Memory-bounded Neural Incremental Parsing for Psycholinguistic Prediction

Lifeng Jin and **William Schuler**
Department of Linguistics
The Ohio State University, Columbus, OH, USA
{jin, schuler}@ling.osu.edu

Abstract

Syntactic surprisal has been shown to have an effect on human sentence processing, and can be calculated from prefix probabilities of generative incremental parsers. Recent state-of-the-art incremental generative neural parsers are able to produce accurate parses and surprisal values, but have unbounded stack memory, which may be used by the neural parser to maintain explicit in-order representations of all previously parsed words, inconsistent with results of human memory experiments. In contrast, humans seem to have a bounded working memory, demonstrated by inhibited performance on word recall in multi-clause sentences (Bransford and Franks, 1971), and on center-embedded sentences (Miller and Isard, 1964). Bounded statistical parsers exist, but are less accurate than neural parsers in predicting reading times. This paper describes a neural incremental generative parser that is able to provide accurate surprisal estimates and can be constrained to use a bounded stack. Results show that accuracy gains of neural parsers can be reliably extended to psycholinguistic modeling without risk of distortion due to unbounded working memory.

1 Introduction

Syntactic surprisal has been shown to have an effect on human sentence processing, and can be calculated from prefix probabilities of generative incremental parsers (Hale, 2001; Levy, 2008), making it a useful baseline predictor when looking for effects of other factors, like limits of memory or attention. Recent work in generative neural network parsing (Dyer et al., 2016; Hale et al., 2018) has shown that generative parsers based on neural networks are more accurate than earlier statistical generative parsers, and can be used for surprisal calculation. Although a typical shift-reduce neural network parser like that used by Hale et al. (2018) and Crabbé et al. (2019) may be successful in predicting brain imaging data, the depth of its stack memory, the model component where past predicted items are faithfully stored, can be as long as the whole derivational history of the parse (Kuncoro et al., 2018). This potentially sentence-length stack may be used by the neural parser to maintain explicit in-order representations of all previously parsed words. In contrast, humans seem to have a bounded working memory, demonstrated by inhibited performance on word recall in multi-clause sentences (Bransford and Franks, 1971), and on center-embedded sentences (Miller and Isard, 1964).[1] Explicit storage of this long parsing history may improve parsing accuracy, but it also risks distorting the predictions of the model when used as a statistical control in psycholinguistic experiments.

Left-corner parsers (Rosenkrantz and Lewis, 1970; Johnson-Laird, 1983) have been argued to provide human-like limits on working memory, because the stack memory requirements of this kind of parser do not grow unboundedly in linguistically common cases of left- or right recursion, only in linguistically rare cases of center recursion. For example, a left corner parser would require only one memory element to process the right recursive sentence, 'The dog chased the cat that ate the rat that nibbled the malt,' but would require three elements to process the center recursive

[1] Specifically, Bransford and Franks (1971) found that subjects were not reliably able to recall word-order information such as whether sentences were in passive or active voice following exposure to sentence stimuli, and Miller and Isard (1964) found that subjects were not able to understand sentences with deeply nested center-embedded structures, such as:

(i) The cart [$_{RC}$ the horse [$_{RC}$ the man bought] pulled] broke.

as easily as non-center-embedded control sentences, despite being composed of familiar rules.

Proceedings of the 16th International Conference on Parsing Technologies and the IWPT 2020 Shared Task, pages 47–60
Virtual Meeting, July 9, 2020. ©2020 Association for Computational Linguistics
https://doi.org/10.18653/v1/P17

sentence, 'The rat that the cat that the dog chased ate nibbled the malt,' consistent with findings that humans have more difficulty understanding the latter sentence. Left-corner parsers also define a fixed set of probabilistic decisions at each word, which naturally paces the surprisal measures produced by the model. Unfortunately, existing left-corner parsers (van Schijndel et al., 2013) are statistical rather than neural, and are therefore substantially less accurate than state-of-the-art neural network parsers.

This paper therefore defines a neural-network left-corner parser with bounded stack memory for parsing and psycholinguistic prediction. Experiments described in this paper show that this generative left-corner neural network parser is competitive with incremental generative parsers that use unbounded stack memory in a parsing task, and outperforms statistical memory-bounded generative left-corner parsers both in parsing accuracy and in fitting human behavioral data on two different datasets, consistent with the conclusion that accuracy gains of neural parsers can be reliably extended to psycholinguistic modeling without risk of distortion due to unbounded working memory.

2 Related work

Incremental generative constituent parsers are able to process sentences in time order and provide psycholinguistically predictive measures like syntactic surprisal and entropy reduction (Levy, 2008; Hale, 2001, 2006), which in turn are used in psycholinguistic experiments for probing effects of syntax on behavioral data (Demberg and Keller, 2008; Demberg et al., 2012; van Schijndel and Schuler, 2015). Statistical incremental parsers like ones proposed by Roark (2001) and van Schijndel et al. (2013) are based on context-free grammars. The Roark (2001) parser builds syntactic structures top-down incrementally and has been used in studies for calculating surprisal (Demberg and Keller, 2008; Roark et al., 2009; Frank, 2009). Left-corner parsers (Rosenkrantz and Lewis, 1970; Johnson-Laird, 1983) are often used to model limits on center embedding (Abney and Johnson, 1991; Gibson, 1991; Resnik, 1992; Stabler, 1994; Lewis and Vasishth, 2005). van Schijndel et al. (2013) proposed an incremental parser that takes working memory constraints into account, and is able to produce probabilistic measures as well as predictions about working memory operations (van Schijndel and Schuler, 2015). Demberg et al. (2013) propose a parser which is also able to produce prefix probabilities for tree-adjoining grammars. All of these statistical parsers lag behind state-of-the-art parsers in parsing accuracy, because of psycholinguistic constraints like incrementality and because they use less expressive statistical models.

State-of-the-art constituency parsers generally are neural network models (Choe and Charniak, 2016; Dyer et al., 2016; Kitaev and Klein, 2018). Dyer et al. (2016) propose a generative neural model for top-down incremental parsing but use it only as a reranker for a discriminative parser. Extensions to the Dyer et al. (2016) model allow the parser to do in-order tree traversal (Liu and Zhang, 2017; Kuncoro et al., 2018).[2] However, the in-order transition system has a bias towards left children of constituents, which is not desirable when the model is used to calculate prefix probabilities. This issue was addressed by using word-synchronous beam search (Stern et al., 2017; Hale et al., 2018) or variable sized beam search (Crabbé et al., 2019) and successfully predict brain imaging data. However, all of these parsers do not limit the number of stack elements the parser has direct access to at any timestep, which in some cases can be equal to number of derivational decisions made up to the current timestep. This behavior of unbounded stack does not match what we know about human working memory and is undesirable for calculating predictors like probabilistically-weighted emdedding depth (Wu et al., 2010). The model described in this paper avoids these problems by using a left-corner transition system, which uses a bounded pushdown store and a fixed set of probabilistic decisions per word. The bounded stack memory not only more closely implements human working memory limits in a model designed to calculate cognitive predictors, other work (Jin et al., 2018) shows that it also helps limit search space for unsupervised grammar acquisition.

[2] Kuncoro et al. (2018) calls the in-order tree traversal a left-corner traversal. In order to avoid confusion, we refer to it as in-order tree traversal, and save the name left-corner traversal for the Johnson-Laird (1983) formulation, which traverses trees from left to right with a bounded stack.

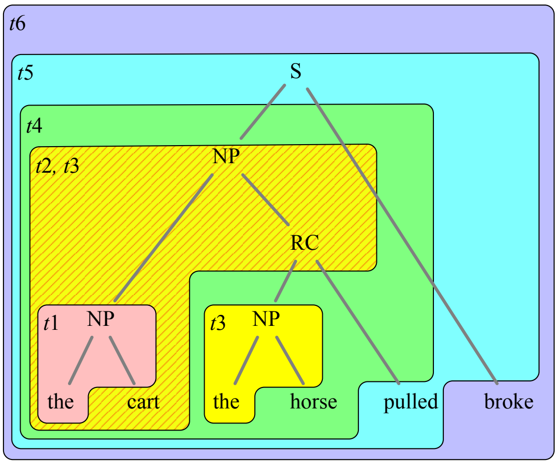


Figure 1: Example of a parse tree for the sentence *the cart the horse pulled broke*. Time step indices are marked inside the derivation fragment built in that time step. A and B nodes are the top and the rightmost node of each derivation fragment. Note that *t*3 has two separate derivation fragments, which indicates that at *t*3 there are two sets of A and B nodes on the stack. At each timestep, A and B nodes of the lowest derivation fragment will be predicted. If they have already been predicted or they do not exist, then the decision will be NULL. POS tags for terminals are omitted.

