Sentence Processing in a Vectorial Model of Working Memory

William Schuler
Department of Linguistics
The Ohio State University
schuler@ling.osu.edu

Abstract

This paper presents a vectorial incremental parsing model defined using independently posited operations over activation-based working memory and weight-based episodic memory. This model has the attractive property that it hypothesizes only one unary preterminal rule application and only one binary branching rule application per time step, which allows it to be smoothly integrated into a vector-based recurrence that propagates structural ambiguity from one time step to the next. Predictions of this model are calculated on a center-embedded sentence processing task and shown to exhibit decreased processing accuracy in center-embedded constructions.

1 Introduction

Current models of memory (Marr, 1971; Anderson et al., 1977; Murdock, 1982; McClelland et al., 1995; Howard and Kahana, 2002) involve a continuous *activation-based* (or 'working') memory, typically modeled as a vector representing the current firing pattern of neurons or neural clusters in the cortex. This activation-based memory is then supported by a durable but rapidly mutable *weight-based* (or 'episodic') memory, typically modeled as one or more matrices formed by summed outer-products of cue and target vectors and cued by simple matrix multiplication, representing variable synaptic connection strengths between neurons or neural clusters.

The lack of discrete memory units in such models makes it difficult to imagine a neural implementation of a typical e.g. chart-based computational account of sentence processing. On the other hand, superposition in vectorial models suggests a natural representation of a parallel incremental processing model. This paper explores how such an austere model of memory not only might be used to encode a simple probabilistic incremental parser, but also lends itself to naturally implement a vectorial interpreter and coreference resolver. This model is based on the left-corner parser formulation of van Schijndel et al. (2013a), which has the attractive property of generating exactly one binary-branching rule application after processing each word. This property greatly simplifies a vectorial implementation because it allows these single grammar rule applications to be superposed in cases of attachment ambiguity.

Predictions of the vectorial model described in this paper are then calculated on a simple center-embedded sentence processing task, producing a lower completion accuracy for center-embedded sentences than for right-branching sentences with the same number of words. As noted by Levy and Gibson (2013), this kind of memory effect is not easily explained by existing information-theoretic models of frequency effects (Hale, 2001; Levy, 2008).

The model described in this paper also provides an explanation for the apparent reality of linguistic objects like categories, grammar rules, discourse referents and dependency relations, as cognitive states in activation-based memory (in the case of categories and discourse referents), or cued associations in weight-based memory (in the case of grammar rules, and dependency relations), without having to posit complex machinery specific to language processing. In this sense, unlike existing chart-based parsers or connectionist models based on recurrent neural networks, this model integrates familiar notions of grammar and semantic relations with current ideas of activation-based and weight-based memory. It is also anticipated that this interface to both linguistic and neuroscientific theories will make the model useful as a basis for more nuanced understanding of linguistic phenomena such as ambiguity resolution, semantic representation, and language acquisition.

2 Related Work

The model described in this paper is based on the left-corner parser formulation of van Schijndel et al. (2013a), which is an implementation of a fully parallel incremental parser. This parser differs from chart-based fully parallel incremental parsers used by Hale (2001), Levy (2008) and others in that it enforces a cognitively-motivated bound on center-embedding depth. This bound allows the parser to represent a tractable set of incremental hypotheses in an explicitly enumerated list as a factored hidden Markov model, without necessitating the use of a parser chart. This model has the attractive property that, in any context, it hypothesizes exactly one binary-branching rule application at each time step.

The model described in this paper extends the van Schijndel et al. (2013a) parser by maintaining possible store configurations as superposed sequence states in a finite-dimensional state vector. The model then exploits the uniformity of its parsing operations to integrate probabilistically weighted grammar rule applications into this superposed state vector. These superposed states are then used to cue more superordinate sequential states as 'continuations' whenever subordinate states conclude. Interference in this cueing process is then observed to produce a natural center-embedding limit.

This model is defined as a recurrence over an activation vector, similar to the simple recurrent network of Elman (1991) and others, but unlike an SRN, which does not encode anything in weight-based memory during processing, this model encodes updates to a processing hierarchy in weight-based memory at every time step. The model is also similar to the ACT-R parser of Lewis and Vasishth (2005) in that it maintains a single state which is updated based on content-based cued association, but unlike the ACT-R parser, which cues category tokens on category types and therefore models memory limits as interference among grammar rules, this model cues category tokens on other category tokens, and therefore predicts memory limits even in cases where grammar rules do not involve similar category types. Also unlike Lewis and Vasishth (2005), this model is defined purely in terms of state vectors and outer-product associative memory and therefore has the capacity

