the sponsors.

[3] Using unsmoothed relative frequency estimates from the training set, a depth limit of $D = 3$, beam with of 2000, and no lexicalization.

References

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

Anderson, J.R. and L.M. Reder. 1999. The fan effect: New results and new theories. *Journal of Experimental Psychology: General*, 128(2):186–197.

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, August.

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

Dahan, Delphine and M. Gareth Gaskell. 2007. The temporal dynamics of ambiguity resolution: Evidence from spoken-word recognition. *Journal of Memory and Language*, 57(4):483–501.

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.

Hale, John. 2003. *Grammar, Uncertainty and Sentence Processing*. Ph.D. thesis, Cognitive Science, The Johns Hopkins University.

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 *Finite State Devices for Natural Language Processing*, pages 383–406. MIT Press, Cambridge, MA.

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

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

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.

Marcus, Mitchell P., Beatrice Santorini, and Mary Ann Marcinkiewicz. 1994. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Marcus, Mitch. 1980. *A theory of syntactic recognition for natural language*. MIT Press.

Miller, George and Noam Chomsky. 1963. Finitary models of language users. In Luce, R., R. Bush, and E. Galanter, editors, *Handbook of Mathematical Psychology*, volume 2, pages 419–491. John Wiley.

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 *Proc. NIPS*, pages 833–840.

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

Sagae, Kenji and Alon Lavie. 2005. A classifier-based parser with linear run-time complexity. In *Proceedings of the Ninth International Workshop on Parsing Technologies (IWPT'05)*.

Schuler, William, Stephen Wu, and Lane Schwartz. in press. A framework for fast incremental interpretation during speech decoding. *Computational Linguistics*.

Smolensky, Paul and Géraldine Legendre. 2006. *The Harmonic Mind: From Neural Computation to Optimality-Theoretic GrammarVolume I: Cognitive Architecture*. MIT Press.

Steedman, Mark. 2000. *The syntactic process*. MIT Press/Bradford Books, Cambridge, MA.

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.