Step	Stack	Decision
1	[ ] [the] [NP/NP ⊤]	GENERATE-the PJA-NP PJB-NULL
2	[NP/NP ⊥] [NP] [NP/NP ⊤]	GENERATE′-cart PJA-NP PJB-RC
3	[NP/RC] [NP/RC the] [NP/RC NP/NP ⊤]	GENERATE-the PJA-NP PJB-NULL
4	[NP/RC NP/NP ⊥] [NP/RC NP] [NP/RC ⊤]	GENERATE′-horse PJA-NULL PJB-NULL
5	[NP/RC ⊥] [NP] [S/S ⊤]	GENERATE′-pulled PJA-S PJB-NULL
6	[S/S ⊥] [S] [⊤]	GENERATE′-broke PJA-NULL PJB-T
	[ ]	

Table 1: Parser decisions and stack states for the sentence "*the cart the horse pulled broke*" shown in Figure 1 using the simple left-corner transition system. It shows that the stack grows only when there is a new center embedded clause within the clause being processed.

3 Incremental left corner transition system with bounded working memory

This paper introduces a neural left-corner transition system for incremental constituency parsing with minimal working memory requirements. This system defines a fixed set of parser decisions at each time step. Following these parsing decisions, the parser incrementally generates each word in a sentence and the syntactic structures associated with that word, in time order. Because the parser needs space on the stack only when there is center-embedding in the sentence, this transition system uses much less stack memory than other shift-reduce transition systems (Kuncoro et al., 2018), modeling the psycholinguistic phenomenon that center-embedding is rare and hard for humans to process.

3.1 Types of nodes in left-corner parsing

A left-corner parser maintains a pushdown store of one or more derivation fragments A/B, each of which consists of a top node of category A lacking a bottom node of category B yet to come. The parser generates each word in a sentence at each time step, and then makes predictions about the top nonterminal category of the current derivation fragment and the bottom rightmost unfinished nonterminal category. This process uses stored states only within center-embedded structures, reflecting the difficulty of center-embedding for humans. For example, in processing the sentence *The cart the horse pulled broke* (see Figure 1), in timestep *t*2 immediately after the word *cart* is generated, the derivation fragment is NP/RC, shown in the figure with an orange-yellow striped plate. The top nonterminal category, or the A category, of this derivation fragment is NP and the bottom rightmost unfinished nonterminal category, or the B category, is RC. At *t*3 when a center-embedded structure appears, a new derivation fragment is created and stored in the stack memory, making *t*3 a timestep with two derivation fragments: NP/RC and NP/NP.

3.2 Parser decisions

Figure 2 defines the set of parser decisions that the parser must make at each time step. They consist of the following:

GENERATE: First a word must be generated given the current state of the parser and pushed onto the stack. There are two rules associated with GENERATE decisions, and they have different stack configurations when the push operation happens. If the stack has a derivation fragment X/Y at its head, then the word is pushed onto the top of the stack without further operation (GENERATE-W). If the top of the stack has a fragment followed by a $\bot$ sign, then the word is first merged with the bottom node Y and then the merged Y node is merged with X. In the end only the top node X remains (GENERATE′-W). The parser deterministically decides which rule to use based on the state of the stack.

PJA : Next a nonterminal top node must be projected onto the head of the stack. The set of possible top nodes include all the nonterminal categories in the training data X as well as a special category NULL. The PJA-X decision projects a nonterminal top node X together with a placeholding bottom node with the same category onto the stack, and appends the stack with a $\top$ sign. PJA-NULL merges the final node Y on the stack, which is often a terminal, with the closest bottom node, and appends the $\top$ sign to the stack.

PJB : Finally a nonterminal bottom node must be projected onto the head of the stack. The set of possible bottom nodes includes all the nonterminal categories in the training data X as well as a special category NULL and discourse level category T. The PJB-X decision merges the last bottom node Y to the bottom node with the predicted category X. PJA-NULL changes the $\top$ sign to the $\bot$ sign at the head of stack.

Table 1 shows how the sentence in Figure 1 is parsed with this left-corner transition system. The state of the stack in the parser in the beginning is implicitly [T/T], which represents the top-level discourse structure which has a top node of T and bottom node of T. We omit this initial fragment in the table for brevity. After parsing the whole sentence, the state of the parser will be [ ], again omitting the top level discourse nodes.

The relationship between a parse tree and a sequence of decisions generated by the transition system is bijective. Trees produced with this system are all binary-branching.

3.3 Use of stack memory

Stack memory depth increases only when a left nonterminal child of a right child is generated (Schuler et al., 2010) as a center-embedded structure is generated. In the current transition system, the PJB decision at the previous time step (PJB_{t-1}) and the PJA decision at the current time step (PJA_t) together decide how depth of a parse will change:

- if PJB_{t-1} = NULL and PJA_t = NULL, then the depth of the sentence will decrease by 1.
- if $\text{PJB}_{t-1} \neq$ NULL and $\text{PJA}_t \neq$ NULL, then the depth of the sentence will increase by 1.
- in all other cases, the depth remains the same as the previous time step.

$$\text{Generate-}W \quad \frac{[\,\sigma \cdot X/Y,\ i\,]}{[\,\sigma \cdot X/Y \cdot W,\ i+1\,]}$$

$$\text{Generate}'\text{-}W \quad \frac{[\,\sigma \cdot X/Y \cdot \bot,\ i\,]}{[\,\sigma \cdot X,\ i+1\,]}$$

$$\text{PJA-}X \quad \frac{[\,\sigma \cdot Y,\ i\,]}{[\,\sigma \cdot X/X \cdot \top,\ i\,]}$$

$$\text{PJA-NULL} \quad \frac{[\,\sigma \cdot Y,\ i\,]}{[\,\sigma \cdot \top,\ i\,]}$$

$$\text{PJB-}X \quad \frac{[\,\sigma \cdot Z/Y \cdot \top,\ i\,]}{[\,\sigma \cdot Z/X,\ i\,]}$$

$$\text{PJB-NULL} \quad \frac{[\,\sigma \cdot X/Y \cdot \top,\ i\,]}{[\,\sigma \cdot X/Y \cdot \bot,\ i\,]}$$

Figure 2: The generative incremental left-corner transition system. Adding a buffer to the system yields a discriminative transition system.

The depth of the stack memory for a parse is closely related to the well-formedness of a parse. As Figure 1 shows, a valid parse starts at depth 0, stays at larger depths during parsing the sentence, and returns to depth 0 at the end of the sentence. The figure also shows that the depth of the stack memory only increases when a center embedding is being parsed at $t3$. The average stack memory depth for the transition system in these parsing experiments is 2, which means that on average there are 4 tree nodes in stack memory, much smaller than 12 which is the average number of items for the top-down system used in Hale et al. (2018). This shows that the left-corner transition system makes much more parsimonious use of stack memory than a typical shift-reduce system. The left-corner model evaluated in the experiments also applies bounding to the stack memory and uses a relatively liberal maximum depth of 5 derivation fragments (10 tree nodes), reflecting the fact that remembering more than 10 items faithfully at once is highly unlikely in sentence processing due to working memory limits in humans.

There are two sets of constraints for different use cases for the parser to prune parses on the beam.[3] The basic set only drops a parse when the

[3] Please see the supplemental materials for details.

parse reaches depth 0 before the end of the sentence. This set is used by the parser when psycholinguistic measures are needed. The extended set provides information to the parser about the length of the sentence currently parsing, guiding the parser to drop parses with stack memory too deep or too shallow while parsing. Because this set provides some forward context, it is only used when the parser is used to find best parses in linguistic evaluation.