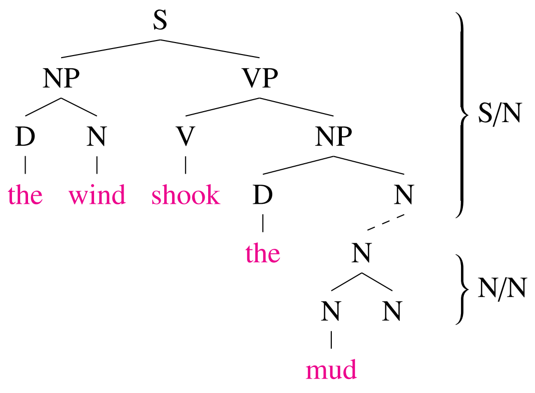


Figure 1: Example incomplete category during processing of the sentence *The wind shook the mud room door*.

to maintain parallel states in superposition.

3 Background: Non-vectorial Incremental Parsing

The model defined in this paper is based on the left-corner parser formulation of van Schijndel et al. (2013a). This parser maintains a set of incomplete categories a/b at each time step, each consisting of an active category a lacking an awaited category b yet to come. For example, Figure 1 shows an incomplete category S/N consisting of a sentence lacking a common noun yet to come, which non-immediately dominates another incomplete category N/N consisting of a common noun lacking another common noun yet to come.

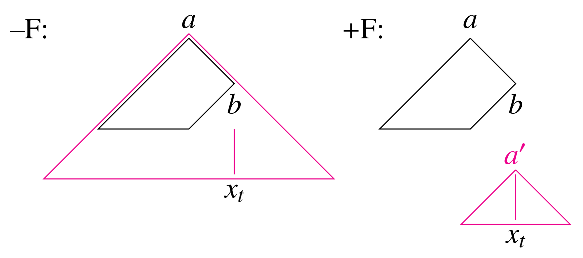
Processing in this model is defined to alternate between two phases:

1. a 'fork' phase in which a word is either used to complete an existing incomplete category, or forked into a new complete category; and

2. a 'join' phase in which one of these complete categories is used as a left child of a grammar rule application and then either joined onto a superordinate incomplete category or kept disjoint.

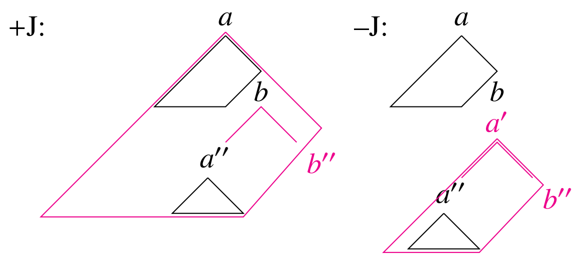
In any case, only one grammar rule is applied after each word. These fork and join operations are shown graphically and as natural deduction rules in Figure 2.

An example derivation of the sentence, *The wind shook the mud room door*, using the productions in Figure 2 is shown in Figure 3, with corresponding partial parse trees shown in Figure 4. Van Schijndel et al. (2013a) show that a



$$\frac{a/b \quad x_t}{a}\; b \rightarrow x_t \tag{–F}$$

$$\frac{a/b \quad x_t}{a/b \quad a'}\; b \overset{+}{\rightarrow} a' \ldots\, ;\; a' \rightarrow x_t \tag{+F}$$



$$\frac{a/b \quad a''}{a/b''}\; b \rightarrow a''\; b'' \tag{+J}$$

$$\frac{a/b \quad a''}{a/b \quad a'/b''}\; b \overset{+}{\rightarrow} a' \ldots\, ;\; a' \rightarrow a''\; b'' \tag{–J}$$

Figure 2: Fork and join operations from the van Schijndel et al. (2013a) left-corner parser formulation. During the fork phase, word x either completes an existing incomplete category a, or forks into a new complete category a'. During the join phase, complete category a'' becomes a left child of a grammar rule application, then either joins onto a superordinate incomplete category a/b or remains disjoint.

probabilistic version of this incremental parser can reproduce the results of a state-of-the-art chart-based parser (Petrov and Klein, 2007).

