Incremental, Predictive Parsing with Psycholinguistically Motivated Tree-Adjoining Grammar

Vera Demberg[*]
Saarland University

Frank Keller[**]
University of Edinburgh

Alexander Koller[†]
University of Potsdam

Psycholinguistic research shows that key properties of the human sentence processor are incrementality, connectedness (partial structures contain no unattached nodes), and prediction (upcoming syntactic structure is anticipated). There is currently no broad-coverage parsing model with these properties, however. In this article, we present the first broad-coverage probabilistic parser for PLTAG, a variant of TAG that supports all three requirements. We train our parser on a TAG-transformed version of the Penn Treebank and show that it achieves performance comparable to existing TAG parsers that are incremental but not predictive. We also use our PLTAG model to predict human reading times, demonstrating a better fit on the Dundee eye-tracking corpus than a standard surprisal model.

1. Introduction

Evidence from psycholinguistic research suggests that human language comprehension is **incremental**. Comprehenders do not wait until the end of the sentence before they build a syntactic representation for the sentence; rather, they construct a sequence of partial representations for sentence prefixes. Experimental results indicate that each new word that is read or heard triggers an update of the representation constructed so far (Tanenhaus et al. 1995; Konieczny 2000).

There is also evidence for **connectedness** in human language processing (Sturt and Lombardo 2005). Connectedness means that all input words are attached to the

[*] Cluster of Excellence Multimodal Computing and Interaction (MMCI), Postfach 151150, 66041 Saarbrücken, Germany. E-mail: vera@coli.uni-saarland.de.

[**] Institute for Language, Cognition, and Computation, School of Informatics, 10 Crichton Street, Edinburgh EH8 9AB, UK. E-mail: keller@inf.ed.ac.uk.

[†] Department of Linguistics, Karl-Liebknecht-Straße 24–25, 14476 Potsdam, Germany. E-mail: koller@ling.uni-potsdam.de.

Submission received: 4 July 2011; revised submission received: 22 December 2012; accepted for publication: 22 January 2013.

doi:10.1162/COLI_a_00160

same syntactic structure (though connected structures can be constructed in parallel); comprehenders build no unconnected tree fragments, even for the sentence prefixes that arise during incremental processing.

Furthermore, a range of studies show that comprehenders make **predictions** about upcoming material on the basis of sentence prefixes. There is experimental evidence that listeners predict complements of verbs based on their selectional restrictions (Altmann and Kamide 1999); readers predict a phrase introduced by *or* on encountering the word *either* (Staub and Clifton 2006); also the subcategorization frame of a verb can be used for prediction (Arai and Keller 2013). These studies find processing facilitation if predictions can be verified successfully, compared with sentences where predictions cannot be made or turn out to be incorrect. Presumably, the human sentence processor uses prediction mechanisms to enable efficient comprehension in real time.

The three concepts of incrementality, connectedness, and prediction are fundamentally interrelated: Maintaining connected partial analyses is only nontrivial if the parsing process is incremental, and prediction means that a connected analysis is required also for words the parser has not yet seen. In this article, we exploit the interrelatedness of incrementality, connectedness, and prediction to develop a parsing model for psycholinguistically motivated TAG (PLTAG; Demberg and Keller 2008b). This formalism augments standard tree-adjoining grammar (TAG; Joshi, Levy, and Takahashi 1975) with a predictive lexicon and a verification operation for validating predicted structures. As we show in Section 2, these operations are motivated by psycholinguistic findings.

We argue that our PLTAG parser can form the basis for a new model of human sentence processing. We successfully evaluate the predictions of this model against reading time data from an eye-tracking corpus, showing that it provides a better fit with the psycholinguistic data than the standard surprisal model of human sentence processing. Crucially, this evaluation relies on the **broad-coverage** nature of our PLTAG parser, that is, the fact that it achieves high coverage and good parsing accuracy on corpus data. Only a broad-coverage parser can be used to model naturalistic data such as reading times from an eye-tracking corpus; this sets our approach apart from most other psycholinguistic models, for which only small-scale implementations for restricted data sets are available.

On the technical side, our key contribution is a novel parsing algorithm for probabilistic PLTAG. Incremental fully connected parsing is fundamentally more difficult than non-incremental parsing or parsing without connectedness: Explicit hypotheses about how the words in a sentence are connected have to be made before all of the relevant evidence has been encountered in the input. The number of connected analyses grows quickly with the length of the sentence, and this problem gets worse in the presence of predicted structure. Our parsing algorithm addresses this by grouping equivalent analyses together by only considering the **fringes** of trees, and by controlling the prediction process via supertagging. We evaluate our parser on a TAG-converted version of the Penn Treebank, achieving a coverage of 98.09% and an F-score of 79.41. These results approach the performance of previous (non-predictive) incremental TAG parsers.

We present a formalization of PLTAG in Section 3, introduce the PLTAG parsing algorithm and probability model in Section 4, show how a PLTAG lexicon can be induced from an augmented version of the Penn Treebank in Section 5, test parsing performance in Section 6, and finally provide a psycholinguistic evaluation on an eye-tracking corpus in Section 7.

2. Background and Related Work

This section situates the current work with respect to the experimental literature on human parsing, and with respect to prior work on incremental parsing.

2.1 Prediction, Incrementality, and Connectedness in Human Parsing

We start with a short review of the experimental evidence for incremental, predictive, and connected processing in human parsing. In a classic study, Altmann and Kamide (1999) showed that listeners can predict the complement of a verb based on its selectional restrictions. Participants heard sentences such as:

(1) a. The boy will *eat* the cake.
b. The boy will *move* the cake.

while viewing images that depicted sets of relevant objects—in this example, a cake, a train set, a ball, and a model car. Altmann and Kamide (1999) monitored participants' eye-movements while they heard the sentences and found an increased number of looks to the cake during the word *eat* compared with the control condition, that is, during the word *move* (only the cake is edible, but all depicted objects are movable). This indicates that selectional preference information provided by the verb is not only used as soon as it is available (i.e., incremental processing takes place), but this information also triggers the prediction of upcoming arguments of the verb. Subsequent work has generalized this effect, demonstrating that syntactic information such as case marking is also used for prediction (Kamide, Scheepers, and Altmann 2003).

More recently, Arai and Keller (2013) used the same experimental paradigm to show that verb subcategorization information is used for prediction. They compared transitive and intransitive verbs in sentences such as:

(2) a. The inmate *offended* the judge.
b. The inmate *frowned* at the judge.

Participants' eye-movements indicate which subcategorization frame they assume when they process the verb. While hearing *offended*, listeners predict upcoming patient information and look at the judge. While hearing *frowned*, no such prediction is possible, and there is no increase of looks at the judge (this increase is observable later, during *at*). This shows that the human parser uses the subcategorization frame of the verb to anticipate upcoming syntactic structure, working out whether this structure contains a noun phrase argument or not.

Selectional restrictions, case marking, and subcategorization arguably are all encoded as part of lexical items, which raises the question whether the prediction of larger structural units is also possible. This was addressed by a study of Staub and Clifton (2006), who investigated prediction in coordinate structures. They compared sentences such as:

(3) a. Peter read *either* a book or an essay in the school magazine.
b. Peter read a book or an essay in the school magazine.

By monitoring eye-movements during reading, they found that the presence of *either* leads to shorter reading times on *or* and on the noun phrase that follows it in Example (3a), compared to the control condition Example (3b). This suggests that the word *either* makes it possible to anticipate an upcoming noun phrase conjunction, ruling out verb phrase conjunction (which remains possible in Example (3b)). This result can be taken as evidence for structural prediction, that is, prediction that goes beyond the lexical information (case marking, subcategorization, etc.) encoded in the word *either*.

Let us now turn to the evidence for connectedness in human parsing. Connectedness means that the parser only generates syntactic trees that cover all of the input received so far. The claim is that comprehenders do not build unconnected tree fragments, even when the syntactic heads needed to build a connected structure are not available yet during incremental processing. Evidence for this claim comes from an experiment by Sturt and Lombardo (2005), who investigated the binding of pronouns in sentences such as:

(4) a. The pilot embarrassed *Mary* and put *herself* in an awkward situation.
b. The pilot embarrassed *Mary* and put *her* in an awkward situation.

They found increased reading times on the word *herself* in Example (4a), but not on *her* in Example (4b). They attribute this to a gender mismatch between *herself* and its antecedent *pilot*, whose stereotypical gender is masculine. No such mismatch occurs in Example (4b), as the antecedent of *her* is *Mary*.

Crucially, this gender mismatch can only be detected if the anaphor is c-commanded by its antecedent. The c-command relationship can only be established if the parser builds a fully connected structure, which includes a path from the anaphor to its antecedent. A parser that operates on unconnected sentence fragments therefore is unable to predict the contrast in Example (4); Sturt and Lombardo (2005) use this to argue for TAG as the basis for a model of human sentence processing, and against formalisms with a weaker notion of connectedness, such as Combinatory Categorial Grammar (CCG; Steedman 2000). Subsequent work has provided evidence for connectedness in a range of other phenomena, including sluicing and ellipsis (Aoshima, Yoshida, and Phillips 2009; Yoshida, Walsh-Dickey, and Sturt 2013).

2.2 Incremental Parsing Models

In the previous section, we identified incrementality, connectedness, and prediction as key desiderata for computational models of human parsing. In what follows, we will review work on parsing in computational linguistics in the light of these desiderata.

Incremental parsers for a range of grammatical formalisms have been proposed in the literature. An example is the work of Shen and Joshi (2005), who propose an efficient incremental parser for a variant of TAG, spinal TAG. This approach, however, allows multiple unconnected subtrees for a sentence prefix and uses a look-ahead of two words, that is, it does not build connected structures. An example of a TAG parser that is both incremental and builds connected structures is the work of Kato, Matsubara, and Inagaki (2004). This comes at the price of strong simplifying assumptions with respect to the TAG formalism, such as not distinguishing modifiers and arguments. (We will return to a discussion of other TAG parsers in Section 6.1.)

An example of an incremental parser based on context-free grammars is the one proposed by Roark (2001). That parser uses a top–down algorithm to build fully connected structures; it is also able to compute probabilities for sentence prefixes, which makes it attractive for psycholinguistic modeling, where prefix probabilities are often used to predict human processing difficulty (see Section 7 for details). The Roark parser has been shown to successfully model psycholinguistic data from eye-tracking corpora (Demberg and Keller 2008a; Frank 2009) and other reading time data (Roark et al. 2009). It therefore is a good candidate for a broad-coverage model of human parsing, and will serve as a standard of comparison for the model proposed in the current article in Section 7. The Roark parser has been extended with discriminative training (Collins and Roark 2004), resulting in a boost in parsing accuracy. Prefix probabilities cannot be computed straightforwardly in a discriminative framework, however, making this approach less interesting from a psycholinguistic modeling point of view.

Wu et al. (2010) propose another approach based on prefix probabilities over context-free structures. These are generated in their approach using a bottom–up parsing algorithm based on hierarchical hidden Markov models (HMMs) (Schuler et al. 2010). They show that prefix probabilities, as well as a new measure based on the embedding depth of the HMM, successfully predicts human reading time data.

In the dependency parsing literature, Nivre (2004) proposes a parser that builds dependency structures word-by-word, based on a shift-reduce algorithm. This approach is highly efficient, but has two disadvantages from a psycholinguistic point of view: Firstly, it cannot guarantee that only connected structures are built, as the stack potentially contains unconnected words (though Nivre [2004] shows that 68.9% of all parse configurations contain only connected components, rising to 87.1% if only valid dependency graphs are considered). Secondly, Nivre (2004) uses a discriminative probability model over parser actions, which means that prefix probabilities cannot be computed directly. It is, however, possible to predict reading times using probabilities over parser actions rather than prefix probabilities, as Boston et al. (2008) have shown by using the Nivre (2004) parser to model reading times in an eye-tracking corpus of German sentences.

An interesting alternative to Nivre's approach has been proposed by Beuck, Köhn, and Menzel (2011), who introduce an incremental version of Weighted Constraint Dependency Grammar (WCDG). The proposed parsing framework is able to produce structures that are both connected and predictive; this is achieved by the introduction of virtual nodes in the dependency tree, an idea akin to our use of prediction trees in TAG (detailed subsequently). WCDG parsing is non-monotonic, that is, it uses a mechanism by which the current analysis can be revised if it becomes incompatible with the input. This contrasts with the fully monotonic approach we use in the present article. In terms of evaluation, Beuck, Köhn, and Menzel (2011) present a comparison of their incremental WCDG parser with the model of Nivre (2004) for parsing German.

What is common to all of these approaches it that they lack an explicit prediction and verification mechanism (WCDG includes prediction, but not verification), which means that they cannot be used to model psycholinguistic results that involve verification cost.[1] A simple form of prediction can be achieved in a chart parser (incomplete edges in the chart can be seen as predictive), but in order to maintain psycholinguistic plausibility, an arc-eager left-corner parsing strategy needs to be used. Other parsing strategies fail to predict human processing difficulty that arises in certain cases, such as for center embedding (Thompson, Dixon, and Lamping 1991; Resnik 1992a). This is an argument against using a top–down parser such as Roark's for psycholinguistic modeling. Furthermore, it is important to emphasize that a full model of human parsing needs to not only model prediction, but also account for processing difficulty associated with the verification of predictions (we will return to this point in Section 7). None of the existing incremental parsing models includes an explicit verification component.