4 Parsing model

This section defines a memory-bounded neural incremental generative parser as a generative probability model for surprisal calculation using the proposed left-corner transition system. In the description below, all LSTMs are stack-LSTMs (Dyer et al., 2016) with coupled input and forget gates (Greff et al., 2017) and all FFs are feed-forward neural networks.

Surprisal at a word w_t is defined as the negative log of the probability of that word given its preceding words $w_{1..t-1}$ under some model θ. This can be computed by marginalizing over the final hidden state of a sequence:

$$-\log \mathsf{P}_\theta(w_t \,|\, w_{1..t-1}) = -\log \frac{\sum_{q_t} \mathsf{P}_\theta(q_t\, w_{1..t})}{\sum_{q_t w_t} \mathsf{P}_\theta(q_t\, w_{1..t})} \tag{1}$$

then decomposing the marginalized term into a recurrence of marginalized transition-observation probabilities:

$$\mathsf{P}_\theta(q_t\, w_{1..t}) = \sum_{q_{t-1}} \mathsf{P}_\theta(w_t\, q_t \,|\, q_{t-1}) \cdot \mathsf{P}_\theta(q_{t-1}\, w_{1..t-1}) \tag{2}$$

using $\mathsf{P}_\theta(q_0\, w_0) = 1$ for some start-of-sentence word w_0 and initial state q_0.

The hidden states q_t of the model described in this paper consist of:

- cell and hidden vectors $\mathbf{c}_t, \mathbf{h}_t \in \mathbb{R}^n$ for a word LSTM,
- a preterminal decision $p_t \in C$ over category labels C,
- a top decision $a_t \in C \cup \{\perp\}$ over labels and null results $\perp$,
- a bottom decision $b_t \in C \cup \{\perp\}$ over labels and null results,
- a top vector $\mathbf{a}_t^d \in \mathbb{R}^n$ for each depth $d \in \{1..D\}$,
- a bottom vector $\mathbf{b}_t^d \in \mathbb{R}^n$ for each depth $d \in \{1..D\}$, and
- cell and hidden vectors $\mathbf{c}'_t, \mathbf{h}'_t \in \mathbb{R}^n$ for a decision LSTM.

Probabilities for observing a word and transitioning to a new hidden state at each time step t, given a hidden state at the previous time step $t-1$, can then be decomposed into terms for each individual decision and resulting vector:

$$\begin{aligned}
&\mathsf{P}(w_t\, \mathbf{c}_t\, \mathbf{h}_t\, p_t\, a_t\, b_t\, \mathbf{a}_t^{1..D}\, \mathbf{b}_t^{1..D}\, \mathbf{c}'_t\, \mathbf{h}'_t | \mathbf{c}_{t-1}\, \mathbf{h}_{t-1}\, \mathbf{a}_{t-1}^{1..D}\, \mathbf{b}_{t-1}^{1..D}\, \mathbf{c}'_{t-1}\, \mathbf{h}'_{t-1}) \\
&= \mathsf{P}(w_t | \mathbf{c}_{t-1}\, \mathbf{h}_{t-1}\, \mathbf{a}_{t-1}^{1..D}\, \mathbf{b}_{t-1}^{1..D}\, \mathbf{c}'_{t-1}\, \mathbf{h}'_{t-1}) \\
&\cdot \mathsf{P}(\mathbf{c}_t\, \mathbf{h}_t | \mathbf{c}_{t-1}\, \mathbf{h}_{t-1}\, \mathbf{a}_{t-1}^{1..D}\, \mathbf{b}_{t-1}^{1..D}\, \mathbf{c}'_{t-1}\, \mathbf{h}'_{t-1}\, w_t) \\
&\cdot \mathsf{P}(p_t | \mathbf{c}_{t-1}\, \mathbf{h}_{t-1}\, \mathbf{a}_{t-1}^{1..D}\, \mathbf{b}_{t-1}^{1..D}\, \mathbf{c}'_{t-1}\, \mathbf{h}'_{t-1}\, w_t\, \mathbf{c}_t\, \mathbf{h}_t) \\
&\cdot \mathsf{P}(a_t | \mathbf{c}_{t-1}\, \mathbf{h}_{t-1}\, \mathbf{a}_{t-1}^{1..D}\, \mathbf{b}_{t-1}^{1..D}\, \mathbf{c}'_{t-1}\, \mathbf{h}'_{t-1}\, w_t\, \mathbf{c}_t\, \mathbf{h}_t\, p_t) \\
&\cdot \mathsf{P}(b_t | \mathbf{c}_{t-1}\, \mathbf{h}_{t-1}\, \mathbf{a}_{t-1}^{1..D}\, \mathbf{b}_{t-1}^{1..D}\, \mathbf{c}'_{t-1}\, \mathbf{h}'_{t-1}\, w_t\, \mathbf{c}_t\, \mathbf{h}_t\, p_t\, a_t) \\
&\cdot \mathsf{P}(\mathbf{a}_t^{1..D} | \mathbf{c}_{t-1}\, \mathbf{h}_{t-1}\, \mathbf{a}_{t-1}^{1..D}\, \mathbf{b}_{t-1}^{1..D}\, \mathbf{c}'_{t-1}\, \mathbf{h}'_{t-1}\, w_t\, \mathbf{c}_t\, \mathbf{h}_t\, p_t\, a_t\, b_t) \\
&\cdot \mathsf{P}(\mathbf{b}_t^{1..D} | \mathbf{c}_{t-1}\, \mathbf{h}_{t-1}\, \mathbf{a}_{t-1}^{1..D}\, \mathbf{b}_{t-1}^{1..D}\, \mathbf{c}'_{t-1}\, \mathbf{h}'_{t-1}\, w_t\, \mathbf{c}_t\, \mathbf{h}_t\, p_t\, a_t\, b_t\, \mathbf{a}_t^{1..D}) \\
&\cdot \mathsf{P}(\mathbf{c}'_t\, \mathbf{h}'_t | \mathbf{c}_{t-1}\, \mathbf{h}_{t-1}\, \mathbf{a}_{t-1}^{1..D}\, \mathbf{b}_{t-1}^{1..D}\, \mathbf{c}'_{t-1}\, \mathbf{h}'_{t-1}\, w_t\, \mathbf{c}_t\, \mathbf{h}_t\, p_t\, a_t\, b_t\, \mathbf{a}_t^{1..D}\, \mathbf{b}_t^{1..D})
\end{aligned} \tag{3}$$

The probability of observing a word depends on bounded representations of the store, the decision sequence and the word sequence:

$$\begin{aligned}
&\mathsf{P}(w_t \mid \mathbf{a}_{t-1}^{1..D}\, \mathbf{b}_{t-1}^{1..D}\, \mathbf{c}'_{t-1}\, \mathbf{h}'_{t-1}\, \mathbf{c}_{t-1}\, \mathbf{h}_{t-1}) = \\
&\qquad \delta_{w_t}{}^{\top} \textsc{SoftMax}(\, \mathrm{FF}_{\theta_\mathrm{W}} [\, \mathbf{q}_{t-1}, \mathbf{h}'_{t-1}, \mathbf{h}_{t-1} \,]\,)
\end{aligned} \tag{4}$$

where δ_i is a Kronecker delta vector, consisting of a one at element i and zeros elsewhere, and $\mathbf{q}_t$ is a summary of the current stack:

$$\mathbf{q}_t = \mathrm{LSTM}_{\theta_\mathrm{Q}} [\, \mathbf{a}_t^1, \mathbf{b}_t^1, \ldots, \mathbf{a}_t^D, \mathbf{b}_t^D \,]. \tag{5}$$

This probability term defines a distribution over GENERATE decisions.