4 Vectorial Parsing

This left corner parser can be implemented in a vectorial model of working memory using vectors as activation-based memory and matrices as weight-based memory. Following Anderson et al. (1977) and others, vectors v in activation-based memory are cued from other vectors u through weight-based memory matrices M using ordinary

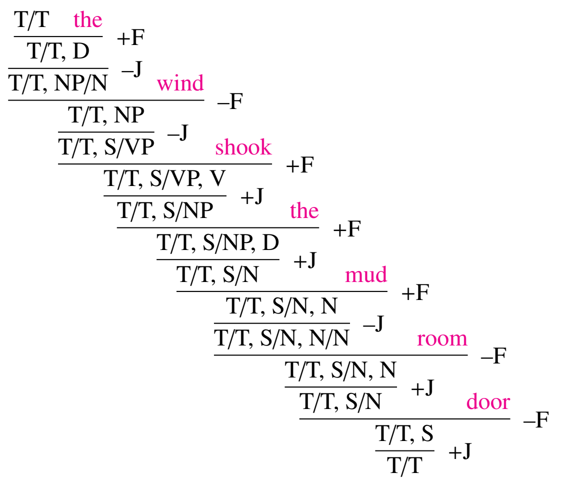


Figure 3: Processing steps in parsing the sentence *The wind shook the mud room door.*

matrix multiplication:[1]

$$v = M\,u \tag{1}$$

This representation has been used to model the influence of activation in antecedent neurons on activation in consequent neurons (Marr, 1971; Anderson et al., 1977).

Unless they are cued from some other source, all vectors in this model are initially randomly generated by sampling from an exponential distribution, denoted here simply by:

$$v \sim \text{Exp} \tag{2}$$

Also following Anderson et al. (1977), weight-based memory matrices M are themselves defined and updated by simply adding outer products of desired cue u and target v vectors:[2]

$$M_t = M_{t\text{-}1} + v \otimes u \tag{3}$$

This representation has been used to model rapid synaptic sensitization in the hippocampus (Marr, 1971; McClelland et al., 1995), in which synapses of activated antecedent neurons that impinge on activated consequent neurons are strengthened.

[1]That is, multiplication of an associative memory matrix M by a state vector v yields:

$$(M\,v)_{[i]} \overset{\text{def}}{=} \textstyle\sum_{j=1}^{J} M_{[i,j]} \cdot v_{[j]} \tag{1'}$$

[2]An outer product $v \otimes u$ defines a matrix by multiplying each combination of scalars in vectors v and u:

$$(v \otimes u)_{[j,i]} \overset{\text{def}}{=} v_{[j]} \cdot u_{[i]} \tag{2'}$$

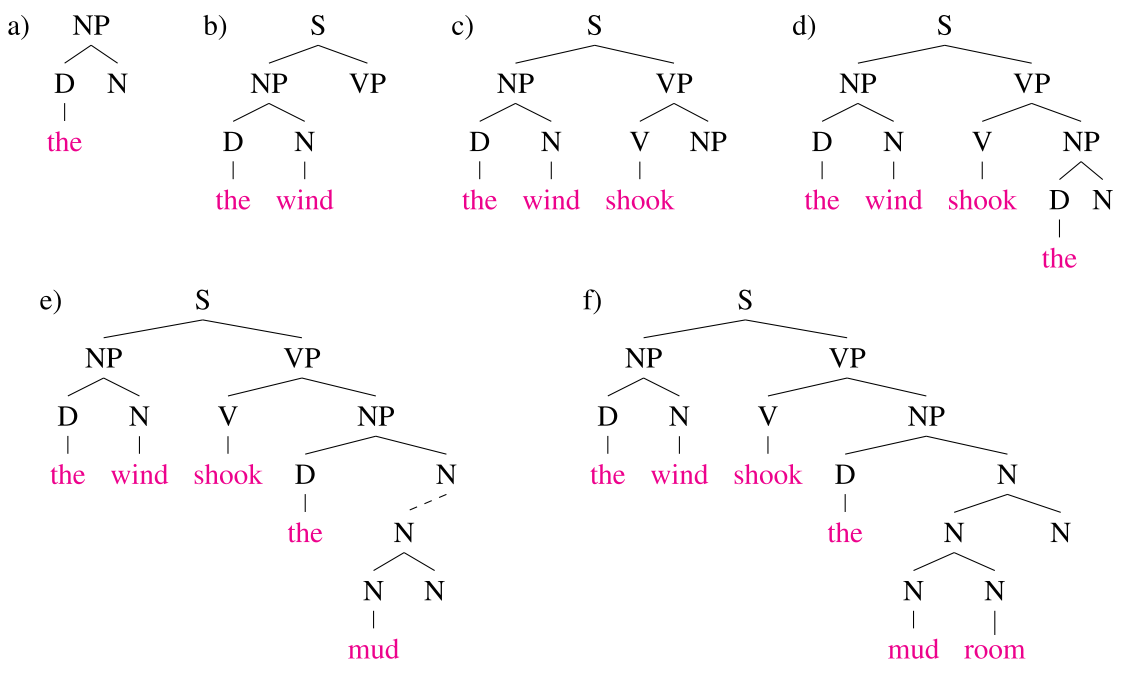

a) NP D N the
b) S NP VP D N the wind
c) S NP VP D N V NP the wind shook
d) S NP VP D N V NP D N the wind shook the
e) S NP VP D N V NP D N the wind shook the N N N mud
f) S NP VP D N V NP D N the wind shook the N N N N mud room

Figure 4: Processing steps in parsing the sentence *The wind shook the mud room door*.