[1] As Demberg and Keller (2009) show, some psycholinguistic results can be accounted for by a model without verification, such as the *either . . . or* finding, whereas other results, such as the relative clause asymmetry, require a verification component; see Section 7.4 for more discussion.

In this article, we propose the first parser that instantiates the properties of incrementality, connectedness, and prediction in a psycholinguistically motivated way. We achieve this by exploiting the fact that these three concepts are closely related: In order to guarantee that the syntactic structure of a sentence prefix is fully connected, it may be necessary to build phrases whose lexical anchors (the words that they relate to) have not been encountered yet. In other words, the parser needs to predict upcoming syntactic structure in order to ensure connectedness. This prediction scheme is complemented by an explicit verification mechanism in our approach. Furthermore, unlike most existing psycholinguistic models (see Keller 2010 for an overview), our model achieves broad coverage and acceptable parsing performance on a standard test corpus. This property is essential for testing psycholinguistic models on realistic data, including eye-tracking corpora.

The PLTAG formalism was first proposed by Demberg-Winterfors (2010), who also presents an earlier version of the parsing algorithm, probability model, implementation, and evaluation described in the current article.

3. The PLTAG Formalism

We start by introducing the PLTAG formalism, which we will use throughout the article.

3.1 Incremental TAG Parsing

Tree Adjoining Grammar (TAG) is a grammar formalism based on combining trees. In what follows we will focus on lexicalized TAG (TAG; Joshi and Schabes 1992), which is the most widely used version of TAG. In this formalism, a TAG lexicon consists of a finite set of **elementary trees** whose nodes are labeled with nonterminal or terminal symbols. Each elementary tree contains an **anchor**, a leaf node labeled with a terminal symbol. At most one other leaf—the **foot node**—may carry a label of the form $A*$, where A is a nonterminal symbol. All other leaves are **substitution nodes** and labeled with symbols of the form $A\downarrow$. Elementary trees that contain a foot node are called **auxiliary trees**; those that contain no foot nodes are **initial trees**. We will generally call leaves of trees that are labeled with words **lexical** leaves. An example TAG lexicon is shown in Figure 1a–c.

TAG builds grammatical derivations out of these elementary trees using two tree-combining operations, **substitution** and **adjunction**. Figure 2 shows a derivation of the

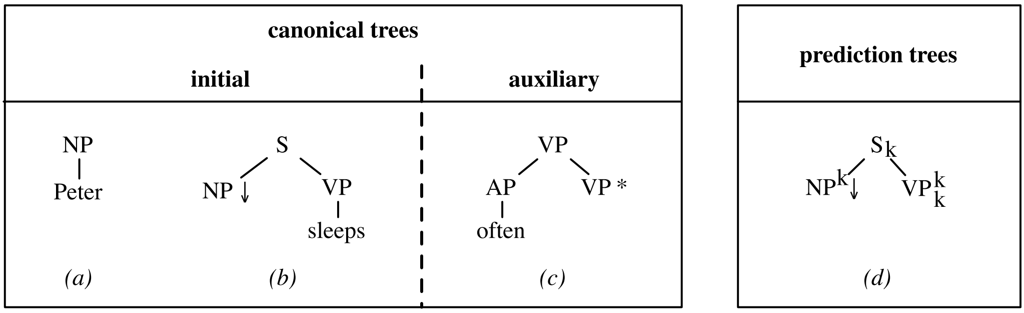


Figure 1
Elementary trees in an example PLTAG lexicon. The predictive status of nodes in a prediction tree (see Section 3.2) is marked with the markers ^{k} and $_k$.

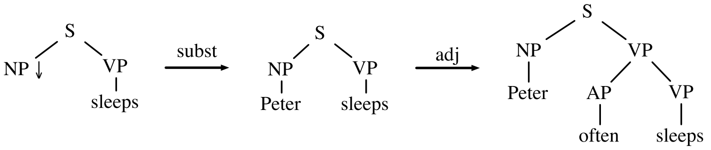


Figure 2
Example of a (non-incremental) TAG derivation of *Peter often sleeps* using the elementary trees from Figure 1.

string *Peter often sleeps*. In the first step, this derivation substitutes the elementary tree (a) from Figure 1 into the substitution node of (b). This is allowed because (a) is an initial tree, the nonterminal at the root of (a) and the nonterminal on the substitution node are the same; the substitution operation then replaces this substitution node by (a). Second, the derivation adjoins the auxiliary tree (c) into the VP node of the tree for *Peter sleeps*; we call the VP node the **adjunction site**. The resulting tree is like the original from the root down to the VP node; then it continues with (c), with the foot node replaced by what was below the VP node before. The result is a tree whose leaves are all labeled with terminal symbols, which read *Peter often sleeps* from left to right. We call such a tree a **derived tree** for this string. We generically call a substitution or adjunction operation an **integration**, and the (inner or substitution) node at which it is applied the **integration site**.

Notice that one can think of the adjunction operation as cutting the adjunction site in two halves. The upper half is identified with the root of the auxiliary tree; if one thinks of root nodes as being only the lower half of a node, these two halves recombine into a complete node. The lower half of the adjunction site is identified with the foot node; we think of the foot node as only having an upper half, which again makes a whole node. We assume that lexical leaves only have an upper half, too; this makes no difference, as no substitution or adjunction can be performed on those nodes anyway. The process is illustrated in Figure 3, which shows the recombination of node halves from different elementary trees in the adjunction step of Figure 2: Black node halves come from the elementary tree for *sleeps*, gray node halves from *Peter*, and white ones from *often*. The idea of distinguishing upper and lower node halves that are pushed apart by adjunction comes from FTAG (Vijay-Shanker and Joshi 1988), which equips each node half with a separate feature structure; at the end of the derivation process, the upper and lower feature structures of each node are unified with each other. Node halves will also play a crucial role in PLTAG.

3.2 Prediction Trees

We have argued earlier that a psycholinguistic model of sentence processing should be incremental. In the context of TAG and related formalisms, this means that a derivation

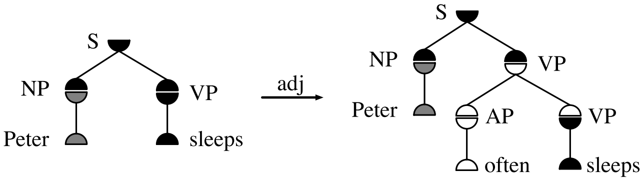


Figure 3
Fine structure of adjunction. The semicircles represent node halves; all node halves from the same elementary tree are drawn in the same color.

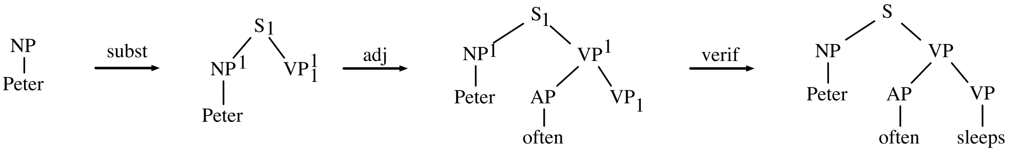


Figure 4
Example of an (incremental) PLTAG derivation of *Peter often sleeps* using the elementary trees from Figure 1.

starts with a lexicalized elementary tree for the first word of the sentence, then combines it with an elementary tree for the second word, and so on. The derivation in Figure 2 is not incremental in this sense, because it combines a tree for the first word directly with a tree for the third word in the sentence, and only then adjoins a tree for the second word. In fact, it is not possible to produce an incremental derivation of *Peter often sleeps* with the TAG grammar in Figure 1a–c (or any other linguistically motivated TAG grammar for English), because the tree for *often* must adjoin into a VP node, which is not yet available after processing *Peter*.

The PLTAG formalism (for psycholinguistically motivated TAG) solves this problem by introducing **prediction trees** in addition to the usual initial and auxiliary trees of TAG (which we will call **canonical trees**). Prediction trees are elementary trees that may or may not contain a lexical anchor (see the right part of Figure 1 for an example). PLTAG derivations can use them to predict syntactic structure that will be introduced by later words in the incremental derivation, in order to keep the syntactic structure connected and provide adjunction sites. Each node in a prediction tree carries one or two **markers** to indicate that this node has only been predicted, and is not actually part of the syntactic structure introduced by any word that has been processed so far. These markers can be thought of as decorating the upper and lower half of the node, as described earlier; therefore internal nodes always have an upper and lower marker, root nodes only have lower markers, and substitution and foot nodes only have upper markers. Note that unlike canonical trees, prediction trees can have leaves that are not substitution nodes, foot nodes, or lexical leaves, and that therefore have two halves (and, therefore, two markers); the VP leaf of the tree in Figure 1d is an example. A discussion of how the prediction tree lexicon is acquired, and how the configurations of prediction trees relate to the canonical elementary trees, is provided in Section 5.1.

When a prediction tree is used in a PLTAG derivation, all of its markers are first instantiated with fresh symbols, so we can always tell apart the markers introduced by different prediction trees. In the example PLTAG derivation of Figure 4, all the k-markers from the lexicon entry are instantiated with 1-markers. The operations of substitution and adjunction are then applied to prediction tree instances in exactly the same way as to canonical trees. In particular, adjoining into a node that carries markers pushes the two markers apart. The upper marker becomes the upper marker of the root of the auxiliary tree, whereas the lower marker becomes the lower marker of the foot node (see the second step of Figure 4). Note that if a prediction tree is adjoined into a node that already carries markers, this may create nodes that have an upper and lower marker with different values.

The use of prediction trees in PLTAG is conceptually similar to the use of type raising in incremental derivations in CCG (Steedman 2000). For example, the prediction tree in Figure 1d effectively raises the NP in Figure 1a to type (S/(S\NP)) so that it can compose with the adverb in Figure 1c. Prediction trees, however, are more powerful in terms of the incremental derivations they support: Some psycholinguistically crucial constructions (such as object relative clauses) are handled easily by PLTAG, but are not incrementally derivable in standard CCG (Demberg 2012). According to Demberg, this problem can be overcome by generalizing the CCG categories involved (in the case of object relative clauses, the category of the relative pronoun needs to be changed).

3.3 Verification

Markers are eliminated from a partial derived tree through a new operation called **verification**. Recall that markers indicate nodes that were predicted during the derivation, without having been introduced by a word that was actually observed so far. The verification operation removes these markers by matching them with the nodes of the canonical elementary tree for a word in the sentence. An example is shown in the last step of Figure 4. This is a verification step for the marker 1, using the canonical tree for *sleeps* as the **verification tree** τ_v.

Informally speaking, the job of the verification operation for the marker *i* is to (a) check that the *i*-marked node halves in the prefix tree correspond to nodes in the verification tree with respect to their labels and their tree structure; (b) remove the markers from these node halves in the prefix tree; and (c) add nodes to the prefix tree that are only present in the verification tree, but not the prediction tree. The formal definition of the verification operation is complicated by the fact that the nodes that were introduced by one prediction tree may have been pushed apart through subsequent adjunctions. This means that the *i*-marked node halves whose labels and relative positions have to be matched to the verification tree in (a) may be scattered throughout the prefix tree. Consider the example in Figure 5. In the middle prefix tree (for *Peter often*), the black node halves were contributed by the same prediction tree (with marker 1), but they were pushed apart by the adjunction of a canonical tree for *often*. These node halves are still in their original configuration in the verification tree in the box (for *sleeps*); furthermore, the canonical tree contains additional nodes that were not present in the prediction tree.

The verification operation solves this problem by establishing a **correspondence** between node halves in the prefix tree τ and the verification tree τ_v. Intuitively, a mapping between nodes can only be called a correspondence if the marked node halves in τ are arranged in the same configuration as the corresponding node halves in τ_v.

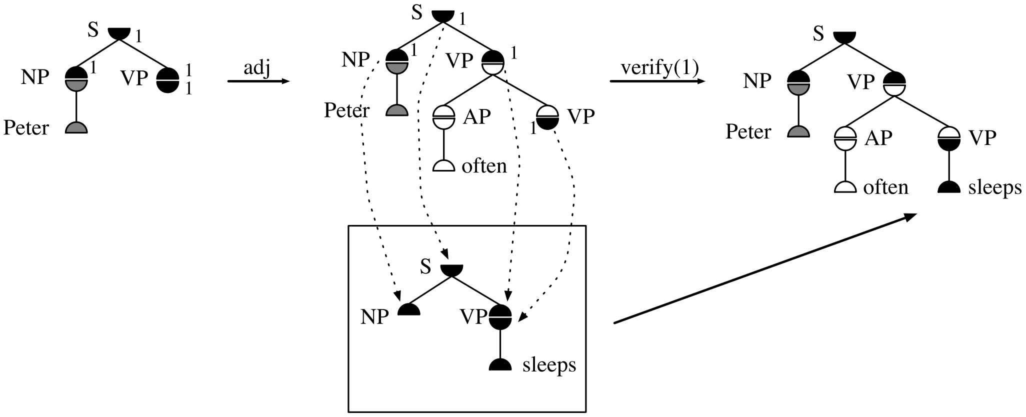


Figure 5
Fine structure of the verification operation. The dotted arrows indicate the correspondence *f* between the 1-marked node halves and the node halves of the *sleeps* tree from Figure 1.