The probability of a cell and hidden vector of the word LSTM is deterministic given the preceding operations, and is modeled as an indicator equal to one when the vectors are as defined by the corresponding LSTM model, zero otherwise:

$$\begin{aligned}
&\mathsf{P}(\mathbf{c}_t\, \mathbf{h}_t \mid \mathbf{a}_{t-1}^{1..D}\, \mathbf{b}_{t-1}^{1..D}\, \mathbf{c}'_{t-1}\, \mathbf{h}'_{t-1}\, \mathbf{c}_{t-1}\, \mathbf{h}_{t-1}\, w_t) = \\
&\qquad [\![\mathbf{c}_t, \mathbf{h}_t = \mathrm{LSTM}_{\theta_\mathrm{H}} [\, \mathbf{c}_{t-1}, \mathbf{h}_{t-1}; \mathbf{w}_t \,]]\!],
\end{aligned} \tag{6}$$

where:

$$\mathbf{w}_t {=} \mathrm{FF}_{\theta_{\mathrm{W}'}} [\, \mathbf{e}_t, \mathbf{e}'_t, \mathbf{E}'\, \textsc{SoftMax}(\, \mathrm{FF}_{\theta_{\mathrm{W}''}} [\, \mathbf{q}_{t-1}, \mathbf{h}_{t-1}, \mathbf{h}'_{t-1}, \mathbf{e}_t, \mathbf{e}'_t \,]\,)\,] \tag{7}$$

and $\mathbf{e}_t = \mathbf{E}''\, \delta_{w_t}$ is a trained word embedding, and $\mathbf{e}'_t = \mathbf{E}'''\, \delta_{w_t}$ is a pre-trained word embedding.

Similarly for the cell and hidden states of the decision LSTM:

$$\mathsf{P}(\mathbf{c}'_t\,\mathbf{h}'_t \mid \mathbf{a}^{1..D}_{t-1}\,\mathbf{b}^{1..D}_{t-1}\,\mathbf{c}'_{t-1}\,\mathbf{h}'_{t-1}\,\mathbf{c}_{t-1}\,\mathbf{h}_{t-1}\,w_t) = \\ [\![\mathbf{c}'_t, \mathbf{h}'_t = \mathrm{LSTM}_{\theta_\mathrm{M}}[\,\mathbf{c}'_{t-1}, \mathbf{h}'_{t-1}; \mathbf{m}_{t-1}\,]]\!], \quad (8)$$

where $\mathbf{m}_{t-1}$ is a trainable embedding for the decision made at timestep $t-1$. Note that there are three timesteps for $\mathbf{m}$ corresponding to three decisions, compared to one timestep for $\mathbf{w}$. Figure 3 shows an illustration of how the model works to predict the GENERATE-the decision at timestep 3 in Table 1. In the illustration, the decision LSTM takes all previous decisions $\mathbf{m}_1, \mathbf{m}_2, \ldots,$ and generates a hidden state $\mathbf{h}'_2$ which represent the decision history (Equation 8). The word LSTM takes the words which have already been generated, and produces a hidden state $\mathbf{h}_2$ which represents the word history (Equation 6). The stack composer composes all top and bottom categories on the stack represented by the vectors, and produces the representation of the stack (Equation 5). Finally, all three representations of different kinds of information are processed by the GENERATE feedforward network $\mathrm{FF}_{\theta_\mathrm{W}}$, which makes a prediction about which word is next (Equation 4). Other decisions are made in a similar fashion as shown below.

The probability of a preterminal category decision depends on a bounded representation of the word sequence at the current time step, and a bounded representation of the decision sequence and the store at the previous time step, and the trained and untrained pre-trained word embeddings:

$$\mathsf{P}(p_t \mid \mathbf{c}_{t-1}\,\mathbf{h}_{t-1}\,\mathbf{a}^{1..D}_{t-1}\,\mathbf{b}^{1..D}_{t-1}\,\mathbf{c}'_{t-1}\,\mathbf{h}'_{t-1}\,w_t\,\mathbf{c}_t\,\mathbf{h}_t) = \\ \delta_{p_t}{}^\top \textsc{SoftMax}(\,\mathrm{FF}_{\theta_\mathrm{P}}[\,\mathbf{h}_t, \mathbf{h}'_{t-1}, \mathbf{q}_{t-1}, \mathbf{e}_t, \mathbf{e}'_t\,]\,) \quad (9)$$

This term defines a distribution over preterminal (part of speech) decisions.

The probability of a top category decision depends on a bounded representation of the word sequence at the current time step and a bounded representation of the decision sequence and the store at the previous time step:

$$\mathsf{P}(a_t \mid \mathbf{c}_{t-1}\,\mathbf{h}_{t-1}\,\mathbf{a}^{1..D}_{t-1}\,\mathbf{b}^{1..D}_{t-1}\,\mathbf{c}'_{t-1}\,\mathbf{h}'_{t-1}\,w_t\,\mathbf{c}_t\,\mathbf{h}_t\,p_t) = \\ \delta_{a_t}{}^\top \textsc{SoftMax}(\,\mathrm{FF}_{\theta_\mathrm{A}}[\,\mathbf{h}_t, \mathbf{h}'_t, \mathbf{q}_{t-1}\,]\,) \quad (10)$$

This term defines a distribution over PJA decisions.

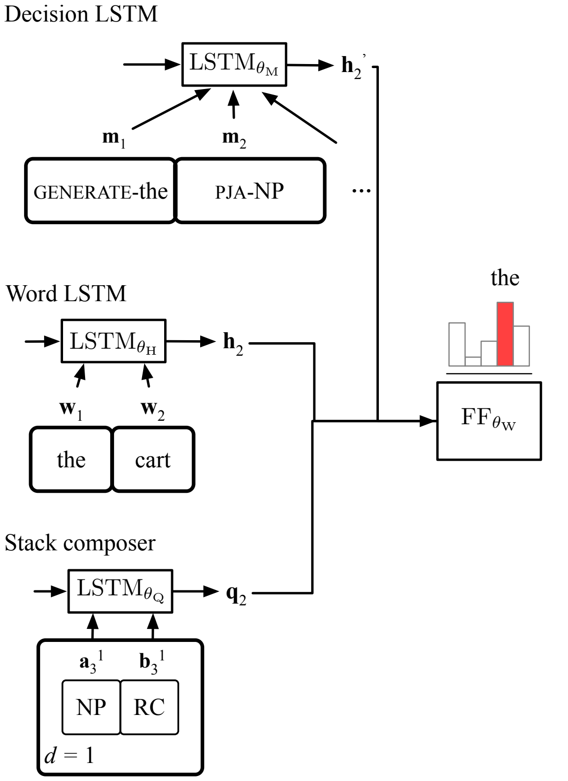


Figure 3: The model makes the prediction for the GENERATE-the decision at timestep 3 in Table 1.

The probability of a bottom category decision depends on a bounded representation of the word sequence at the current time step and decision sequence including the current top decision and the store at the previous time step:

$$\mathsf{P}(b_t \mid \mathbf{c}_{t-1}\,\mathbf{h}_{t-1}\,\mathbf{a}^{1..D}_{t-1}\,\mathbf{b}^{1..D}_{t-1}\,\mathbf{c}'_{t-1}\,\mathbf{h}'_{t-1}\,w_t\,\mathbf{c}_t\,\mathbf{h}_t\,p_t\,a_t) = \\ \delta_{b_t}{}^\top \textsc{SoftMax}(\,\mathrm{FF}_{\theta_\mathrm{B}}[\,\mathbf{h}_t, \mathbf{h}'_{t-.5}, \mathbf{q}_{t-1}\,]\,) \quad (11)$$

where $\mathbf{c}'_{t-.5}, \mathbf{h}'_{t-.5} = \mathrm{LSTM}_{\theta_{H'}}[\,\mathbf{c}'_{t-1}, \mathbf{h}'_{t-1}; \delta_{a_t}\,]$ is the result of adding the bottom decision to the decision LSTM. This term defines a distribution over PJB decisions.