Finally, cued associations can be combined using pointwise or diagonal products:[3]

$$w = \operatorname{diag}(u)\, v \tag{4}$$

Unlike a symbolic statistical model, a vectorial model must explicitly distinguish token representations from types in order to define structural relations that would be implicit in the positions of data structure elements in a symbolic model. Thus, the active or awaited distinction is applied to category tokens rather than types, but grammar rule applications are defined over category types rather than tokens.

The vectorial left-corner parser described in this paper is therefore defined on a single category token vector b_t which encodes the awaited category token of the most subordinate incomplete category at the current time step t. A hierarchy of nested incomplete category tokens is then encoded in two 'continuation' matrices:

- A_t, which cues the active category token a of the same incomplete category as a given awaited token b; and
- B_t, which cues the awaited category token b of the incomplete category that non-immediately dominates any active category token a.

Together, the cued associations in these continuation matrices trace a path up from the most subordinate awaited category token b to the most superordinate category token currently hypothesized as the root of the syntactic tree. Vectors for category types c can then be cued from any category token a or b through an associative matrix C_t. All three of these matrices may be updated from time step to time step by associating cue and target vectors through outer product addition, as described above.

The model also defines vectors for binary-branching grammar rules g, which are associated with parent, left child, or right child category types via 'accessor' matrices G, G', or G''.[4] These accessor matrices are populated from binary-branching rules in a probabilistic context-free grammar (PCFG) in Chomsky Normal Form (CNF). For example, the PCFG rule $\mathsf{P}(\mathsf{S} \rightarrow \mathsf{NP\ VP}) = 0.8$ may be encoded using a

[3]A diagonal product $\operatorname{diag}(v)\, u$ defines a vector by multiplying corresponding scalars in vectors v and u:

$$(\operatorname{diag}(v)\, u)_{[i]} \overset{\text{def}}{=} v_{[i]} \cdot u_{[i]} \tag{3$'$}$$

[4]This use of reification and accessor matrices for grammar rules emulates a tensor model (Smolensky, 1990; beim Graben et al., 2008) in that in the worst case grammar rules (composed of multiple categories) would require a space polynomially larger than that of category types, but since this space is sparsely inhabited in the expected case, this reified representation is computationally more tractable.

grammar rule vector $g_{\text{S} \rightarrow \text{NP VP}}$ and category vectors $c_{\text{S}}, c_{\text{VP}}, c_{\text{NP}}$ with the following outer-product associations:

$$G \overset{\text{def}}{=} g_{\text{S} \rightarrow \text{NP VP}} \otimes c_{\text{S}} \cdot 0.8$$
$$G' \overset{\text{def}}{=} g_{\text{S} \rightarrow \text{NP VP}} \otimes c_{\text{NP}}$$
$$G'' \overset{\text{def}}{=} g_{\text{S} \rightarrow \text{NP VP}} \otimes c_{\text{VP}}$$

Grammars with additional rules can then be encoded as a sum of outer products of rule and category vectors. Grammar rules can then be cued from category types by matrix multiplication, e.g.:

$$g_{\text{S} \rightarrow \text{NP VP}} = G' \, c_{\text{NP}}$$

and category types can be cued from grammar rules using transposed versions of accessor matrices:

$$c_{\text{NP}} = G'^{\top} g_{\text{S} \rightarrow \text{NP VP}}$$

The model also defines:

- vectors x_t for observation types (i.e. words),
- a matrix P cueing category types from observation types, populated from unary rules in a CNF PCFG, and
- a matrix $D = D_K$ of leftmost descendant categories cued from ancestor categories, derived from accessor matrices G and G' by K iterations of the following recurrence:[5]

$$D'_0 \overset{\text{def}}{=} \text{diag}(\mathbf{1}) \quad (5)$$
$$D_0 \overset{\text{def}}{=} \text{diag}(\mathbf{0}) \quad (6)$$
$$D'_k \overset{\text{def}}{=} G'^{\top} \, G \, D'_{k\text{-}1} \quad (7)$$
$$D_k \overset{\text{def}}{=} D_{k\text{-}1} + D'_k \quad (8)$$

  where each D'_k cues a probabilistically-weighted descendant at distance k from its cue, and D_k is the superposition of all such descendant associations from length 1 to length K. This produces a superposed set of category types that may occur as leftmost descendants of a (possibly superposed) ancestor category type.