All node halves with a given marker in the prefix tree must have corresponding node halves in τ_v. Conversely, τ_v may contain node halves without correspondents in τ. These unmatched nodes are added to τ.

Technically, a mapping f of node halves in the tree τ to node halves in the verification tree τ_v is called a **correspondence** if it has the following properties:

- for each node half h, the node labels of h and $f(h)$ are the same;
- for each lower (upper) node half $h, f(h)$ is also a lower (upper) node half;
- the mapping is injective, that is, $h_1 \neq h_2$ entails $f(h_1) \neq f(h_2)$ for all node halves h_1 and h_2;
- for any two node halves h_1 and h_2, if h_1 is above h_2 in τ, then the node half $f(h_1)$ is above the node half $f(h_2)$ in τ_v;
- for any two node halves h_1 and h_2, if h_1 precedes h_2, then $f(h_1)$ precedes $f(h_2)$.

A node half h' of τ_v **matches** the node half h in τ if $f(h) = h'$. Because f need not be subjective, not every node half in τ_v necessarily matches something. We will say for short that an entire node u in τ_v matches a node half h if it is clear from the context which half of u matches h.

Not all correspondences are useful when performing a verification. In particular, we must require that the region of τ_v that matches node halves in τ is contiguous and starts at the root of τ_v; and if some node in τ_v has unmatched children, then these must be the rightmost children of their parent. A correspondence is called **admissible** if the following holds:

- if a node in τ_v has two halves and one half matches something, then the other half also matches something;
- if the upper half of the $i + 1$-st child of some node u of τ_v matches anything, then the upper half of the i-th child of u must also match something (for any $i \geq 1$);
- if the upper half of any child of some node u of τ_v matches something, then the lower half of u must match something as well.

The verification operation for the marker i can now be performed on the prefix tree τ with a verification tree τ_v if there is an admissible correspondence f that maps all i-marked node halves of τ to node halves of τ_v. The effect of the operation is to add subtrees to certain nodes of τ: If some node u in τ_v with children $u_1, \ldots, u_n$ (from left to right) matches a (lower) node half h, and $u_1, \ldots, u_k$ but not $u_{k+1}, \ldots, u_n$ match (upper) node halves in τ (i.e., upper node halves of children of h), then the subtrees of τ_v below $u_{k+1}, \ldots, u_n$ are added to τ as the $k + 1$-st to n-th child of h. Furthermore, the marker i is removed from all node halves in τ; these predicted nodes have now been verified.

We illustrate the definition with the example in Figure 5. The verification rule first establishes a correspondence between the four 1-marked node halves of the tree for *Peter often* and the node halves of the canonical tree for *sleeps* below, drawn as dotted arrows in the figure. This correspondence maps one lower half to the root, one upper half to the substitution node, and one upper and one lower half to the VP node, and the lexical anchor node *sleeps* does not match any node half. This correspondence is admissible.

We can therefore apply the verification rule. It removes all 1-markers from the tree. Furthermore, because the node corresponding to the lower black VP node half has an unmatched child (the *sleeps* node), this node is added to the tree by the verification operation. In addition to the new lexical anchor, the verification tree could also contain further nodes below and to the right of predicted nodes; all of these would then be added.

3.4 Derivations in PLTAG

A **PLTAG derivation** starts with an elementary tree for the first input word, and then applies substitution, adjunction, and verification operations. We require a PLTAG derivation to be **incremental**, in the sense that after each derivation step, the first i leaves of the partial derived tree are unmarked and labeled with the words $w_1 \dots w_i$, for some i, and there are no other unmarked lexical leaves. Because of this property, the partial derived trees are **prefix trees**. In the case of substitution, adjunction, or verification steps with canonical trees, we need to always add an elementary tree with anchor w_{i+1} to a prefix tree for the first i words; this constraint does not apply to unlexicalized prediction trees. We call a derivation of a sentence $w_1 \dots w_n$ **complete** if $i = n$, the prefix tree contains no more substitution nodes, foot nodes, or prediction markers, and the root symbol of the prefix tree is S. The **string language** of a PLTAG grammar is the set of string yields of its complete prefix trees.

As explained in Section 2, incrementality and connectedness are key desiderata for a psycholinguistically motivated parsing model. In order to implement these properties in PLTAG we need to ensure that the parser always constructs well-formed prefix trees. This entails that PLTAG parsing differs in crucial ways from standard non-incremental TAG parsing. The reason for this is that not all operations that could be performed in a non-incremental parsing regime lead to a well-formed prefix tree. The simplest example involves a prefix tree that contains two substitution nodes: A derivation which substitutes a canonical tree into the rightmost substitution node will produce a derived tree in which a substitution node comes before a lexical leaf, and which therefore violates the incrementality requirement. Similarly, if a verification tree contains any unmatched nodes to the left of its spine (the **spine** is the path from the root to the lexical anchor), the result of the verification operation will also contain leaves that are to the left of the rightmost leaf that is labeled with a word. Such derivation steps can never be used in an incremental PLTAG derivation.

To conclude the exposition of the definition of PLTAG, let us compare the expressive capacities of PLTAG and TAG. First, PLTAG cannot be more expressive than TAG. In any derivation of some PLTAG grammar G, the nodes that each prediction tree contributes to the derived tree are replaced by the nodes of some (canonical) verification tree. Therefore the resulting derived tree can be built (non-incrementally) from only canonical trees, using substitution and adjunction. Thus if we build a TAG grammar G' from G by removing all prediction trees, G and G' describe the same set of derived trees.

Conversely, for any TAG grammar G we can also build a PLTAG grammar G' that describes the same derived trees. The basic idea is that G' contains all elementary trees of G; for each elementary tree α of G, G' furthermore contains an unlexicalized version of α as a prediction tree. Given some derived tree of G, we can then build an incremental derivation of G' which first builds an unlexicalized version of the derived tree using substitution and adjunction of prediction trees, and then verifies this derived tree from left to right. In other words, anything that TAG can derive, PLTAG can derive incrementally.

3.5 Heads

We will now describe a probability model for PLTAG derivations. Having such a model is crucial for two reasons. First, our psycholinguistic model depends on prefix probabilities to predict surprisal and reading times (Section 7). Second, the probability model will allow our parser to perform a more focused search for an incremental derivation (Section 4).

The PLTAG probability model is a bilexical probability model; that is, the probability for an operation combining elementary and prefix trees depends both on the (lexicalized) elementary tree and on the lexical head of the integration site. We use two different perspectives on lexical heads. First, we keep track of what (possibly lexicalized) elementary tree τ contributed each node in the prefix tree. When we perform a substitution or adjunction, we can condition the probability of the operation on the identity of τ.

Second, we consider a notion of heads based on the phrase structure of the (prefix or elementary) tree. We assume that this phrase structure is linguistically valid, in that the syntactic categories represent various types of phrases, and we can recognize from the node labels of a parent and its children which of the children is the head child. For instance, in the final derived tree in Figure 4, the head daughter of the S node is the VP, whereas the head daughter of the upper VP is the lower VP. If u is any node in the tree, we can follow the head daughters down to a leaf; we call this leaf the **head** of u. The head is often a lexical leaf (in the final derived tree in Figure 4, the head of the S node is *sleeps* and the head of the NP node is *Peter*), but could also be the non-lexical leaf of a prediction tree (the head of the upper VP node in the third prefix tree is the lower VP node). The head of any node on the spine of a canonical elementary tree is always the lexical anchor.

3.6 Probability Model

We are now ready to define the probability model for PLTAG. This model allows us to define a probability distribution over the derivations of any given PLTAG grammar. It makes the same independence assumptions as standard models for probabilistic TAG (Resnik 1992b; Chiang 2000): Any two applications of derivation rules are statistically independent events. We deviate from these models, however, with regard to what these events are. Earlier approaches always modeled the probability of substituting or adjoining the **lower** elementary tree, given the **upper** elementary tree and the integration site. This is inconsistent with the incremental perspective we take here, which assumes that the prefix tree is given, and we must decide how to integrate an elementary tree for the next word with it. We therefore model the probability of substituting, adjoining, or verifying the **elementary** tree, given the **prefix** tree.

Because a substitution or adjunction step may either integrate the elementary tree into the prefix tree or vice versa, we must distinguish the direction in which the operation is applied. We do this by conditioning the probability of integrating the elementary tree on the prefix tree, as well as the lower tree's root node and the upper tree's integration site. We refer to these nodes as u_e for the root or integration site node in the elementary tree, and u_p for the prefix tree root or integration site node, depending on which out of the prefix and elementary tree is the upper tree in the operation. We obtain probability distributions P_S and P_A for substitution and adjunction, as well as P_I for the initial operation that introduces the first tree, as follows. τ_e stands for the (possibly lexicalized) elementary tree and τ_p for the prefix tree.

$$\textbf{Initial:} \quad \sum_{\tau_e} P_I(\tau_e) = 1 \tag{5}$$

$$\textbf{Substitution:} \quad \sum_{\tau_e} P_S(\tau_e|\tau_p, u_e, u_p) = 1 \tag{6}$$

$$\textbf{Adjunction:} \quad \sum_{\tau_e} P_A(\tau_e|\tau_p, u_e, u_p) + P_A(NONE|\tau_p, u_p) = 1 \tag{7}$$

Each sum is over all elementary trees τ_e in the lexicon, that is, all canonical and prediction trees. For the adjunction probabilities P_A, we must also consider the possibility that the derivation process chooses not to adjoin anything at a given node u_p of the prefix tree; this is modeled as the probability of *NONE*.

For a verification step, we condition on the original integration site of the prediction tree $u_{predict}$ and the prediction tree $\tau_{predict}$ that is matched by the verification tree; areas of the prefix tree τ_p outside of $\tau_{predict}$ are ignored. The probability distribution P_V for verification operations is defined as follows:

$$\textbf{Verification:} \quad \sum_{\tau_v} P_V(\tau_v|\tau_{predict}, u_{predict}) = 1 \tag{8}$$

Here the sum is over all canonical trees τ_v in the lexicon. The probability of an entire derivation is then the product of the probabilities of the individual operations in the derivation.

Because we are unlikely to ever have seen the exact prefix tree before, we will approximate its probability using only certain **features** of the trees and nodes involved. We will discuss the features we use in Section 5.3. Furthermore, we circumvent the usual sparse data problem involved in probability models of lexicalized TAG by assuming that the choice of the unlexicalized elementary tree and the lexical anchor are independent. For substitution and adjunction probabilities, we assume that:

$$P(\tau_e|\tau_p, u_e, u_p) = P(\mathsf{unlex}(\tau_e)|\tau_p, u_e, u_p) \cdot P(\mathsf{anchor}(\tau_e)|\mathsf{unlex}(\tau_e), \mathsf{head}(u_r)) \tag{9}$$

where u_r is the root of the elementary tree that introduced u_p in the prefix tree, $\mathsf{unlex}(\tau_e)$ is the elementary tree τ_e without its lexical anchor (i.e., the tree template), $\mathsf{anchor}(\tau_e)$ is the lexical anchor of the elementary tree τ_e, and $\mathsf{head}(u)$ is the head of the node u.

If the elementary tree τ_e is substituted or adjoined into the prefix tree, then u_p is the integration site at which τ_e is added to the prefix tree τ_p, and thus $\mathsf{head}(u_r)$ is the anchor of the elementary tree that introduced the integration site. On the other hand, if the prefix tree is substituted or adjoined into the elementary tree, then u_p is the root of the prefix tree τ_p, and $\mathsf{head}(u_r)$ amounts to the head of the entire prefix tree that we have seen so far; this may be a word (*Peter* in the first tree of Figure 4), or a syntactic category (e.g., the lower VP node in the second tree of Figure 4). Note that if τ_e is an unlexicalized prediction tree, $\mathsf{anchor}(\tau_e)$ is undefined; in these cases, $\tau_e = \mathsf{unlex}(\tau_e)$.

For verification, we condition the choice of the lexical anchor of the verification tree on the head of the node $u_{predict}$, that is, the root or integration site of the tree that was combined with $\tau_{predict}$ when $\tau_{predict}$ was added in the derivation. For instance, in the last step of Figure 4, $\mathsf{head}(u_{predict})$ is *Peter*, because the prediction tree was added to the derivation by substituting the elementary tree for *Peter* into it. This replicates the situation in a non-incremental, top–down TAG derivation, in which the verification tree would have been directly integrated with $u_{predict}$ without the intermediate prediction step. Thus we factorize the bilexical probability as follows:

$$P(\tau_v|\tau_{predict}, u_{predict}) = P(\mathsf{unlex}(\tau_v)|\tau_{predict}) \cdot P(\mathsf{anchor}(\tau_v)|\mathsf{unlex}(\tau_v), \mathsf{head}(u_{predict})) \quad (10)$$

In order to calculate the probability of a derived tree using this probability model, each node is visited twice during the derivation, once from the left side and once from the right side. At each step, a substitution,[2] an adjunction, or no adjunction can happen, and the probability of the event is factored into the derivation probability.