The probability of a top vector is deterministic given the preceding operations, and is modeled as an indicator function equal to one when the vectors are as defined by a set of LSTMs over dependent top and bottom store vectors, depending on the previous bottom category and current top category decisions:

$$\mathsf{P}(\mathbf{a}^{1..D}_t | \mathbf{c}_{t-1}\,\mathbf{h}_{t-1}\,\mathbf{a}^{1..D}_{t-1}\,\mathbf{b}^{1..D}_{t-1}\,\mathbf{c}'_{t-1}\,\mathbf{h}'_{t-1}\,w_t\,\mathbf{c}_t\,\mathbf{h}_t\,p_t\,a_t\,b_t) = \\ \begin{cases} [\![\mathbf{a}^{\bar{d}-1}_t = \mathbf{a}^{\bar{d}-1}_{t-1}]\!] \cdot \phi_{\bar{d}-1} & \text{if } b_{t-1} = \perp, a_t = \perp \\ [\![\mathbf{a}^{\bar{d}}_t = \mathrm{LSTM}_{\theta_{\mathrm{Q}'}}[\,\mathbf{w}_t, \mathbf{a}^{\bar{d}}_{t-1}, \mathbf{E}\,\delta_{a_t}\,]]\!] \cdot \phi_{\bar{d}} & \text{if } b_{t-1} = \perp, a_t \neq \perp \\ [\![\mathbf{a}^{\bar{d}}_t = \mathbf{a}^{\bar{d}}_{t-1}]\!] \cdot \phi_{\bar{d}} & \text{if } b_{t-1} \neq \perp, a_t = \perp \\ [\![\mathbf{a}^{\bar{d}+1}_t = \mathrm{LSTM}_{\theta_{\mathrm{Q}'}}[\,\mathbf{w}_t, \mathbf{E}\,\delta_{a_t}\,]]\!] \cdot \phi_{\bar{d}+1} & \text{if } b_{t-1} \neq \perp, a_t \neq \perp \end{cases} \quad (12)$$

where $\bar{d} = \text{argmax}_d\{\mathbf{a}^d_{t-1} \neq \mathbf{0}\}$ is the previous store depth, and $\phi_d = [\![\mathbf{a}^{1..d-1}_t = \mathbf{a}^{1..d-1}_{t-1}, \mathbf{a}^{d+1..D}_t = \mathbf{0}]\!]$ is a maintenance constraint on stores. These stack operations related to the top category are illustrated in Figure 5 in the appendix.

The probability of a bottom vector is also deterministic and modeled as an indicator function equal to one when the vectors are as defined by a set of LSTMs over dependent top and bottom store vectors, depending on the previous bottom category and current top category decisions:

$$\mathsf{P}(\mathbf{b}^{1..D}_t | \mathbf{c}_{t-1}\; \mathbf{h}_{t-1}\; \mathbf{a}^{1..D}_{t-1}\; \mathbf{b}^{1..D}_{t-1}\; \mathbf{c}'_{t-1}\; \mathbf{h}'_{t-1}\; w_t\; \mathbf{c}_t\; \mathbf{h}_t\; p_t\; a_t\; b_t\; \mathbf{a}^{1..D}_t) =$$
$$\begin{cases} [\![\mathbf{b}^{\bar{d}-1}_t = \text{LSTM}_{\theta_{\text{Q}'}}[\,\mathbf{w}_t, \mathbf{a}^{\bar{d}}_{t-1}, \mathbf{b}^{\bar{d}-1}_{t-1}, \mathbf{E}\,\delta_{b_t}\,]]\!] \cdot \psi_{\bar{d}-1} & \\ & \text{if } b_{t-1}{=}\bot, a_t{=}\bot \\ [\![\mathbf{b}^{\bar{d}}_t = \mathbf{E}\,\delta_{b_t}]\!] \cdot \psi_{\bar{d}} & \text{if } b_{t-1}{=}\bot, a_t{\neq}\bot \\ [\![\mathbf{b}^{\bar{d}}_t = \text{LSTM}_{\theta_{\text{Q}'}}[\,\mathbf{w}_t, \mathbf{b}^{\bar{d}}_{t-1}, \mathbf{E}\,\delta_{b_t}\,]]\!] \cdot \psi_{\bar{d}} & \text{if } b_{t-1}{\neq}\bot, a_t{=}\bot \\ [\![\mathbf{b}^{\bar{d}+1}_t = \mathbf{E}\,\delta_{b_t}]\!] \cdot \psi_{\bar{d}+1} & \text{if } b_{t-1}{\neq}\bot, a_t{\neq}\bot \end{cases} \qquad (13)$$

where $\bar{d} = \text{argmax}_d\{\mathbf{a}^d_{t-1} \neq \mathbf{0}\}$ is the previous store depth, and $\psi_d = [\![\mathbf{b}^{1..d-1}_t = \mathbf{b}^{1..d-1}_{t-1}, \mathbf{b}^{d+1..D}_t = \mathbf{0}]\!]$ is a maintenance constraint on stores.

Finally, the probability of a cell and hidden vector of the decision LSTM is also deterministic and modeled as an indicator equal to one when the vectors are as defined by the corresponding LSTM model, zero otherwise:

$$\mathsf{P}(\mathbf{c}'_t\; \mathbf{h}'_t | \mathbf{c}_{t-1}\; \mathbf{h}_{t-1}\; \mathbf{a}^{1..D}_{t-1}\; \mathbf{b}^{1..D}_{t-1}\; \mathbf{c}'_{t-1}\; \mathbf{h}'_{t-1}\; w_t\; \mathbf{c}_t\; \mathbf{h}_t\; p_t\; a_t\; b_t\; \mathbf{a}^{1..D}_t\; \mathbf{b}^{1..D}_t) =$$
$$[\![\mathbf{c}'_t, \mathbf{h}'_t = \text{LSTM}_{\theta_{\text{H}}}[\,\mathbf{c}'_{t-1}, \mathbf{h}'_{t-1}; \delta_{a_t}, \delta_{b_t}\,]]\!] \qquad (14)$$

4.1 Training and Parsing

The proposed model here is a generative model for sequence prediction with no forward context, therefore ideally it should be trained with a structured training scheme (Weiss et al., 2015). However since it is expensive to search a wide beam in training with a neural network, this model uses a two-stage training scheme. The model is first trained to minimize a cross-entropy loss objective with an l_2 regularization term, defined by:

$$L_\theta(w_{1..T}\; q_{1..T}) = -\log \mathsf{P}_\theta(w_{1..T}\; q_{1..T}) + \frac{\lambda}{2}||\theta||^2 \qquad (15)$$

where $\mathsf{P}_\theta(w_{1..T}\; q_{1..T}) = \prod_t \mathsf{P}_\theta(w_t\; q_t \,|\, q_{t-1})$, and λ is an l_2 regularization strength hyper-parameter.

Training with the local cross-entropy objective quickly leads to overfitting, because the left-corner parsing decisions can be ambiguous at early parts of the sentence, and the objective drives the model to make such decisions perfectly by memorizing the training data. This model therefore stops the cross-entropy training when parsing performance starts to decrease on a development set, and switches to use the REINFORCE algorithm (Williams, 1992; Le and Fokkens, 2017) to fine-tune the model with sequence level supervision. The loss becomes:[4]

$$L'_\theta(w_{1..T}\; q_{1..T}) = \frac{\lambda}{2}||\theta||^2 - \mathsf{E}_{q'_{1..T} \sim \mathsf{P}_\theta(q'_{1..T} \,|\, w_{1..T})}$$
$$(F(q'_{1..T}, q_{1..T}) - \hat{b}) \log \mathsf{P}_\theta(q'_{1..T} \,|\, w_{1..T}) \qquad (16)$$

where $\mathsf{P}_\theta(q'_{1..T} \,|\, w_{1..T}) = \frac{\mathsf{P}_\theta(w_{1..T}\; q'_{1..T})}{\sum_{q_{1..T}} \mathsf{P}_\theta(w_{1..T}\; q_{1..T})}$, F is a function from gold and hypothesized decision sequences to parsing F-scores, and $\hat{b}$ is a global running average of F scores of all sampled trees.

After the model is trained, the parser uses beam search to find the approximate best parse. A large beam width is desirable because it provides more accurate parses and straightforward ways to calculate psycholinguistic measures like surprisal which requires marginalization.