In order to exclude active category types $C_t \, a_t$ that are not compatible with awaited category types $C_t \, b_t$ in the same incomplete category, the model also defines:

- a matrix $E = E_K$ of rightmost descendant categories cued from ancestor categories, derived in the same manner as D, except using G'' in place of G'.

The parser proceeds in two phases, generating a complete category token vector a''_t from $b_{t\text{-}1}$ during the F phase, then generating an awaited category token vector b_t of an incomplete category during the J phase. Since the parser proceeds in two phases, this paper will distinguish variables updated in each phase using a subscript for time step $t-.5$ at the end of the first phase and t at the end of the second phase.

The vectorial parser implements the F phase of the left-corner parser (the 'fork/no-fork' decision) by first defining two new category tokens for the possibly forked or unforked complete category:

$$a_{t\text{-}.5}, \; a'_{t\text{-}.5} \sim \text{Exp}$$

The parser then obtains:

- the category type of the most subordinate awaited category token at the previous time step: $C_{t\text{-}1} \, b_{t\text{-}1}$ (which involves no fork), and
- a superposed set of non-immediate descendants of the category type of this most subordinate awaited category token: $D \, C_{t\text{-}1} \, b_{t\text{-}1}$ (which involves a fork),

These fork and no-fork categories are then diagonally multiplied (intersected) with a superposed set of preterminal categories for the current observation ($P \, x_t$):

$$c^-_t = \text{diag}(P \, x_t) \, C_{t\text{-}1} \, b_{t\text{-}1}$$
$$c^+_t = \text{diag}(P \, x_t) \, D \, C_{t\text{-}1} \, b_{t\text{-}1}$$

The B and C continuation and category matrices are then updated with a superordinate awaited category token and category type for a and a':

$$a_{t\text{-}1} = A_{t\text{-}1} \, b_{t\text{-}1}$$
$$B_{t\text{-}.5} = B_{t\text{-}1} + b_{t\text{-}1} \otimes a'_{t\text{-}.5} + B_{t\text{-}1} \, a_{t\text{-}1} \otimes a_{t\text{-}.5}$$
$$C_{t\text{-}.5} = C_{t\text{-}1} + c^+_t \otimes a'_{t\text{-}.5} + \text{diag}(C_{t\text{-}1} \, a_{t\text{-}1}) \, E^{\top} c^-_t \otimes a_{t\text{-}.5}$$

where the updated category for $a_{t\text{-}.5}$ results from an intersection (diagonal product) of the current category at $a_{t\text{-}1}$ with the set of categories that can occur with c^-_t as a rightmost child, as defined by E. The intersected fork and no-fork category types are then used to weight superposed hypotheses for

[5]Here $\mathbf{1}$ and $\mathbf{0}$ denote vectors of ones and zeros, respectively.

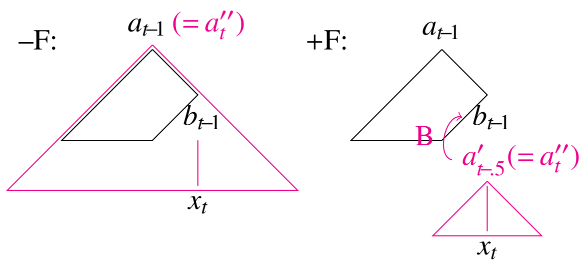


Figure 5: Updates to continuation matrices during the 'fork' phase of a left-corner parser.

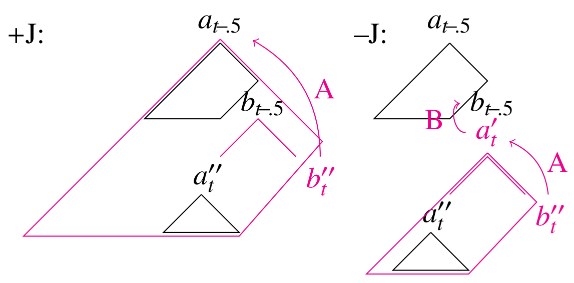


Figure 6: Updates to continuation matrices during the 'join' phase of a left-corner parser.

the complete category token a''_t that will result from this phase of processing, and the b vector is updated to encode the category token:[6]

$$a''_t = \frac{a_{t\text{-}.5}\,||c^-_t|| + a'_{t\text{-}.5}\,||c^+_t||}{||a_{t\text{-}.5}\,||c^-_t|| + a'_{t\text{-}.5}\,||c^+_t||||}$$
$$b_{t\text{-}.5} = B_{t\text{-}.5}\,a''_t$$

These updates can be represented graphically as shown in Figure 5.