4. The Parsing Algorithm

Now that we have defined PLTAG, we can address the question of how PLTAG can be **parsed** efficiently. Parsing, in the context of PLTAG, means finding the most probable incremental derivation whose yield is the given input string.

A naive parser might proceed as follows. It starts with a canonical tree for the first word as the initial prefix tree. Then it nondeterministically adds an arbitrary number of prediction trees (or possibly none at all) using substitution and adjunction. Third, it nondeterministically adds a canonical tree for the second word using substitution, adjunction, or verification. It repeats the second and third step until it either gets stuck (no suitable canonical tree for the next word is available) or it has found a complete derivation for the input sentence.

An actual implementation of this algorithm must perform a search that spells out all the nondeterministic choices that the algorithm makes. An unconstrained implementation is therefore prohibitively slow. The naive algorithm shares this problem with all other incremental parsing algorithms that maintain connected partial structures. In particular, parsing algorithms based on bottom–up dynamic programming are not applicable because they necessarily compute constituents for substrings that are not prefixes. A parser for PLTAG faces the additional challenge that the use of predictive trees adds a large amount of non-determinism to the parsing task, as most predictive trees are not lexically anchored in the current input, and their use is therefore less constrained.

We will now describe how the naive parser can be refined into an incremental parser for PLTAG that is efficient enough for practical use. We first introduce the concept of **fringes** of trees in Section 4.1, and describe their use in an efficient schema for tabulating parse items (Section 4.2). Then we will show how to use **supertagging** to control the use of prediction trees (Section 4.3). We will focus on the parsing algorithm from an implementation perspective here. The rule schema that our parser implements is formalized in Appendix A.

[2] Substitution can only happen if the node being considered is a substitution node.

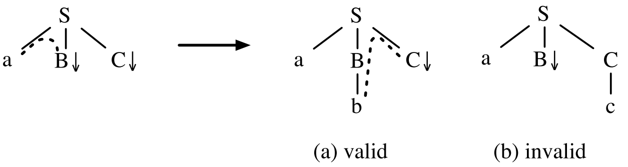


Figure 6
Substitution in an incremental derivation. Substitution into the second substitution node (b) leads to an invalid prefix tree. The dashed line indicates the current fringe.

4.1 Fringes

The crucial insight for efficient PLTAG parsing is that an incremental derivation is highly constrained in the nodes at which it may integrate a canonical tree with a prefix tree. Consider, for instance, the situation of Figure 6. The prefix tree has two substitution nodes, for *B* and *C*. Only substitution into *B*, however, leads to a valid prefix tree; if we substitute into *C*, we obtain the tree in Figure 6(b), which does not represent a prefix of the input string. A similar situation occurs for adjunction, as in Figure 7. We may adjoin an auxiliary tree β_1 whose foot node is the leftmost leaf into the left *A* node; the result is shown in (a). Alternatively, we may adjoin an auxiliary tree β_2 whose foot node is the rightmost leaf into the right *A* node, to obtain (b). If we try to adjoin β_1 into the right *A* node, we get (c), which is not a prefix tree. If we try to adjoin β_2 into the left *A* node, the result is (d); this is a tree whose first two leaves are lexical, but we have processed the second word before the first in the derivation. Thus, only the steps (a) and (b) are valid.

We can generalize this observation using the concept of **fringes**. Imagine a depth-first, left-to-right traversal of an (elementary or prefix) tree; this produces a sequence σ of nodes of the tree in which each node occurs exactly twice, once when it is first visited and once when it is left after its entire subtree has been processed. If the node is a leaf, its "downward" and "upward" occurrences are adjacent in σ. We can cut σ into components at these points. A fringe is a maximal subsequence of σ that starts at the downward visit of the root or the upward visit of a leaf, and ends at the upward visit of the root or the downward visit of a leaf. Consider, for example, the second prefix tree in Figure 4. Abusing node labels to indicate nodes to simplify the presentation, the traversal sequence σ is $\mathsf{S}^+\mathsf{NP}^+\mathsf{Peter}^+\mathsf{Peter}^-\mathsf{NP}^-\mathsf{VP}^+\mathsf{VP}^-\mathsf{S}^-$, where $+$ indicates downward visits and $-$ upward visits (see Figure 8). This sequence consists of three fringes, namely, $\mathsf{S}^+\mathsf{NP}^+\mathsf{Peter}^+$ (= root to first leaf), $\mathsf{Peter}^-\mathsf{NP}^-\mathsf{VP}^+$ (= first leaf to second leaf), and $\mathsf{VP}^-\mathsf{S}^-$ (= second leaf to root).

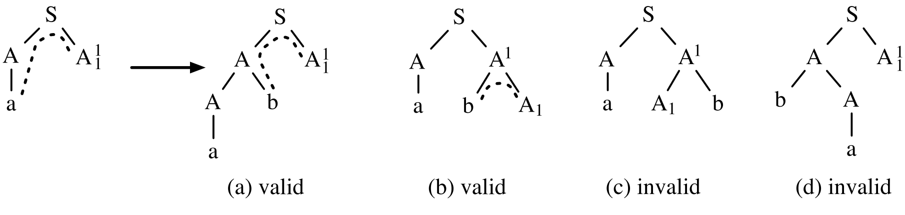


Figure 7
Adjunction in an incremental derivation. Only adjunction of auxiliary trees with leftmost foot nodes into upward visits of the current fringe (a) or with rightmost foot nodes into downward visits (b) result in valid prefix trees; the other two combinations are invalid (c, d).

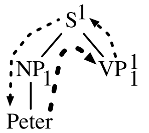


Figure 8
A depth-first, left-to-right traversal of a tree consisting of three fringes. The second fringe is current.

All substitution and adjunction operations that are applied to a (prefix or elementary) tree τ in a successful incremental derivation must use a node on the **current** fringe of τ. This is the fringe that starts with the upward visit of the rightmost lexical leaf that has no markers. Consider the examples in Figure 6; the current fringe of the original prefix tree is indicated by a dashed line. Here the first substitution node is at the end of the current fringe, and is therefore available for substitution. The second substitution node is not on the current fringe, and can therefore not be used at this point. For the adjunctions in Figure 7, we must distinguish further between the "upward" and the "downward" part of the current fringe. This distinction is important because the current fringe contains both an upward visit of the left A, which we may right-adjoin into (see Figure 7(a)), and a downward visit of the right A, at which we may left-adjoin (see Figure 7(b)). Right-adjoining into a downward visit, as in Figure 7(c), or left-adjoining into an upward visit, as in Figure 7(d), is not allowed, and the downward visit to the left A and the upward visit to the right A are not on the current fringe.

Substitution and adjunction of **prediction** trees is not constrained in the same way. A derivation step that substitutes an unlexicalized prediction tree into the C substitution node of Figure 6 would produce a prefix tree, because the derivation step does not contain lexical leaves that the original prefix tree did not. The prediction tree might even contain a lexical leaf for some word w_k: Because all nodes in a prediction tree carry markers, the resulting tree would be a valid prefix tree, and eventually the marker on the w_k leaf would be removed by a verification step at the right time. Nevertheless, we may still assume that prediction trees are only integrated with the prefix tree at a time when the integration site is on the current fringe, because an integration to the right of the current fringe does not affect the operations that can be performed to the current fringe. For the purposes of the following parsing algorithm, we therefore assume that prediction trees are only integrated at the current fringe as well.

Finally, **verification** is a more global operation whose applicability cannot be determined just from the current fringe, because the marked nodes of the matched prediction tree may be dispersed throughout the prefix tree. A verification step for the marker i will only be successful, however, if the lower half of the last node on the current fringe is marked with i. Thus the current fringe can at least provide a necessary condition for filtering out obviously useless verification operations.

4.2 Chart Parsing for PLTAG

Our parser exploits fringes to tabulate intermediate results. It manipulates a chart-like data structure with two dimensions: the index of the word up to which we have parsed the sentence, and the current fringe. The cell for (i,f) contains all the prefix trees whose first i leaves are the first i words (without markers) and whose current fringe is f.

The parser successively computes all chart entries for i from 1 to the sentence length. It starts by initializing the chart column for 1 with all canonical trees for the first word.

The current fringe of each of these trees is the sequence from the first to the second leaf, so each of these trees is stored under that fringe.

To extend the prefix trees for i to prefix trees for $i+1$, the parser then retrieves all current fringes f such that the chart has entries in the cells (i,f). For each such fringe, it determines the canonical trees that can be combined with f using substitution or adjunction, based on the heuristics outlined earlier. The prefix tree is integrated into the canonical tree if f ends in the root; otherwise the canonical tree is integrated into the prefix tree. In this way, the test whether a substitution or adjunction is possible can be done efficiently, for all prefix trees in (i,f) at once, simply by comparing f with the current fringes of the canonical trees in the lexicon. For those canonical trees that can be combined with f, the parser computes the resulting prefix trees, determines their probabilities, and enters each resulting prefix tree with current fringe f' into the cell $(i+1,f')$.

Determining and performing the applicable ways of combining the entries of (i,f) with **prediction** trees proceeds in exactly the same way, except that the parser is restricted to never combine the prefix tree with two prediction trees in a row. Strictly speaking, this makes it incomplete for the problem of computing PLTAG derivations: There can be PLTAG derivations that can only be obtained by combining a prefix tree with two or more prediction trees in a row. This restriction keeps the parser from guessing prediction trees in a completely uncontrolled fashion; unlike canonical trees, prediction trees are not necessarily licensed by a word in the input string and could therefore be added arbitrarily. In practice, the restriction to single prediction steps is counterbalanced by our extraction of "pre-combined" prediction trees from the treebank (see Section 5.1), which can contain nodes from multiple canonical trees at once.

Finally, the parser also attempts a **verification** step for all prefix trees in (i,f) for which the last node on the current fringe carries some marker k. It looks up the prediction tree that introduced the marked node and determines all verification trees that match this prediction tree and whose lexical anchor is the $i+1$-st word of the sentence. These verifications are then applied to the prefix tree, and the results are added to the appropriate cells as before.

In order to increase parsing speed, the parser uses beam search, which only retains the most likely prefix trees (see Section 6) by pruning both unlikely chart entries and unlikely analyses within a chart entry.

4.3 Supertagging for Prediction Trees

A major remaining source of complexity is the fact that (unlexicalized) prediction trees can be combined freely with prefix trees, which creates a great number of new prefix trees: At each prediction step, thousands of prediction trees can potentially be combined with all prefix trees; this is computationally not feasible. Non-incremental parsers, which do not use the unlexicalized prediction trees, have to deal with the much lower level of ambiguity among canonical trees (about 50 trees per word on average if using a lexicon the size of our canonical lexicon).

In our parser implementation, we use **supertagging** to select only the best prediction trees in each step, which reduces the search space considerably. Supertagging (Bangalore and Joshi 1999) is a common approach used in the context of TAG and CCG parsing; the idea is to limit the elementary trees for each word to those that are evaluated highly by some shallow statistical model. We only use supertagging for prediction trees; for canonical trees, we use all (lexicalized) trees that the grammar contains for the word (rare words are replaced by "UNK").

Because our parser must run incrementally, the supertagger should not be allowed to have any look-ahead. We found, however, that not having any look-ahead has a detrimental impact on supertagging quality, and subsequently on parsing accuracy. We therefore allow the supertagger a restricted look-ahead. The supertagger estimates the usefulness of a prediction tree $\tau_{predict}$ based on a current fringe of the prefix tree f_p and the POS tag of the next word $t_{w_{i+1}}$. Note that the parser has at that point already read and fully processed word w_i, and its task is to choose possible prediction trees for the prediction step just before processing word w_{i+1}. Knowing the POS tag of that next word w_{i+1} does not compromise incrementality if the POS tag is determined without any further look-ahead.[3] It does, however, make the interpretation of prediction weaker: Predictions for maintaining a fully connected structure are only made once the identity of the next word is known and its POS tag has been estimated.

The probability model of the supertagger is parametrized as follows:

$$\sum_{\tau_{predict}} P(\tau_{predict} | f_p, t_{w_{i+1}}) = 1 \tag{11}$$

$$\text{where } P(\tau_{predict} | f_p, t_{w_{i+1}}) = P(\tau_{predict} | f_{predict}, sl_{predict}) P(f_{predict}, sl_{predict} | f_p, t_{w_{i+1}})$$

In order to reduce data sparsity, we independently estimate the probability of a particular prediction tree given its first fringe $f_{predict}$ and category of the leaf node on the spine $sl_{predict}$, and the probability of some tree with first fringe $f_{predict}$ and category of the leaf node on the spine $sl_{predict}$ given a prefix tree with current fringe f_p and estimated POS tag of the next word $t_{w_{i+1}}$. A further simplification is that we represent the current fringes $f_{predict}$ and f_p as an alphabetically ordered set of the categories occurring on it. The reasoning behind this decision is that the order of nodes is less important than the identity of the nodes as possible integration sites. The supertagging model is smoothed with the procedure described by Brants (2000), as it yielded better results than Witten-Bell smoothing (which suffers from data sparsity in the supertagging task). We use one level of back-off where we estimate $P(f_{predict}, sl_{predict} | f_p, t_{i+1})$ based only on the most likely integration site n_p instead of the whole fringe f_p:

$$\max_{n_p} P(f_{predict}, t_{predict} | n_p, t_{w_{i+1}}) \tag{12}$$

The reason for backing off to the most probable integration site is that a fringe with more unique categories should not have a lower probability of a particular tree adjoining into it than a fringe containing the same category, but fewer other categories.