5 Experiments

A first set of experiments compare the linguistic accuracy of the bounded neural parser to other generative incremental parsers using bracketing F1 scores on the Penn Treebank (Marcus et al., 1994). A second set of experiments then compare the psycholinguistic accuracy of the bounded neural parsing model against an equivalently bounded non-neural parsing model by regressing syntactic surprisal derived from each model to self-paced reading times from the Natural Stories Corpus (Futrell et al., 2018) and eye-tracking fixation durations of newspaper article reading from the Dundee Eye-tracking Corpus (Kennedy et al., 2003).

The neural parser used in all experiments is trained on Sections 02 - 21 of the Wall Street Journal part of the Penn Treebank. Hyper-parameters of the parser are tuned on the development set, WSJ Section 22. The cross-entropy objective is used for about 9 epochs before accuracy on the development set starts to decrease, with stochastic gradient descent (SGD) using initial learning rate = 0.1 and gradually decreasing the learning

[4] This term does not include p_t because preterminal (POS) decisions are soft, and F scores provide no supervision to POS tagging accuracy. To increase efficiency in sampling, GENERATE decisions are not sampled.

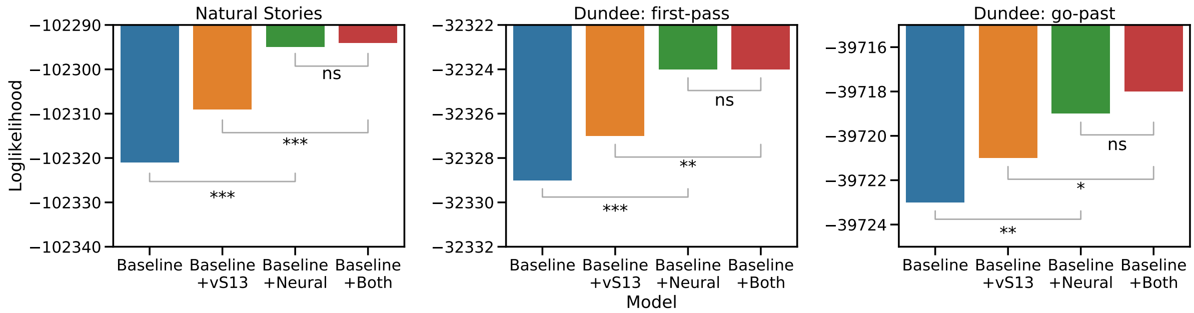


Figure 4: Goodness of fit of the regression model with different surprisal values calculated by the incremental left-corner parser in van Schijndel et al. (2013) and the incremental neural parser to human reading times and eye tracking data. Likelihood ratio tests are conducted to obtain p values. $^{***}: p < 1 \times 10^{-4}$, $^{**}: p < 1 \times 10^{-3}$, $^{*}: p < 1 \times 10^{-2}$, ns: $p > 0.05$.

rate to be 0.001, and $\lambda = 1 \times 10^{-6}$. Training then switches to the REINFORCE objective until the parser reaches the maximum F1 score (about 3 epochs), with SGD using learning rate = 5×10^{-3} and $\lambda = 1 \times 10^{-5}$. Using REINFORCE adds 0.3 F1 points on the development set. The pretrained English word embeddings are from Liu and Zhang (2017). Dropout is applied to input to all layers.

Experiments first evaluate model performance on Section 22 of WSJ as the development set and Section 23 as the test set for linguistic accuracy evaluation with a beam width of 2000. These experiments use the extended set of constraints for parsing WSJ for efficiency. This evaluation reports EVALB F scores on both datasets. Trees in the training set are binarized with left-branching constituents and the unary nodes are removed from gold trees following van Schijndel et al. (2013).

The trained model then is used to calculate surprisal for sentences in the Natural Stories Corpus and the Dundee Corpus for psycholinguistic accuracy evaluation. These experiments only use the exploratory set of both corpora. Corpus cleaning follows van Schijndel and Schuler (2013). The parser uses the basic set of constraints to parse the Natural Stories and Dundee corpora, only rejecting parses that would lead to premature termination of the parsing process while doing beam search, with width 2000.

5.1 Linguistic accuracy evaluation

A linguistic accuracy evaluation compares the performance of the bounded neural parser with the published results of generative incremental parsers that are able to calculate psycholinguistic predictors. These experiments first compare parsing scores of the current parser on the development set of WSJ with results reported in Hale et al. (2018) in Table 2. Results show that there is a 1.2 point difference between this parser and the parser used in Hale et al. (2018). This decrease may be attributable to the bounded stack losing information about past parsing decisions. Table 2 also shows labeled bracketing F1 scores of the current parser compared with other generative incremental parsers widely used for calculating surprisal predictors, especially van Schijndel et al. (2013) which is the previous state-of-the-art memory-bounded generative incremental parser, on the test set. The neural parser is more accurate than all of the published results of the memory-bounded parsers.

Model	F_1	
	dev	test
memory-bounded parsers		
Demberg et al. (2013)	-	78.7
Roark (2001)	-	85.7
van Schijndel et al. (2013)	-	87.8
this work	90.1	**89.5**
memory-unbounded parsers		
Hale et al. (2018)	**91.3**	-

Table 2: Parsing results (%) on development data, WSJ section 22 and test data, WSJ section 23 for memory-bounded and unbounded generative incremental parsers.

5.2 Psycholinguistic accuracy evaluation

The psycholinguistic accuracy of the parser is evaluated by comparing surprisal predictors calculated by the neural left-corner model against surprisal predictors from the statistical left-corner parser of van Schijndel et al. (2013), which is the memory-bounded generative incremental parser with current state-of-the-art linguistic accuracy. This evaluation uses linear mixed effects models in lme4[5] to regress to both reading time (how long a word is read) data in the Natural Stories Corpus and first-pass (how long a word is first fixated) and go-past (how long before a subsequent word is fixated) fixation durations in the Dundee Corpus, with all four combinations of the neural psycholinguistic (referred to as Neural) and van Schijndel et al. (2013) (referred to below as vS13) surprisal predictors in a diamond ANOVA.[6] All the models also have random intercepts for subject-sentence interaction and word, and random by-subject slopes for all fixed effects. Since this evaluation uses ablative testing to determine whether a fixed effect significantly improves the fit of a model compared to that model without that fixed effect, all models also include random slopes for all fixed effects, even if that particular fixed effect is not used in that model.

Psycholinguistic evaluation results are shown in Figure 4 in terms of model fit to human behavioral data. Results show that surprisal values derived from the bounded neural parser explain behavioral data better than the bounded statistical parser. First, the results show that the neural parser produces more human-like surprisal values than the vS13 parser in all three experiments. This is shown by the fact that adding vS13 to a model which already has Neural (Baseline+Both vs. Baseline+Neural) yields no significant improvement in model fit. The other comparison in which Neural surprisal values are added on top of vS13 surprisal values (Baseline+Both vs. Baseline+vS13) also shows this effect, because significant improvement in model fit is observed. Second, both surprisals derived from memory-bounded generative incremental parsers significantly increase model fit, showing that surprisal is a reliable predictor of both reading times and fixation durations, but in all three experiments, the results show that Baseline+Neural achieves much better model fit to the data than Baseline+vS13 with larger loglikelihood improvements compared to Baseline.

[5] https://cran.r-project.org/web/packages/lme4/index.html

[6] Please see the supplemental materials for detailed independent variable description and lmer formulae.

6 Conclusion

This paper proposes a new incremental left-corner transition system that can calculate surprisal and other psycholinguistic predictors, and a new neural generative incremental parser to use this transition system to do memory-bounded incremental generative parsing. Experiments described in this paper show that this generative left-corner neural network parser is competitive with incremental generative parsers that use unbounded stack memory in a parsing task, and outperforms statistical memory-bounded generative left-corner parsers both in parsing accuracy and in fitting human behavioral data on two different datasets, showing that accuracy gains of neural parsers can be reliably extended to psycholinguistic modeling without risk of distortion due to unbounded working memory.

References

Steven P Abney and Mark Johnson. 1991. Memory Requirements and Local Ambiguities of Parsing Strategies. *J. Psycholinguistic Research*, 20(3):233–250.

J D Bransford and J J Franks. 1971. The Abstraction of Linguistic Ideas. *Cognitive Psychology*, 2:331–350.