The vectorial parser then similarly implements the J phase (the 'join/no-join' decision) of the left-corner parser by first defining a new category token a' for a possible new active category of the most subordinate incomplete category, and b'' for a new awaited category token:

$$a'_t,\ b''_t \sim \text{Exp}$$

The parser then obtains:

- a superposed set of grammar rules with parent category matching the category of the most subordinate awaited category token at the previous time step: $G\,C_{t\text{-}.5}\,b_{t\text{-}.5}$ (which assumes a join), and
- a superposed set of grammar rules with parent category non-immediately descended from the category of this most subordinate awaited category token: $G\,D\,C_{t\text{-}.5}\,b_{t\text{-}.5}$ (which assumes no join)

These join and no-join grammar rule vectors are then diagonally multiplied (intersected) with the superposed set of grammar rules whose left child category type matches the category type of the most subordinate complete category token ($G'\,C_{t\text{-}.5}\,a''_t$):

$$g^+_t = \text{diag}(G'\,C_{t\text{-}.5}\,a''_t)\,G\,C_{t\text{-}.5}\,b_{t\text{-}.5}$$
$$g^-_t = \text{diag}(G'\,C_{t\text{-}.5}\,a''_t)\,G\,D\,C_{t\text{-}.5}\,b_{t\text{-}.5}$$

These intersected join and no-join grammar rule vectors are then used to weight superposed hypotheses for the incomplete category that will result from this phase of processing in updates to the continuation and category matrices A, B, and C:

$$A_t = A_{t\text{-}1} + \frac{A_{t\text{-}1}\,b_{t\text{-}.5}\,||g^+_t|| + a'_t\,||g^-_t||}{||A_{t\text{-}1}\,b_{t\text{-}.5}\,||g^+_t|| + a'_t\,||g^-_t||||} \otimes b''_t$$
$$B_t = B_{t\text{-}.5} + b_{t\text{-}.5} \otimes a'_t$$
$$C_t = C_{t\text{-}.5} + G^\top g^-_t \otimes a'_t + \frac{G''^\top g^+_t + G''^\top g^-_t}{||G''^\top g^+_t + G''^\top g^-_t||} \otimes b''_t$$

These updates can be represented graphically as shown in Figure 6. Finally the the most subordinate awaited category token is updated for the next word:

$$b_t = b''_t$$

5 Predictions

In order to assess the cognitive plausibility of the memory modeling assumptions in this vectorial parser, predictions of the implementation defined in Section 4 were calculated on center-embedding and right-branching sentences, exemplified by:

(1) *If either Kim stays or Kim leaves then Pat leaves.* (center-embedded condition)

(2) *If Kim stays then if Kim leaves then Pat leaves.* (right-branching condition)

[6]This uses the two norm $||v||$, which is the magnitude of vector v, defined as the square root of the sum of the squares of its scalar values:

$$||v|| \overset{\text{def}}{=} \sqrt{\textstyle\sum_i (v_{[i]})^2} \qquad (4')$$

Dividing a vector by its two norm has the effect of normalizing it to unit length.

$$
\begin{aligned}
\mathsf{P}(\text{T} \rightarrow \text{S T}) &= 1.0\\
\mathsf{P}(\text{S} \rightarrow \text{NP VP}) &= 0.5\\
\mathsf{P}(\text{S} \rightarrow \text{IF S THEN S}) &= 0.25\\
\mathsf{P}(\text{S} \rightarrow \text{EITHER S OR S}) &= 0.25\\
\mathsf{P}(\text{IF} \rightarrow \text{if}) &= 1.0\\
\mathsf{P}(\text{THEN} \rightarrow \text{then}) &= 1.0\\
\mathsf{P}(\text{EITHER} \rightarrow \text{either}) &= 1.0\\
\mathsf{P}(\text{OR} \rightarrow \text{or}) &= 1.0\\
\mathsf{P}(\text{NP} \rightarrow \text{kim}) &= 0.5\\
\mathsf{P}(\text{NP} \rightarrow \text{pat}) &= 0.5\\
\mathsf{P}(\text{VP} \rightarrow \text{leaves}) &= 0.5\\
\mathsf{P}(\text{VP} \rightarrow \text{stays}) &= 0.5
\end{aligned}
$$