5. Treebank Conversion and Lexicon Acquisition

For lexicon induction and parameter estimation, our parsing model requires training data, which we obtain by converting a standard treebank for English into TAG format. The procedures used are based on well-established approaches in the TAG literature (in particular Xia, Palmer, and Joshi 2000); we will therefore only give a brief overview

[3] In the case of the PLTAG parser, we first retrieve the elementary trees for the upcoming lexeme. If the word occurred with more than one POS tag, we choose the POS tag with highest conditional probability given the previous two POS tags.

of treebank conversion and lexicon induction here; the reader is referred to Demberg-Winterfors (2010) for full details.

Our PLTAG lexicon (both canonical trees and prediction trees) is derived from the Wall Street Journal section of the Penn Treebank, complemented by noun phrase annotation (Vadas and Curran 2007), and Propbank (Palmer, Gildea, and Kingsbury 2003), as well as a slightly modified version of the head percolation table of Magerman (1994). These additional resources are used to determine the elementary trees for a TAG lexicon, following the procedures proposed by Xia, Palmer, and Joshi (2000). This involves first adding noun phrase annotation to the Penn Treebank, and then determining heads with the head percolation table, augmented with more detailed heuristics for noun phrases.[4] As a next step, information from Propbank is used to establish argument and modifier status and to determine which lexical items should be encoded in the same elementary tree (currently, this is restricted to particle verbs like *show up* and some hand-coded constructions in which the first part is predictive of the second part, such as *either* ... *or* or *both* ... *and*). Then we remove quotation marks, brackets, sentence-final punctuation, and some of the traces from the Treebank. Finally, we heuristically insert more structure into flat quantifier phrases, adding explicit right branching and additional nodes wherever adjunction is possible. Furthermore, auxiliaries are assigned a special POS tag in order to enable the lexicon induction algorithm to extract them as auxiliary trees. Deviating from the standard TAG analysis, copula verbs are treated in PLTAG as subcategorizing for two NPs and are therefore marked as copula verbs during treebank conversion.

Given the converted trees, which are less flat and contain information about headedness and the argument/modifier distinction, we extract the canonical lexicon by traversing the converted tree from each leaf up towards the root, as long as the top node is the head child of its parent. If a subtree is not the head child of its parent, we extract it as an elementary tree and proceed in this way for each word of the converted tree. Given the argument/modifier distinction, we then create substitution nodes in the parent tree or a root and foot node in the child tree.

5.1 Creating the Prediction Tree Lexicon

After converting the treebank and extracting the canonical lexicon as described previously, the next step is to generate a prediction tree lexicon. The prediction tree lexicon contains the prediction trees needed to derive the treebank sentences using PLTAG. Remember that prediction trees are usually not lexicalized, and can in principle have any (tree) shape. Only those prediction trees that are verifiable with a canonical tree can yield valid PLTAG derivations, however.

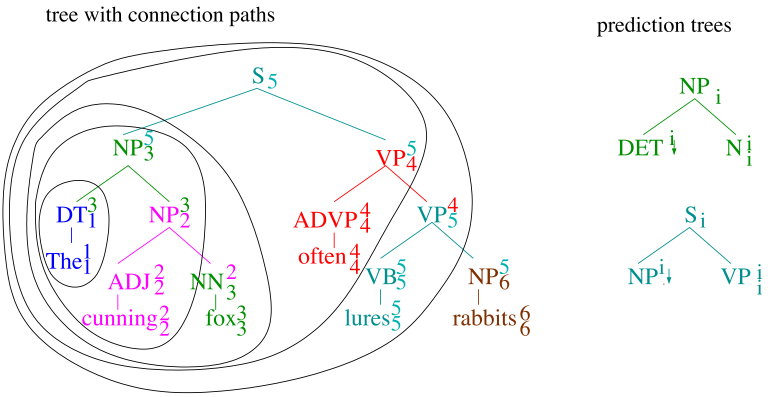
As we explained in Section 3.4, verifiable prediction trees are prediction trees that are identical to canonical TAG trees from the lexicon, except subtrees below and to the right of nodes on the spine may be missing. Other prediction trees are technically allowed in the lexicon, but they can never be used in successful derivations. When building prediction trees from canonical trees by removing subtrees at the bottom and to the right, we must make a choice about what and how many subtrees are removed. This design choice has implications on the granularity of prediction in the parser: For

[4] The head percolation table and the code for the NP heuristics are available at `http://www.coli.uni-saarland.de/~vera/page.php?id=corpora`. The PLTAG-converted version of the Penn Treebank is also available from this URL.

example, if substitution sites to the right of the spine are included in a verbal prediction tree, then this amounts to predicting the subcategorization frame of a verb we have not yet seen. As there is not enough psycholinguistic evidence to determine the correct level of granularity of prediction, we decided to take a conservative view, that is, we do not predict any substitution nodes to the right of the spine. Similarly, there is only psycholinguistic evidence for lexical prediction in very constrained contexts, therefore we do not usually predict unary nodes at the bottom of the spine, and only predict lexical anchors for a limited set of constructions, for example, *either … or*, which has been explicitly implicated in prediction in experimental studies (see Section 2). For future work, we plan to extend the prediction of lexical anchors to predictable parts of collocations and idioms.

Prediction trees are learned directly from the converted version of the Penn Treebank by calculating the connection path (Mazzei, Lombardo, and Sturt 2007) at each word in a tree. A connection path for words $w_1 \ldots w_n$ is the minimal amount of structure that is needed to connect all words $w_1 \ldots w_n$ into the same syntactic tree. We use the connection paths and the knowledge of how the sentence tree can be decomposed into the canonical trees to determine which parts of the structure need to be included in the connection path for $w_1 \ldots w_n$, but are not part of any of the elementary trees with anchors $w_1 \ldots w_n$. These remaining nodes need to be contained in a prediction tree (see Figure 9, in which prediction trees are necessary for connectedness at words 2 and 4).

The nodes necessary to achieve connectedness can belong to one or more elementary trees with anchors beyond w_n. Because we restricted our parser to only use one prediction tree per step, we need to generate a "pre-combined" prediction tree whenever nodes originating from more than one elementary tree are needed to achieve connectedness. For example, if we changed the sentence from Figure 9 to *the cunning fox very often lures rabbits*, we would have to predict nodes from both the adverb tree and the verbal tree when processing *very*, resulting in the extraction of a pre-combined prediction tree, as shown in Figure 10. Nodes originating from different elementary trees have distinct markers to indicate which of them should be verified in a single step. In particular, the nodes at the integration site of two prediction trees has two different markers, as its upper and lower half originate from different elementary trees. In our example, the two VP nodes in Figure 10 have different top and bottom indices because the auxiliary prediction tree based on the auxiliary tree for *often* adjoined into VP node of the prediction tree, which is generated from the *lures* elementary tree.


tree with connection paths

prediction trees

Figure 9
Illustration of connection paths and prediction trees generated from the syntactic tree.

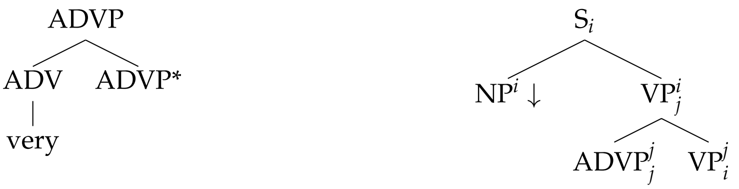


Figure 10
Lexicon entry for *very* and the precombined prediction tree that would be needed in order to integrate the word *very* with the prefix *the cunning fox*.

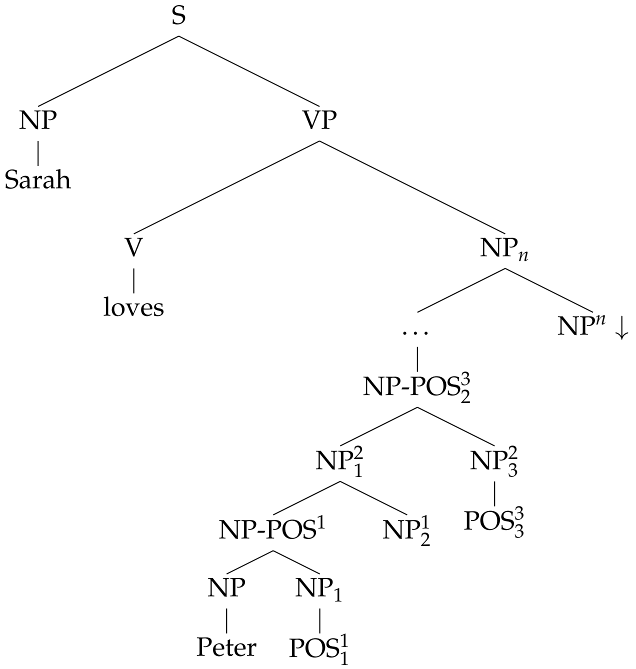


Figure 11
Left recursion: Prefix tree (after integrating *Peter*) needed to parse the sentence *Sarah loves Peter's father's . . . books*.

There are cases where more than two prediction trees will have to be combined, and in principle, due to left recursion, the number of prediction trees that are needed to form a precombined prediction tree is unbounded. Consider the example of parsing the sentences *Sarah loves Peter's books*, *Sarah loves Peter's father's books*, *Sarah loves Peter's father's neighbor's books*, and so on; a schematic example is shown in Figure 11. Before integrating *Peter*, we would need to predict a precombined prediction tree for every possible embedding depth of *Peter*, thus requiring an infinitely large prediction tree lexicon.[5] We therefore restrict our prediction tree lexicon to precombined prediction trees that we have observed during training. This bounds the embedding depth that

[5] Note that such left-recursion examples could be handled in alternative ways, for example, by changing them to right-recursive structures (as suggested in Mazzei 2005). Using right-recursive structures to avoid problems with left-recursion, however, leads to elementary trees that are not well motivated linguistically.

the induced grammar can use. The grammar can still reconstruct all derivations in the training corpus incrementally, however.

Note also that the Penn Treebank does not contain any sentence for which more than five prediction trees need to be pre-combined in order to allow for incremental processing—and there were just four instances of such large pre-combined prediction trees; more than 99% of prediction trees extracted from Penn Treebank are based on three or fewer elementary trees.

5.2 Lexicon Extraction Statistics

Our conversion algorithm extracted 6,700 tree templates from Sections 2–21 of the Penn Treebank.[6] The grammars extracted by Chen (2001) and Xia, Palmer, and Joshi (2000) are of similar size: They extracted 3,000–8,600 tree templates, depending on parameters such as number of categories, treatment of traces/null elements, punctuation, head percolation rules, and modifier–argument distinction. Our lexicon is fairly large as it contains traces, null elements, and sentence-internal punctuation. Furthermore, it contains some trees with several anchors, to account for constructions such as for *either ... or* or particle verbs like *show ... up*. In such trees with more than one anchor, all but the leftmost anchor are predicted (see Demberg-Winterfors [2010] and Kaeshammer and Demberg [2012] for more details). The size of the prediction tree lexicon we extracted is 2,595 trees.

A smaller grammar was extracted by Chiang (2000) (2,104 tree templates). The smaller lexicon can be attributed to the decision to use sister adjunction instead of normal adjunction. Chiang furthermore removes traces and null elements, and does not extract multi-anchored trees. Smaller lexicons lead to less data sparsity and faster parsing performance, as there is less ambiguity between the elementary trees.

There were some sentences in the Penn Treebank that we could not automatically convert to PLTAG format. This affected about 1.8% of sentences. Failure to convert sentences was due to incorrect annotation in the original treebank, as well as fragmentary or ungrammatical sentences. In 0.4% of cases, conversion failure was due to a modifier occuring between two arguments. Similar rates of unsuccessful conversion have been reported, for example, by Hockenmaier and Steedman (2007).

5.3 Features and Parameter Estimation

As explained in Section 3.6, we approximate the probability distributions for substitution, adjunction, and verification using features of the prefix tree, the elementary tree, and the integration site. These features are shown in Table 1.

The presentation of the features relies on the notation from Sections 3.5 and 3.6. The node u_p is the node in the prefix tree τ_p at which the elementary tree τ_e was integrated; u_r is the root of the elementary tree that introduced u_p, and $\mathtt{head}(u_r)$ is the head of that node. There are features that use the lexeme at the node $\mathtt{head}(u_r)$ directly, if it is a lexical leaf; other features use the syntactic category of the parent of $\mathtt{head}(u_r)$, which will typically be the part-of-speech tag of that word. If u_p was introduced by an unlexicalized prediction tree, $\mathtt{head}(u_r)$ may not be a lexical leaf (an example is the second tree in

[6] A tree template is an elementary tree without its lexical anchor (e.g., [DT the] and [DT a] belong to the same template).