Do Kook Choe and Eugene Charniak. 2016. Parsing as Language Modeling. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Benoit Crabbé, Murielle Fabre, and Christophe Pallier. 2019. Variable beam search for generative neural parsing and its relevance for the analysis of neuro-imaging signal. In *EMNLP-IJCNLP*, pages 1150–1160. Association for Computational Linguistics.

Vera Demberg and Frank Keller. 2008. Data from eye-tracking corpora as evidence for theories of syntactic processing complexity. *Cognition*, 109(2):193–210.

Vera Demberg, Frank Keller, and Alexander Koller. 2013. Incremental, Predictive Parsing with Psycholinguistically Motivated Tree-Adjoining Grammar. *Computational Linguistics*, 39(4):1025–1066.

Vera Demberg, Asad B Sayeed, Philip J Gorinski, and Nikolaos Engonopoulos. 2012. Syntactic surprisal affects spoken word duration in conversational contexts. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 356–367.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent Neural Network Grammars. *Proceedings of the 54th Annual*

Meeting of the Association for Computational Linguistics, 3(2013).

Stefan Frank. 2009. Surprisal-based comparison between a symbolic and a connectionist model of sentence processing. *Proceedings of the 31st annual conference of the cognitive science society*, pages 1139–1144.

Richard Futrell, Edward Gibson, Harry J . Tily, Idan Blank, Anastasia Vishnevetsky, Steven Piantadosi, and Evelina Fedorenko. 2018. The Natural Stories Corpus. In *Proceedings of the 11th International Conference on Language Resources and Evaluation*, Paris, France.

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

Klaus Greff, Rupesh K Srivastava, Jan Koutnik, Bas R Steunebrink, and Jurgen Schmidhuber. 2017. LSTM: A Search Space Odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, 28(10):2222–2232.

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 on Language technologies*, pages 1–8.

John Hale. 2006. Uncertainty about the rest of the sentence. *Cognitive Science*, 30(4):643–672.

John Hale, Chris Dyer, Adhiguna Kuncoro, and Jonathan R. Brennan. 2018. Finding Syntax in Human Encephalography with Beam Search. In *Proceedings of the 56st Annual Meeting of the Association for Computational Linguistics*.

Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H Clark, and Philipp Koehn. 2013. Scalable modified Kneser-Ney language model estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 690–696, Sofia, Bulgaria.

Lifeng Jin, William Schuler, Finale Doshi-Velez, Timothy A Miller, and Lane Schwartz. 2018. Unsupervised Grammar Induction with Depth-bounded PCFG. *Transactions of the Association for Computational Linguistics (TACL)*.

Philip N Johnson-Laird. 1983. *Mental models: Towards a cognitive science of language, inference, and consciousness*. Harvard University Press, Cambridge, MA, USA.

Alan Kennedy, James Pynte, and Robin Hill. 2003. The Dundee Corpus. In *Proceedings of the 12th European conference on eye movement*.

Nikita Kitaev and Dan Klein. 2018. Constituency Parsing with a Self-Attentive Encoder. In *Proceedings of the 56st Annual Meeting of the Association for Computational Linguistics*.

Adhiguna Kuncoro, Chris Dyer, John Hale, Dani Yogatama, Stephen Clark, and Phil Blunsom. 2018. LSTMs Can Learn Syntax-Sensitive Dependencies Well, But Modeling Structure Makes Them Better. In *Proceedings of the 56st Annual Meeting of the Association for Computational Linguistics*, pages 1–11.

Minh Le and Antske Fokkens. 2017. Tackling Error Propagation through Reinforcement Learning: A Case of Greedy Dependency Parsing. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, volume 1, pages 677–687.

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.

Jiangming Liu and Yue Zhang. 2017. In-Order Transition-based Constituent Parsing. *Transactions of the Association for Computational Linguistics*.

Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyreand Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The Penn Treebank: Annotating predicate argument structure. In *Proceedings of the ARPA Human Language Technology Workshop*.

George A. Miller and Stephen Isard. 1964. Free recall of self-embedded english sentences. *Information and Control*, 7(3):292–303.

Philip Resnik. 1992. Probabilistic tree-adjoining grammar as a framework for statistical natural language processing. In *Proceedings of the Fourteenth International Conference on Computational Linguistics*, pages 418–424, Nantes, France.

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

Brian Roark, Asaf Bachrach, Carlos Cardenas, and Christophe Pallier. 2009. Deriving lexical and syntactic expectation-based measures for psycholinguistic modeling via incremental top-down parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Langauge Processing*, pages 324–333.

D J Rosenkrantz and P M Lewis. 1970. Deterministic left corner parsing. In *11th Annual Symposium on Switching and Automata Theory (swat 1970)*, pages 139–152.

Marten van Schijndel, Andy Exley, and William Schuler. 2013. A Model of Language Processing as Hierarchic Sequential Prediction. *Topics in Cognitive Science*, 5(3):522–540.

Marten van Schijndel and William Schuler. 2013. An Analysis of Memory-based Processing Costs using Incremental Deep Syntactic Dependency Parsing. In *Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics*, pages 37–46.

Marten van Schijndel and William Schuler. 2015. Hierarchic syntax improves reading time prediction. In *Proceedings of Human Language Technologies: The 2015 Annual Conference of the North American Chapter of the ACL*, pages 1597–1605. Association for Linguistics.

William Schuler, Samir AbdelRahman, Tim Miller, and Lane Schwartz. 2010. Broad-coverage parsing using human-Like memory constraints. *Computational Linguistics*, 36(1):1–30.

Edward Stabler. 1994. The finite connectivity of linguistic structure. In *Perspectives on Sentence Processing*, pages 303–336. Lawrence Erlbaum.

Mitchell Stern, Daniel Fried, and Dan Klein. 2017. Effective Inference for Generative Neural Parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Langauge Processing*.

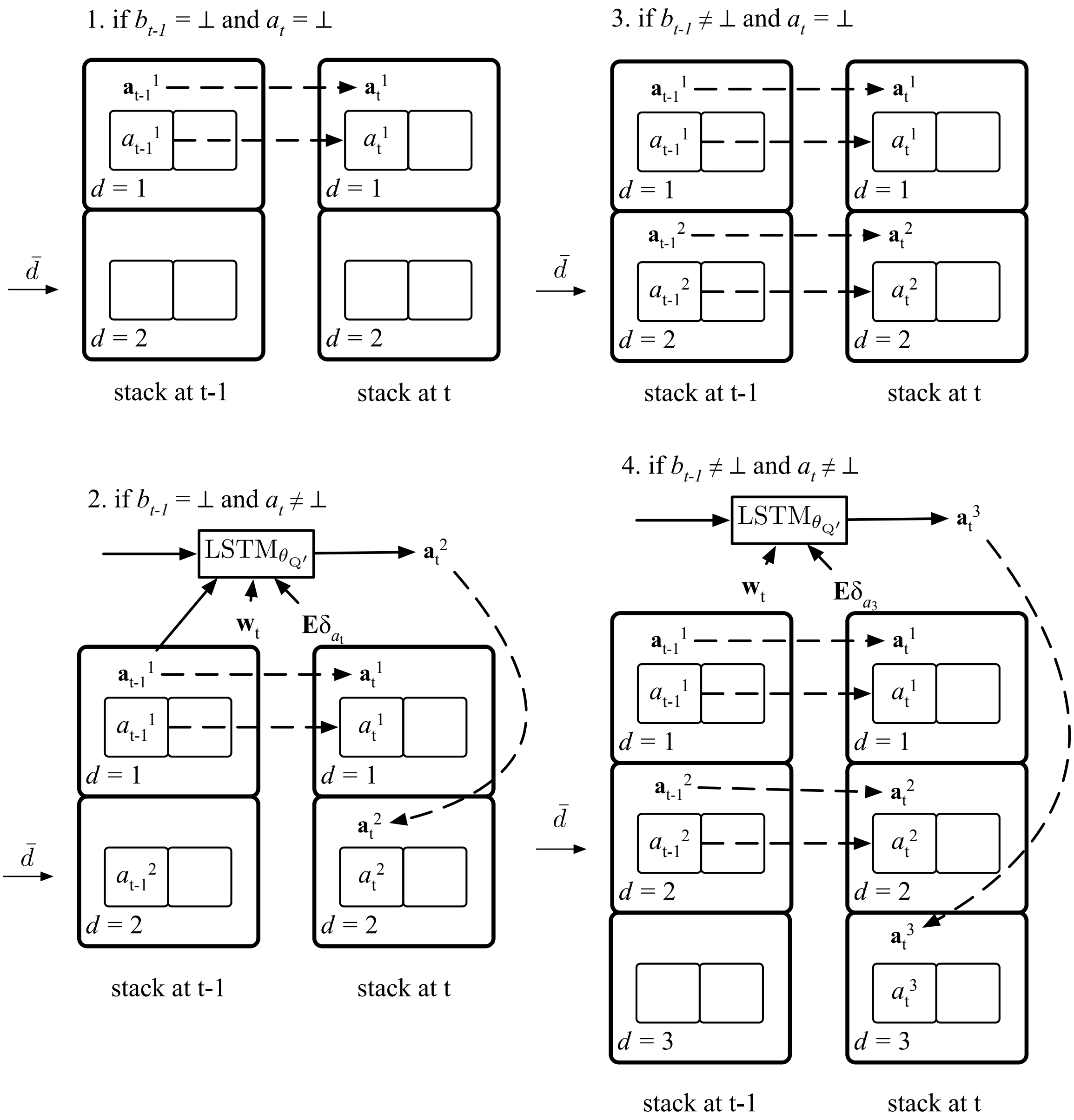
David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. 2015. Structured Training for Neural Network Transition-Based Parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 323–333.