Figure 7: 'If . . . then . . . ' grammar used in sentence processing experiment. Branches with arity greater than two are decomposed into equivalent right-branching sequences of binary branches.

both of which contain the same number of words. These sentences were processed using the grammar shown in Figure 7, which assigns the same probability to both center-embedding and right-branching sentences. The *if . . . then . . .* and *either . . . or . . .* constructions used in these examples are taken from the original Chomsky and Miller (1963) paper introducing center-embedding effects, and are interesting because they do not involve the same grammar rule (as is the case with familiar nested object relative constructions), and do not involve filler-gap constructions, which may introduce overhead processing costs as a possible confound.

This assessment consisted of 500 trials for each sentence type. Sentences were input to an implementation of this model using the Numpy package in Python, which consists of the equations shown in Section 4 enclosed in a loop over the words in each sentence. Each trial initially sampled a, b, c, and g vectors from random exponential distributions of dimension 100, and the parser initialized b_0 with category type T as shown in Figure 3, with the active category token at $A_0\, b_0$ also associated with category type T.

Accuracy for this assessment was calculated by finding the category type with the maximum cosine similarity for the awaited category b_T at the end of the sentence. If this category type was T (as it is in Figure 3), the parser was awarded a point of accuracy; otherwise it was not. The results of this assessment are shown in Table 1. The parser processes sentences with right-branching structure substantially more accurately than sentences with center-embedded structure. These results are strongly significant ($p < .001$) using a χ^2 test.

sentence	correct	incorrect
center-embedded	231	269
right-branching	297	203

Table 1: Accuracy of vectorial parser on each sentence type.

These predictions seem to be consistent with observations by Chomsky and Miller (1963) that center-embedded structures are more difficult to parse than right-branching structures, but it is also important to note how the model arrives at these predictions. The decreased accuracy of center-embedded sentences is not a result of an explicit decay factor, as in ACT-R and other models (Lewis and Vasishth, 2005), or distance measures as in DLT (Gibson, 2000), nor is it attributable to cue interference (as modeled by Lewis and Vasishth for nested object relative constructions), since the inner and outer embeddings in these sentences use different grammar rules. The decreased accuracy for center-embedding is also not attributable to frequency effects of grammar rules (as modeled by Hale, 2001), since the rules in this grammar are relatively common and equally weighted.

Instead, the decrease for center-embedded structures emerges from this model as a necessary result of drift due to repeated superposition of targets encoded in continuation matrices A and B. This produces a natural decay over time as sequences of subordinate category token vectors b_t introduce noise in updates to A_t and B_t. When these matrices are cued in concert, as happens when cueing across incomplete categories, the distortion is magnified. This decay is therefore a consequence of encoding hierarchic structural information using cued associations. In contrast, right-branching parses are not similarly as badly degraded over time because the flat treatment of left- and right- branching structures in a left-corner parser does not cue as often across incomplete categories using matrix B.

6 Extensions

This model is also interesting because it allows semantic relations to be constructed using the same outer product associations used to define continuation and category matrices in Section 4. First, discourse referent instances and numbered relation types are defined as vectors i and n, respectively. Then relation tokens are reified as vectors r, similar to the reification of grammar rules described in Section 4, and connected to relation type vectors n by cued association R and to source and target discourse referents i by cued associations R' and R''. Semantic relation types can then be cued from grammar rules g using associative matrix N, allowing relations of various types to be constructed in cases of superposed grammar rules. In future work, it would be interesting to see whether this representation is consistent with observations of local syntactic coherence (Tabor et al., 2004).

This model can also constrain relations to discourse referents introduced in a previous sentence or earlier in the same sentence using a vector of *temporal features* (Howard and Kahana, 2002). This is a vector of features z_t, that has a randomly chosen selection of features randomly resampled at each time step, exponentially decreasing the cosine similarity of the current version of the z_t vector to earlier versions $z_{t'}$. If discourse referents i are cued from the current temporal features z_t in an outer product associative matrix Z, it will cue relatively recently mentioned discourse referents more strongly than less recently mentioned referents. If discourse referents for eventualities and propositions j are connected to explicit predicate type referents k (say, cued by a relation of type '0'), and if temporal cues are combined in a diagonal product with cues by semantic relations from a common predicate type, the search for a consistent discourse referent can be further constrained to match the gender of a pronoun or other relations from a definite reference. In future work, it would be interesting to compare the predictions of this kind of model to human coreference resolution, particularly in the case of parsing conjunctions with reflexive pronouns, which has been used to argue for fully connected incremental parsing (Sturt and Lombardo, 2005).