Table 1
Features used in the PLTAG parser with back-off levels.

feature for $P(\mathsf{unlex}(\tau)\|\tau_p, u_e, u_p)$ (used in subst. and adj.)	l1	l2	l3	l4	l5	l6
tree template that contributed u_p	+	+	+	+	−	−
lexeme of $\mathsf{head}(u_r)$	+	−	−	−	−	−
syntactic category of $\mathsf{head}(u_r)$	+	+	+	−	−	−
position of the integration site within its elementary tree	+	+	+	+	−	−
syntactic category of the integration site	+	+	+	+	+	−
is the beginning or end of the current fringe a trace?	+	+	−	−	−	−
category of the leftmost leaf below u_p	+	+	+	−	−	−
category of the rightmost leaf to the left of u_p	+	+	+	−	−	−
if adjunction: position of integration site among alternatives	+	+	+	+	−	−

feature for $P(\mathsf{unlex}(\tau_v)\|\tau_{predict})$ (used in verification)	l1	l2	l3
tree template of matched prediction tree $\tau_{predict}$	+	+	−
is the beginning or end of the current fringe a trace?	+	−	−

feature for $P(\mathsf{anchor}(\tau)\|\mathsf{unlex}(\tau), \mathsf{head}(u))$	l1	l2	l3	l4
tree template, $\mathsf{unlex}(\tau)$	+	+	+	−
syntactic category of τ's anchor node	+	+	+	+
lexeme of $\mathsf{head}(u)$	+	−	−	−
syntactic category of $\mathsf{head}(u)$	+	+	−	−

Figure 4); in this case, we use the syntactic category of $\mathsf{head}(u_r)$ both for the lexeme and the category (in the example, this is "VP").

Two of the features traverse the entire prefix tree to find the leftmost leaf below u_p and the leaf directly to its left. If u_p dominates the first leaf of the prefix tree, then there is no leaf directly to the left; in this case, the second feature takes a null value. In the case of adjunctions, one feature inspects the position of the integration site among all other nodes in the tree that could have been used as adjunction sites for the same auxiliary tree. This allows us to distinguish high and low attachment.

The probability models are now obtained via maximum likelihood estimation from the training data. Many of the substitution and adjunction events are seen rarely or not at all with their full contexts, which indicates the need for smoothing. We use back-off with deleted interpolation, as detailed in Table 1. The weight for each of these contexts is automatically determined by a variant of Witten-Bell smoothing, which calculates a weight for each of the back-off levels for each context (Witten and Bell 1991). We implemented the version described by Collins (2003). For the verification operation, data sparsity for the probability of the tree template τ_v is less of an issue because the probability of a tree template verifying a prediction tree is conditioned only on the identity of the prediction tree and the trace feature.

6. Evaluation

In order to compare the PLTAG parser to other probabilistic parsers, we evaluated parsing accuracy on the Penn Treebank (PTB). We first converted the PTB into a PLTAG treebank as described in Section 5. We then trained the parser on Sections 2–21 of the Penn Treebank and evaluated it on Section 23; only sentences of length 40 or less were used for evaluation. It is important to note that because the parser is trained and evaluated on a converted treebank, its accuracy is not directly comparable to parsers that run directly on the PTB. We will discuss this point in more detail in Section 6.1.

For the results reported here, the beam width of the parser was set such that all analyses whose log probability is less than the log probability of the best analysis minus 8 are removed (our implementation uses the natural logarithm). The beam width was set using the development set (Section 0 of the PTB). Furthermore, only the 250 best analyses in a chart entry are maintained. The supertagger was set to select the best 20 prediction trees at each step.

We found that parsing accuracy is higher when using a probability model that does not factor in the *NONE*-adjunction events (parsing accuracy decreases by about 1.5 percentage points in a model that takes these events into account). We believe that this decrease in parsing performance is due to the low probability of adjunction operations relative to substitution operations, which is a result of the normalization with *NONE* events. The high occurrence of *NONE* adjunctions in our parser is a consequence of the PLTAG conversion procedure, which leads to trees with more nodes overall, and hence many nodes at which no adjunction happens. In what follows we will present results for parsing with the probability model that does not include *NONE*-adjunction.

Coverage. We first evaluated the coverage of our PLTAG parser on the test set. The parser found valid parses for 98.09% of the sentences in Section 23 within reasonable memory usage (2 GB RAM). The reason for failing to parse a sentence can be that all valid parses have fallen out of the beam, that an essential prediction tree was not selected by the supertagger, or that no parse can be derived given the PLTAG lexicon acquired during training.

A way of dealing with out-of-coverage sentences is to return a flat structure in which all words are attached directly to the root node (Roark 2001). This way a coverage of 100% is obtained, which facilitates the comparison between parsers. We also give results for such a "full coverage" version of our parser in the following.

Parsing Accuracy. Table 2 gives the parsing results for the variants of the PLTAG model that we evaluated. The full PLTAG probability model as described in Section 3.6 achieved an F-score of 79.41 with Witten-Bell smoothing, given the gold-standard POS tags. When gold-standard POS tags are given, the algorithm only retrieves elementary trees for a word which includes the correct POS tag, whereas it retrieves all elementary trees for a word, and hence has a larger search space, when no gold-standard POS tags are given. Without gold-standard POS tags, parsing performance drops to an F-score of

Table 2
Parsing results for the PLTAG parser with gold standard POS tags. Prediction tree oracle = correct prediction tree provided; No gold POS = PLTAG parser with no gold-standard POS tags provided; full cov = flat structure returned for out-of-coverage sentences.

Model	Precision	Recall	F-score	Coverage	F-score (full cov)
PLTAG parser	79.43	79.39	79.41	98.09	78.65
No gold POS	77.57	77.24	77.41	98.09	76.64
Prediction tree oracle	81.15	81.13	81.14	96.18	80.08

77.41. Table 2 also gives the F-scores for the full coverage version of the parser, which are about 0.8 points lower.

Error Analysis. An aspect in which our parser fundamentally differs from other parsers is its use of prediction trees. We evaluated the impact of using prediction trees on parsing performance, by running the parser in an oracle scenario where only the correct prediction trees were given (the parser is, however, not forced to use them; the probability model can still assign a low probability to an analysis with the correct prediction tree). Table 2 includes the result for the prediction tree oracle condition. In this condition, we observed an increase of 1.7 points in F-score; the modest improvement indicates that our supertagger selects good prediction trees most of the time. The full coverage version of the oracle parser performs about one point worse than the standard version due to lower coverage in the oracle condition (the lower coverage in the oracle condition is caused by the fact that no other than the correct prediction trees can be used). We also tested the coverage of the prediction tree lexicon on our test data: In order to obtain the correct parse, a prediction tree that the parser has observed less than five times during training is needed in 1.6% of all instances where some prediction tree is needed.

It is important to note that not using the supertagger slows down the parser considerably. In fact, it is not possible to run the parser without the supertagger on a 16 GB RAM machine without making the beam so small that it is no longer useful. In order to nevertheless evaluate the loss in accuracy through supertagging of prediction trees, we ran the parser with and without supertagger on WSJ10, the portion of the test set that includes only sentences of 10 words or less. The F-score on WSJ10 was 85.82 without the supertagger, and 85.18 with the supertagger, indicating that supertagging for prediction trees incurs a small loss in parsing accuracy compared to full parsing.

We then compared for each sentence of the test set the probability of the best analysis with the probability of the correct analysis and found that in 55% of the cases, the probability model assigns a higher probability to an incorrect analysis, meaning that a large proportion of the errors can be attributed to the probability model. In the remaining 45% of cases, the best analysis found by the parser has a lower probability than the correct one. This could be due to an incomplete lexicon, supertagging errors, or the beam being too narrow. We first quantified the loss in parsing accuracy arising from missing lexicon entries: We compared parser performance for a version of the lexicon obtained from the training data and a version of the lexicon that additionally included all trees needed to parse a test sentence. This only improved parsing F-score by 0.5 points.

Supertagging errors are the next potential cause for the parser returning an analysis that has a lower probability than the correct one. Our analysis, however, based on a prediction tree oracle showed that these errors are a fairly minor problem, leading to a decrease of 1.7 points in overall F-score. Therefore, it seems likely that a large proportion of errors is due to the correct analysis falling out of the beam during parsing. Although our beam width is similar to the beam widths used in other parsers, it may be insufficient for a PLTAG parser. The reason for this lies in the strict incrementality: The role of the prediction trees is essentially to list all the possible ways in which words can be combined, whereas parsers that keep unconnected treelets can wait for more evidence before connecting all parts; the space of analyses for a non-incremental parser hence grows more slowly.

To further investigate the performance of the probability model, we evaluated the probability model in terms of attachment decisions. We provided the parser with the

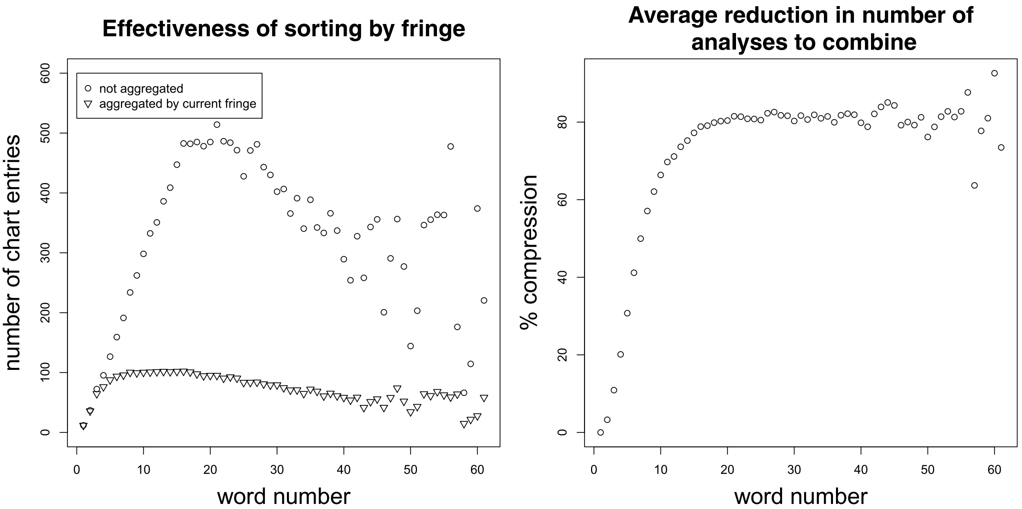


Figure 12
Aggregating different analyses by their current fringe significantly reduces the number of tree combination operations that need to be performed at each word.

correct elementary trees, which amounts to assuming perfect supertagging for all trees (not just for prediction trees). The F-score given perfect supertagging is 93.7%, indicating that the best analysis often does not get the largest probability: in 41% of sentences, an incorrect analysis gets assigned a higher probability than the correct analysis.[7]

Effectiveness of Chart Parsing. As described in Section 4.2, partial analyses are aggregated by their current fringes. This significantly reduces the number of tree combinations that have to be calculated at each step, as shown in Figure 12. Aggregation only becomes relevant after the first few words, but then the benefit of aggregating all analyses with the same current fringe becomes substantial, reducing the number of tree combination calculations by 80% on average with the beam search settings described here. The benefit of this strategy can be expected to be even larger with less rigorous pruning.

6.1 Comparison to Other Parsers

The PLTAG parser is trained and evaluated on a version of the Penn Treebank that was first converted to a PLTAG format. This means that our results are not directly comparable to parsers that reproduce the Penn Treebank bracketing, as our parser produces deeper tree structures informed by Propbank and the noun phrase annotation of Vadas and Curran (2007). It is not trivial to convert PLTAG structures back to PTB structures. In lieu of a full conversion, we flattened both our PLTAG structures and the original PTB structures such that a node cannot have a child with the same non-terminal symbol. For an example, see Figure 9, where an NP is a child of another NP, and a VP is the child of another VP. In the flattened representation, the NP under S has the children

[7] Note that this figure is slightly lower than the 45% reported earlier, as the experiment uses perfect lexical trees for parsing, rather than the actual lexicon derived from the training set.

DT, ADJ, and NN, and the VP under S has the children ADVP, VB, and NP instead of ADVP and VP. We then evaluated the flattened output of our parser against the flattened version of the original PTB test set (rather than using the PLTAG version of that test set). For comparison, we applied the same procedure to the output of the Charniak parser (chosen to represent a typical PTB parser), that is, we flattened its output and evaluated it against the flattened version of the PTB test set. We found that F-score decreased by two points in both cases.

In the following, we compare to other TAG parsers only; but even these comparisons should be taken with a grain of salt because these still differ in which variant of the formalism they use (Lexicalized TAG, Spinal TAG, Incremental TAG). Table 3 gives the F-scores of existing TAG parsers and compares them along the hierarchy of psycholinguistic requirements (incrementality, connectedness, prediction; see Section 1). The formalism that comes closest to ours in terms of psycholinguistic properties is that of Mazzei, Lombardo, and Sturt (2007), which has not been implemented or evaluated on corpus data, however. Numerically, our results are comparable to those of Kato, Matsubara, and Inagaki (2004). Their parser is incremental and builds connected structures, but makes strong simplifying assumptions, such as failing to distinguish modifiers and arguments. It uses gold POS tags as input and is not lexicalized. The parsers of Chiang (2000) and Shen and Joshi (2005) achieve higher F-scores, but at the cost of not being incremental (Chiang 2000) or not building connected structures (Shen and Joshi 2005). Furthermore, Shen and Joshi use a look-ahead of two words, which significantly weakens their incrementality claim. Note also that their F-score is measured on dependencies rather than labeled bracketing.