Ronald J Williams. 1992. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Machine Learning*, 8(3-4):229–256.

Stephen Wu, Asaf Bachrach, Carlos Cardenas, and William Schuler. 2010. Complexity Metrics in an Incremental Right-corner Parser. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1189–1198.

A Illustration for the stack operations for the top category

Figure 5 shows the four kinds of stack manipulations for the top category described in Equation 12. The stack manipulations for the bottom category are similar to those illustrated here. At each depth of the stack, there are top and bottom categories, as well as vectors representing them. Figure 5 only shows stack depths with non-empty elements, as well as the stack depth the parser is currently at. Because Equation 12 only deals with the top category, only top categories are shown in the illustration. For all operations, the current working depth is 2. When both the bottom decision at the previous timestep b_{t-1} and the top decision at the current time step a_t are NULL, the parser returns to the stack depth above, copying the top category a^1_{t-1} and its vector $\mathbf{a}^1_{t-1}$ from the stack at the previous timestep to the new stack, shown in Figure 5.1. If the bottom category is NULL, but the top category at the current timestep is predicted to be a real category, the parser copies the categories from the stack depths above and generates a new vector $\mathbf{a}^2_t$ for the newly predicted top category a_t using the top category from the depth above $\mathbf{a}^2_{t-1}$, the current word $\mathbf{w}_t$ and the embedding of the predicted top category a_t as input to $\text{LSTM}_{\theta_{Q'}}$, shown in Figure 5.2. If the bottom category from the previous timestep is not NULL, but the current top category is NULL, the parser copies top categories from all depths into the new stack, shown in Figure 5.3. Finally, if the bottom category and the top category are not NULL, the parser first copies all top categories and vectors, and then generates a new embedding $\mathbf{a}^3_t$ using the current word $\mathbf{w}_t$ and the category embedding of the new top category for the top category a_t in the stack depth below the current depth, creating a new derivational fragment, shown in Figure 5.4.


1. if $b_{t-1} = \perp$ and $a_t = \perp$

2. if $b_{t-1} = \perp$ and $a_t \neq \perp$

3. if $b_{t-1} \neq \perp$ and $a_t = \perp$

4. if $b_{t-1} \neq \perp$ and $a_t \neq \perp$

Figure 5: The stack operations for the top categories described in Equation 12.

B Constraint sets for parsing

There are two sets of constraints for different use cases for the parser to prune parses on the beam.

Basic set is the set of constraints used when psycholinguistic measures are needed. It includes two constraints: the first PJA must not be NULL, and all parses with $d = 0$ are removed from the beam while parsing.

Extended set is the set of constraints used when searching for a best parse. It guarantees that all parses on the beam to be valid parses. Let n be the length of the sentence, d the depth at the current time step and o be the offset of the current word at i to the end of the sentence, we can use the following constraints to ensure well-formed parses:

1. if $d = o - 1$, then both PJA_t and PJB_t must be NULL.
2. if $d = 1$ and $o > 1$, then PJA_t and PJB_t cannot be NULL at the same time.
3. if $d = o - 2$, if constraint 2 is also true, then PJA_t and PJB_t cannot be both NULL and both X, otherwise PJA_t and PJB_t cannot be both X.

The parser with the extended set can be seen as the parser with the basic set and a wider beam if it is used for getting the best parse. If a parse is at the top of the beam with the extended set, then it will be also at the top of beam with the basic set provided that it is not lost in beam search and a better one is not found due to using a wider beam.

Hyper-parameter	Value
LSTM layer	2
Word embedding dim	80
English pretrained word embedding dim	100
POS tag embedding dim	48
Decision embedding dim	50
Stack-LSTM input dim	256
Stack-LSTM hidden dim	256
Dropout	0.3
Feed-forward layer	2

Table 3: Hyper-parameters of the model used in the evaluations.

C Hyperparameters

Table 3 shows the hyperparameters the model use for all experiments. These values are tuned on the development set.

D lmer formulae for psycholinguistic experiments

The following sections record the lmer formulae for all psycholinguistic experiments mentioned in the paper. The independent variables included in all models are: word length (wlen), unigram probability (unigram) and 5-gram forward probability of the current word given the preceding context (fwprob5surp). All independent variables are centered and scaled before being added to each model. The 5-gram probabilities are interpolated 5-grams computed over the Gigaword corpus using KenLM (Heafield et al., 2013). Regressions to eye tracking data also include word position (wdelta) as well as whether the previous word was fixated on (prevwasfix). For regressing to go-past durations, one-position spillover measures for unigram (unigramS1) and 5-gram forward probability (fwprob5surpS1) are also added.

D.1 Natural stories

The lmer formula for regression to reading times in the Natural Stories Corpus is:

log(reading times) ~ z.(wlen) + z.(unigram) + z.(fwprob5surp) + z.(neuralsurp) + z.(vssurp) + (1 + z.(wlen) + z.(unigram) + z.(fwprob5surp) + z.(neuralsurp) + z.(vssurp) | subject) + (1 | word) + (1 | sentid:subject).

The difference between four evaluated models is whether each surprisal variable is used as a fixed effect or not. This is true for all the experiments.

D.2 Dundee: first pass

The lmer formula for regression to first pass fixation durations in the Dundee Corpus is:

log(first pass fixation duration) ~ z.(sentpos) + z.(wlen) + z.(wdelta) + z.(prevwasfix) + z.(fwprob5surp) + z.(cumfwprob5surp) + z.(totsurp) + z.(cumtotsurp) + z.(totsurpNeural) + (1 + z.(sentid) + z.(sentpos) + z.(wlen) + z.(wdelta) + z.(prevwasfix) + z.(fwprob5surp) + z.(cumfwprob5surp) + z.(vssurp) + z.(cumtotsurp) + z.(neuralsurp) | subject) + (1 | word) + (1 | sentid).

D.3 Dundee: go past

The lmer formula for regression to go past fixation durations in the Dundee Corpus is:

log(go past fixation duration) ~ z.(wlen) + z.(wdelta) + z.(prevwasfix) + z.(unigram) + z.(unigramS1) + z.(cumfwprob5surp) + z.(cumfwprob5surpS1) + z.(totsurpNeural) + z.(totsurp) + (1 + z.(wlen) + z.(wdelta) + z.(prevwasfix) + z.(unigram) + z.(unigramS1) + z.(cumfwprob5surp) + z.(cumfwprob5surpS1) + z.(neuralsurp) + z.(vssurp) | subject) + (1 | word) + (1 | sentid:subject)