7 Conclusion

This paper has presented a vectorial left-corner parsing model defined using independently posited operations over activation-based working memory and weight-based episodic memory. This model has the attractive property that it hypothesizes only one unary branching rule application and only one binary branching rule application per time step, which allows it to be smoothly integrated into a vector-based recurrence that propagates structural ambiguity from one time step to the next. Predictions of this model were calculated on a center-embedded sentence processing task and the model was shown to exhibit decreased processing accuracy in center-embedded constructions, as observed by Chomsky and Miller (1963), even in the absence of repeated grammar rules or potential confounding overhead costs that may be associated with filler-gap constructions.

This model is particularly interesting because, unlike other vectorial or connectionist parsers, it directly implements a recursive probabilistic grammar with explicit categories of syntactic context. This explicit implementation of a probabilistic grammar allows variations of this processing model to be evaluated without having to also posit a human-like model of acquisition. For example, the model can simply be defined with a PCFG derived from a syntactically annotated corpus.

The model is also interesting because it serves as an existence proof that recursive grammar is not incompatible with current models of human memory.

Finally, the fact that this model predicts memory effects at boundaries between incomplete categories, in line with predictions of fully parallel left-corner parsers (van Schijndel and Schuler, 2013; van Schijndel et al., 2013b), suggests that measures based on incomplete categories (or based on connected components of other kinds of syntactic or semantic structure) are not simply arbitrary but rather may naturally emerge from the use of associative memory during sentence processing.

Although the model may not scale to broad-coverage parsing evaluations in its present form, future work will explore hybridization of some of these methods into a parser with an explicit beam of parallel hypotheses. It is anticipated that an algorithmic-level comprehension model such as this will allow a more nuanced understanding of human semantic representation and grammar acquisition.

References

James A. Anderson, Jack W. Silverstein, Stephen A. Ritz, and Randall S. Jones. 1977. Distinctive features, categorical perception and probability learning: Some applications of a neural model. *Psychological Review*, 84:413–451.

Peter beim Graben, Sabrina Gerth, and Shravan Vasishth. 2008. Towards dynamical system models of language-related brain potentials. *Cognitive Neurodynamics*, 2(3):229–255.

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

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

Edward Gibson. 2000. The dependency locality theory: A distance-based theory of linguistic complexity. In *Image, language, brain: Papers from the first mind articulation project symposium*, pages 95–126, Cambridge, MA. MIT Press.

John Hale. 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.

Marc W. Howard and Michael J. Kahana. 2002. A distributed representation of temporal context. *Journal of Mathematical Psychology*, 45:269–299.

Roger Levy and Edward Gibson. 2013. Surprisal, the pdc, and the primary locus of processing difficulty in relative clauses. *Frontiers in Psychology*, 4(229).

Roger Levy. 2008. Expectation-based syntactic comprehension. *Cognition*, 106(3):1126–1177.

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

David Marr. 1971. Simple memory: A theory for archicortex. *Philosophical Transactions of the Royal Society (London) B*, 262:23–81.

J. L. McClelland, B. L. McNaughton, and R. C. O'Reilly. 1995. Why there are complementary learning systems in the hippocampus and neocortex: Insights from the successes and failures of connectionist models of learning and memory. *Psychological Review*, 102:419–457.

B.B. Murdock. 1982. A theory for the storage and retrieval of item and associative information. *Psychological Review*, 89:609–626.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Proceedings of NAACL HLT 2007*, pages 404–411, Rochester, New York, April. Association for Computational Linguistics.

Paul Smolensky. 1990. Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial intelligence*, 46(1-2):159–216.

Patrick Sturt and Vincent Lombardo. 2005. Processing coordinate structures: Incrementality and connectedness. *Cognitive Science*, 29:291–305.

W. Tabor, B. Galantucci, and D Richardson. 2004. Effects of merely local syntactic coherence on sentence processing. *Journal of Memory and Language*, 50(4):355–370.

Marten van Schijndel and William Schuler. 2013. An analysis of frequency- and recency-based processing costs. In *Proceedings of NAACL-HLT 2013*. Association for Computational Linguistics.

Marten van Schijndel, Andy Exley, and William Schuler. 2013a. A model of language processing as hierarchic sequential prediction. *Topics in Cognitive Science*, 5(3):522–540.

Marten van Schijndel, Luan Nguyen, and William Schuler. 2013b. An analysis of memory-based processing costs using incremental deep syntactic dependency parsing. In *Proceedings of CMCL 2013*. Association for Computational Linguistics.