6.2 Discussion

Differences in performance with other TAG parsers are likely due to the incrementality restriction (incremental parsers generally have slightly lower performance), not doing any supertagging for canonical trees, a large lexicon, and data sparsity in the probability model. The latter is due to the large lexicon and the additional parser operation of verification. A further effect of having separate prediction and verification steps is that many bilexical dependencies are lost when prediction trees are integrated. Because prediction trees are not necessarily lexicalized, the statistical model cannot condition them on lexemes. To make up for this, it would be necessary to take into account bilexical dependencies at verification time, but this is not part of the current probability model. An improvement in parsing performance is likely to result from addressing this shortcoming.

Table 3
Comparison of this work with other TAG parsers. impl = implemented model; incr = incrementality; con = connectedness; pred = prediction; F = F-score; Shen and Joshi (2005) evaluate on dependencies and use a look-ahead of two words.

Model	incr	con	pred	impl	F
Mazzei, Lombardo, and Sturt (2007)	+	+	+	−	n/a
This work (gold POS)	**+**	**+**	**+**	**+**	**78.65**
Kato, Matsubara, and Inagaki (2004)	+	+	−	+	79.65
Shen and Joshi (2005)	(+)	−	−	+	(87.4)
Chiang (2000)	−	−	−	+	86.7

7. Psycholinguistic Evaluation

Although we have focused on the computational issues of PLTAG parsing so far, a key motivation behind our incremental parser is to develop a more realistic model of human language processing. A treebank-based evaluation as in the previous section does not directly provide evidence of psycholinguistic validity; however, a parser with good coverage and high parsing accuracy is a prerequisite for an evaluation on eye-tracking corpora, which Keller (2010) argues are the benchmark for models of human sentence processing. In what follows, we report an evaluation study that uses our PLTAG parser to predict human reading times, and compares its performance on this task to a standard model based on surprisal (Hale 2001).

Surprisal assumes that processing difficulty is associated with expectations built up by the sentence processor: A word that is unexpected given its preceding context is harder to process. Mathematically, the amount of surprisal at word w_i can be formalized as the negative logarithm of the conditional probability of w_i given the preceding words in the sentence $w_1 \ldots w_{i-1}$:

$$\begin{aligned} \text{Surprisal}_{\text{w}_\text{i}} &= -\log P(w_i|w_1 \ldots w_{i-1}) \qquad (13) \\ &= -\log \frac{P(w_1 \ldots w_i)}{P(w_1 \ldots w_{i-1})} \\ &= -\log \sum_{\tau_{p_{w_1 \ldots w_i}}} P(\tau_{p_{w_1 \ldots w_i}}) + \log \sum_{\tau_{p_{w_1 \ldots w_{i-1}}}} P(\tau_{p_{w_1 \ldots w_{i-1}}}) \end{aligned}$$

Here, $P(\tau_{p_{w_1 \ldots w_i}})$ is the probability of the prefix tree $\tau_{p_{w_1 \ldots w_i}}$ that spans the words $w_1 \ldots w_i$. If the surprisal at word w_i is high, then w_i should be difficult to process. This manifests itself in elevated reading times, for example, in eye-tracking data (Boston et al. 2008; Demberg and Keller 2008a; Frank 2009).

Surprisal can be estimated in two different ways: as lexical surprisal and structural surprisal, following Demberg and Keller (2008a). We calculated lexical surprisal using the prefix probabilities returned by the incremental probabilistic parser of Roark (2001). Lexical surprisal takes into account the lexical items that make up a sentence prefix, and is thus influenced by word frequency and by the probability of a word being assigned a specific part of speech. This quantity will be referred to as the factor LEXICALSURPRISAL in the following. This can be contrasted with structural surprisal (factor STRUCTURALSURPRISAL), which uses unlexicalized parses, and is based only on the probability of the syntactic structures assigned to a sentence prefix. Following Demberg and Keller (2008a), we replaced each word in the training corpus with its part-of-speech tag and then trained the Roark parser on this version of the corpus. The unlexicalized parser obtained this way was run on the Dundee corpus and prefix probabilities were obtained in the usual way to compute unlexicalized surprisal scores.

7.1 Estimating Processing Difficulty using PLTAG

Prediction Theory was proposed by Demberg and Keller (2008b, 2009) as a model of processing difficulty in human parsing based on PLTAG. Prediction Theory, like surprisal, returns word-by-word difficulty scores that can be tested against human reading times. Unlike surprisal, however, Prediction Theory aims to formalize how predictions about upcoming linguistic structure are made by the sentence processor.

To achieve this, it incorporates an explicit prediction and verification process, as well as memory decay.

Prediction Theory estimates of processing difficulty can be obtained straightforwardly from the PLTAG parser presented in this article. The Prediction Theory framework therefore directly addresses the psycholinguistic desiderata of incremental, connected, and predictive processing that we argued for in Section 2. In addition, Prediction Theory incorporates a linking theory that relates model quantities to behavioral data (Demberg and Keller 2009). Specifically, there are two components that account for processing difficulty in Prediction Theory. Firstly, surprisal is used to quantify difficulty in terms of updates to the parser's representation of possible analyses as the sentence unfolds. Surprisal is calculated from the probability distribution over prefix trees spanning the input as defined in Equation (13). This component of the model is equivalent to Hale's surprisal model, except that prefix tree probability are computed using the probabilistic PLTAG parser proposed in this article, rather than a PCFG parser.

The second component of processing difficulty in Prediction Theory is verification cost, which arises when previously predicted structures are checked against what is actually encountered in the input. Verification cost depends on the probability of the original prediction tree $\tau_{predict}$ that is to be verified, as well as on how much this prediction has decayed in memory:

$$\text{VerificationCost}_{w_i} = -\log \sum_{\tau_{predict}} P(\tau_{predict})^{(1-d^{t-t_{\tau_{predict}}})} \tag{14}$$

Here, t is the time at which the verification takes place and $t_{\tau_{predict}}$ is the time at which $\tau_{predict}$ was last accessed. For simplicity, we assume that t is the index associated with w_i, that is, $t = i$ (more sophisticated approaches that take into account word length are possible). By analogy, $t_{\tau_{predict}}$ is the index of the word that was processed when $\tau_{predict}$ was retrieved from the prediction tree lexicon and integrated into the prefix tree. The decay constant d is a parameter of the model, which we set to 0.9 (the results reported subsequently are stable across a large range of values of d).

The motivation for the verification cost component of Prediction Theory comes from a large body of psycholinguistic evidence that shows that human sentence processing incurs processing difficulty when new material has to be integrated into an existing representation (for a review of this evidence, see Gibson [2000]; Lewis and Vasishth [2005]). Integration cost occurs when the processor integrates a syntactic argument with the head that the argument depends on (e.g., an NP argument is integrated with a verbal head that has already been processed). The cost of this integration (e.g., measured as elevated reading time) has been shown to depend on the type of integration and on the distance between the argument and its head (see Gibson [1998], and much subsequent work). Prediction Theory verification cost models this in terms of prediction difficulty (i.e., the probability of the predicted element) and memory decay (i.e., distance between prediction and verification). In the following analyses, Prediction Theory scores are represented by the factor PREDICTIONTHEORY, and are computed as the sum of the surprisal in Equation (13) and the verification cost in Equation (14).

To summarize, the central aim of Prediction Theory is to unify two types of processing difficulty: The cost of updating syntactic representations (surprisal) and the cost of integrating predicted structure (verification cost). These two components have so far been observed and modeled separately in the psycholinguistic literature (Demberg and Keller 2008a; Staub 2010). Given its unifying nature, Prediction Theory can be expected to capture a wider range of experimental results than surprisal alone. In the following, we will test this claim by evaluating Prediction Theory on the reading times in the Dundee eye-tracking corpus.

7.2 Method

In order to test whether Prediction Theory or surprisal scores correlate with reading times we use linear mixed effects (LME) models (Pinheiro and Bates [2000]). These models can be thought of as a generalization of linear regression that allows the inclusion of random factors (such as participants or items) as well as fixed factors (e.g., word frequency or surprisal). The fixed factors can be discrete (such as whether the previous word was fixated) or continuous (such as word frequency). When reporting mixed models, we normally give the estimates of the coefficients β of the fixed factors included in the model; these can be interpreted as the weights of the factors in the model (though only coefficients of factors on the same scale can be compared directly). In addition, each coefficient is associated with a standard error (SE), which expresses the amount of variation in the estimate of that coefficient, and a t-value, which indicates whether the coefficient is significantly different from zero. For the model as a whole, we can measure the log-likelihood, which is an indicator of how well the model fits the data. Two models can be compared by testing whether their log-likelihood values are significantly different.

All predictors in our LME models were centered (i.e., the mean value of the predictor was subtracted from each instance) to reduce collinearity. We treat participants as a random factor, which means that our models contain an intercept term for each participant, representing the individual differences in the rates at which they read. Furthermore, we include a random slope for the predictor of interest (Prediction Theory and surprisal difficulty, respectively), essentially accounting for idiosyncrasies of a participant with respect to the predictor of interest, such that only the part of the variance that is common to all participants and can be attributed to that predictor.[8]

We built the models by initially including all predictors as fixed factors, as well as all binary interactions between them. To obtain a minimal model, we then performed stepwise removal of fixed factors that did not significantly improve model fit (based on a χ^2-test comparing model log-likelihoods), starting with the factor that improved model fit least. This procedure ensures that a model is obtained that achieves the best fit to the data with the minimum number of predictors. In the following, results tables give the coefficients and significance levels for those predictors that remained in the minimal models. No model selection was performed on the random effect structure, which always included PARTICIPANT as random intercept, as well as random slopes under PARTICIPANT for factors of theoretical interest (LEXICALSURPRISAL, STRUCTURALSURPRISAL, PREDICTIONTHEORY).[9]

Before fitting LME models, we applied outlier removal: We computed the mean reading time (over all items and participants), and then removed all data points that deviated more than two standard deviations from the mean. This led to a loss of roughly

[8] Other random factors that are appropriate for our analyses are word and sentence; due to the large number of instances for these factors, however, we were not able to include them: The model fitting algorithm we used (implemented in the R package LME4) does not converge for such large models.

[9] Model selection on random effects has been shown in simulation studies to be anti-conservative (Barr et al. 2013). Note that the models reported here are updated in this respect compared to the results in Demberg-Winterfors (2010).

4% of data points. Outliers can strongly influence the results of analyses on the Dundee corpus, as Roland et al. (2012) show. Furthermore, this way of trimming the data also reduces the long tail of the reading time distribution, resulting in a distribution that is closer to normal.

Data. For our psycholinguistic evaluation, we used the English portion of the Dundee eye-tracking corpus (Kennedy and Pynte 2005), which contains 20 texts taken from *The Independent* newspaper. The corpus consists of 51,502 tokens and 9,776 types in total. It is annotated with the eye-movement records of 10 English native speakers, who each read the whole corpus. The eye-tracking data was preprocessed following the methodology described by Demberg and Keller (2008a). From this data, a range of reading time measures can be computed for each word in the corpus. Here, we will only discuss first-pass times (other measures give similar results). First-pass times are defined as the sum of the duration of fixations from first entering the word from the left to leaving it (only cases where no later words have been fixated are counted). Our statistical analyses are based on actual reading times, and so we only included words that were not skipped. Furthermore, all data points for which one of the values used in the regression is missing (e.g., launch distance is not available for data points that are the first fixation on a line or after track loss; Prediction Theory scores are not available for sentences that could not be parsed, about 1.5% of data) are discarded for regression analysis. The resulting data set for first pass times consisted of 159,378 data points.

7.3 Results and Discussion

The simplest evaluation is to fit an LME model in which the measure of interest is included as a single predictor for observed reading times (including in the model also a random intercept per subject and a random slope for LEXICALSURPRISAL or PREDICTIONTHEORY, respectively, under subject). Under this approach, we find that both LEXICALSURPRISAL and PREDICTIONTHEORY are significant positive predictors of first pass reading times ($\beta = 4.9$; $t > 9$; $p < 0.001$ and $\beta = 7.8$; $t > 9$; and $p < 0.001$, respectively). PREDICTIONTHEORY provides better fit than LEXICALSURPRISAL, as evidenced by the significantly larger log likelihood of the model. STRUCTURALSURPRISAL is not a significant predictor of reading times in such a model (and the log likelihood of a model including STRUCTURALSURPRISAL is even lower).

It is possible, however, that the effect of a single predictor of interest can be explained away by simpler predictors that are not of theoretical relevance, but are highly correlated with reading times. Therefore, we fitted full models that included predictors that relate to properties of the word, like word length in characters (WORDLENGTH), word frequency in log scale (WORDFREQUENCY), as well as word position in the sentence (WORDNOINSENTENCE). In order to account for spill-over effects, where processing difficulty on one word influences processing difficulty on the next word, we also include the frequency of the previous word (PREVWORDFREQ) and a binary flag indicating whether the previous word was fixated (PREVWORDFIXATED). Furthermore, we include predictors that relate to the reading process, such as the launch distance of the eye-movement in characters (LAUNCHDISTANCE), the squared relative landing position of a fixation on a word in characters (LANDINGPOSITION), and bigram probability (BIGRAMPROB), which can be regarded as a simple version of surprisal. In addition, we included the binary interactions between these predictors. We found that the interaction between word length and word frequency (WORDLENGTH:FREQUENCY), as well as between word length and landing position (WordLength:LandingPos) significantly improved model fit.

Some of these predictors are correlated with the measures of interest, leading to collinearity in the models which cannot be removed by centering. Specifically PredictionTheory is correlated with WordLength, WordFrequency, BigramProb, and PrevWordFreq after centering. We therefore residualize PredictionTheory against these predictors. The residualized predictor ResidPredictionTheory then only accounts for that part of the prediction theory score that cannot be explained in terms of word frequency, bigram probability, and so forth. In analogy with this, we obtained ResidLexicalSurprisal by residualizing lexical surprisal against WordLength, WordFrequency, and BigramProb. ResidBigramProb was residualized with respect to WordFrequency, and ResidStructuralSurprisal was residualized against BigramProb. All remaining correlations are smaller than 0.06, indicating an acceptably low level of collinearity. All models reported in Table 4 include random slopes for the predictor of interest. Whenever possible (no failure to reach convergence or indicated otherwise through increased correlation in fixed effects which points to an overspecification in random slopes), we also added a random slope for WordLength.

Table 4 gives the final LME models for first pass times for the predictors of interest: ResidPredictTheory (residualized prediction theory score), ResidLexicalSurprisal (residualized Roark lexical surprisal), and ResidStructSurprisal (residualized Roark structural surprisal). We also ran separate models with the two components of prediction theory: ResidPLTAGSurprisal (residualized PLTAG surprisal), and PLTAGVerification (PLTAG verification cost; not residualization necessary). The relevant scores were obtained by computing only the surprisal scores or only the verification cost scores as defined by Prediction Theory, according to Equations (13) and (14).

The results in Table 4 indicate that only residualized Prediction Theory is a significant positive predictor of reading times. Both types of Roark lexical surprisal fail to reach significance.[10] Model comparison reveals that all models reported in Table 4, with the exception of the model including Roark lexicalized surprisal, significantly improve model fit over a baseline model that does not include them.[11]

In the analysis of the two components of prediction theory, PLTAG surprisal and PLTAG verification cost, we can see that the surprisal component is responsible for the effect of Prediction Theory. PLTAG verification cost is not a significant predictor on its own, which can be attributed to the fact that most verification cost values are zero, as no verification takes place at most words. Effectively, therefore, verification cost does not make predictions for the bulk of the words in the corpus, which explains why no significant effect is observed overall. A similar observation has previously been reported for integration cost (Gibson 1998; conceptually related to verification cost) on the Dundee data (Demberg and Keller 2008a). It is important to note, though, that adding verification cost to the baseline LME model increases model fit significantly, which provides some evidence for effectiveness of the verification cost component.

[10] The result for Roark structural surprisal differs from that reported by Demberg and Keller (2008a) and Demberg-Winterfors (2010). This can be attributed to the different outlier removal and more conservative treatment of random effects in the present article.

[11] Surprisal has subsequently been reported to be a significant predictor of Dundee reading time by Fossum and Levy (2012), who used a context-free grammar induced using the state-split model of Petrov and Klein (2007) in combination with a standard probabilistic Earley parser to compute surprisal estimates.

Table 4
Linear mixed effects models of first-pass time for predictors of theoretical interest: Prediction Theory cost, PLTAG surprisal, PLTAG verification cost, Roark lexical surprisal, and Roark structural surprisal, each residualized against low-level predictors (see text for details). Random intercepts of participant and random slopes under participants for the predictors of interest were also included. β = coefficient; SE = standard error; *t* = significance value. Note that the SE values for PLTAG surprisal and verification cost are almost identical to those of Prediction Theory and have been omitted.

	Prediction Theory			PLTAG Surprisal		PLTAG Verif. Cost	
Predictor	β	SE	t	β	t	β	t
Intercept	230.65	6.038	38.20***	230.62	37.95***	230.76	37.98***
WordLength	3.77	0.798	4.73***	3.91	32.94***	3.95	33.31***
WordFrequency	−9.54	0.214	−44.57***	−9.59	−44.64***	−9.59	−44.60***
PrevWordFreq	−3.81	0.151	−25.16***	−3.86	−25.39***	−3.87	−25.45***
PrevWordFixated	−19.32	0.532	−36.26***	−19.35	−36.22***	−19.34	−36.19***
LaunchDistance	−2.32	0.063	−36.63***	−2.38	−37.45***	−2.38	−37.38***
LandingPosition	−28.20	0.662	−42.59***	−28.23	−42.58***	−28.23	−42.58***
WordNoInSentence	−0.12	0.017	−6.90***	−0.12	−6.88***	−0.11	−6.60***
ResidBigramProb	−2.86	0.187	−15.32***	−2.86	−15.25***	−2.82	−15.02***
WordLen:WordFreq	−0.82	0.066	−12.39***	−0.74	−11.19***	−0.68	−10.56***
WordLen:LandPos	−13.03	0.255	−50.99***	−13.05	−51.04***	−13.05	−51.06***
ResidPredictTheory	0.34	0.155	2.22*				
ResidPLTAGSurprisal				0.37	2.27*		
PLTAGVerification						−2.87	−1.61

*$p < 0.05$, **$p < 0.01$, ***$p < 0.001$.

	Lexical Surprisal			Structural Surprisal		
Predictor	β	SE	t	β	SE	t
Intercept	230.77	6.078	37.96***	230.73	6.075	37.98***
WordLength	3.95	0.118	33.30***	3.95	0.118	33.32***
WordFrequency	−9.54	0.214	−44.46***	−9.57	0.214	−44.60***
PrevWordFreq	−3.87	0.152	−25.46***	−3.82	0.153	−24.88***
PrevWordFixated	−19.33	0.534	−36.18***	−19.35	0.534	−36.22***
LaunchDistance	−2.38	0.063	−37.42***	−2.38	0.063	−37.43***
LandingPosition	−28.24	0.663	−42.58***	−28.23	0.663	−42.58***
WordNoInSentence	−0.11	0.017	−6.69***	−0.11	0.017	−6.68***
ResidBigramProb	−2.83	0.187	−15.08***	−2.79	0.189	−14.75***
WordLen:WordFreq	−0.67	0.065	−10.34***	−0.69	0.064	−10.81***
WordLen:LandPos	−13.05	0.255	−51.06***	−13.05	0.255	−51.04***
ResidLexicalSurprisal[‡]	−0.12	0.075	−1.64			
ResidStructSurprisal				0.25	0.209	1.24

*$p < 0.05$, **$p < 0.01$, ***$p < 0.001$. [‡]Inclusion does not significantly improve model fit.

Furthermore, verification cost is able to explain experimental evidence that cannot be accounted for by surprisal alone, such as the relative clause asymmetry (Demberg and Keller 2009); see also Section 7.4.

To summarize, our implementation of the PLTAG incremental parsing algorithm allowed us to carry out a crucial test of Prediction Theory, that is, an evaluation using the naturally occurring reading data that the Dundee eye-tracking corpus provides. Due to the good coverage of the PLTAG parser, Prediction Theory estimates could be calculated for all but 1.5% of data points. Based on these estimates, we were able to show that Prediction Theory provides a better fit to the Dundee reading time data than a rival theory, namely, lexical and structural surprisal, as estimated using Roark's incremental parser.

As we have argued in Section 2, we cannot easily compare Prediction Theory to sentence processing models other than surprisal, as these are typically formalized but not implemented (e.g., Mazzei, Lombardo, and Sturt 2007) or only small-scale implementations based on restricted data sets are available (e.g., McRae, Spivey-Knowlton, and Tanenhaus 1998; Lewis and Vasishth 2005).

7.4 Other Experimental Results

In previous work, Prediction Theory difficulty scores estimated using PLTAG have been shown to successfully account for individual experimental results in psycholinguistics. Demberg and Keller (2009) showed that Prediction Theory can account for the relative clause asymmetry (Gibson 1998), that is, the fact that subject relative clauses are easier to process than object relative clauses. Prediction Theory provides an explanation for this fact in terms of higher verification cost for object relative clauses. Demberg and Keller (2009) also show that Prediction Theory successfully models the *either ... or* effect (Staub and Clifton 2006): Coordinate structures involving *either* are easier to process than ones involving just *or*. Prediction Theory explains this in terms of the lexicon entry for *either*, which introduces a prediction tree for the whole coordinate structure.

8. Conclusion

This article presented a probabilistic parser for PLTAG, a psycholinguistically motivated version of TAG. Our parser is incremental, builds fully connected structures (partial structures contain no unattached nodes), and models prediction, that is, the anticipation of upcoming syntactic material that is an important feature of human sentence processing. We proposed an efficient implementation of our parser based on fringes—data structures that indicate which nodes in a partial tree are available for the application of parsing rules. We trained our parser on a TAG-transformed version of the Penn Treebank and we showed that it achieves broad coverage (98.09%) and an F-score of 79.41 (with gold-standard POS tags) for TAG structure recovery.

We argued that broad coverage and competitive parsing accuracy are essential properties when it comes to testing a psycholinguistic model on realistic data, including the eye-tracking corpora which have recently become the gold-standard for psycholinguistic evaluation (e.g., Boston et al. 2008; Demberg and Keller 2008a; Frank 2009; Mitchell et al. 2010). We showed how our PLTAG parser underpins a theory of human sentence processing, Prediction Theory, and used Prediction Theory to derive estimates of human reading times. An evaluation on the Dundee eye-tracking corpus showed that Prediction Theory estimates achieve a better fit with human reading times than standard surprisal estimates computed using Roark's incremental parser.

Appendix A: A Parsing Schema for PLTAG

In this appendix, we supplement the presentation of the parsing algorithm in Section 4 with a parsing schema that spells out in detail what parsing rules the parser applies. We present this schema in two steps. First, we explain a simplified version of the parsing schema in Section A.1. This simplified scheme handles the different ways in which substitution and adjunction can be applied, but treats prediction trees like any other elementary tree. We then extend the parsing schema in Section A.2 to handle prediction trees correctly by adding markers to nodes when they are introduced, and removing the markers through applications of the Verification rule.

A.1 Parsing without Prediction Markers

The parsing schema manipulates parsing items that consist of a depth-first, left-to-right traversal of (prefix or elementary) trees, written as a sequence of downward or upward visits to the nodes of the tree, as explained in Section 4.1. The beginning of the current fringe—that is, the position just before the downward visit of the root or just before the upward visit of a leaf—is marked in the parsing items with a dot, •. For instance, we represent the prefix tree on the left of Figure 6 in the parsing item $\mathsf{S}^+\mathsf{a}^+ \bullet \mathsf{a}^-\mathsf{B}\downarrow^+ \mathsf{B}\downarrow^- \mathsf{C}\downarrow^+ \mathsf{C}\downarrow^- \mathsf{S}^-$. Because the parser proceeds incrementally, we know that all fringes of a prefix tree before the current one can no longer be modified by processing later words. This means that the **past fringes**, that is, the part of the node sequence before the dot, will remain fixed until parsing finishes.

We use this notation to write the rules of the PLTAG parser as parsing schemata (Shieber, Schabes, and Pereira 1995) that manipulate dotted strings of downward or upward node visits. We will first set aside the management of markers, and focus on substitution and adjunction. A parsing schema for these operations is shown in Figure A.1. We write f^+ (f^-) to indicate that f must be a possibly empty string of

$$\frac{f_0 \bullet f_1^- f_2^+ \mathsf{a}^+ \mathsf{a}^- f_3}{f_0 f_1^- f_2^+ \mathsf{a}^+ \bullet \mathsf{a}^- f_3}\ \textsf{Scan} \qquad\qquad \frac{f \text{ is node sequence of an elementary tree}}{\bullet f}\ \textsf{Init}$$

$$\frac{f_0 \bullet f_1^- f_2^+ \mathsf{A}\downarrow^+ \mathsf{A}\downarrow^- f_3 \qquad \mathsf{A}^+ g_1^+ g_2 \mathsf{A}^-}{f_0 \bullet f_1^- f_2^+ \mathsf{A}^+ g_1^+ g_2 \mathsf{A}^- f_3}\ \textsf{SubstDown}$$

(if right tree canonical: g_1 must be of form $g_3\mathsf{a}$)

$$\frac{\mathsf{A}^+ f_0 \bullet f_1^- \mathsf{A}^- \qquad g_1^+ \mathsf{A}\downarrow^+ \mathsf{A}\downarrow^- g_2}{g_1 \mathsf{A}^+ f_0 \bullet f_1^- \mathsf{A}^- g_2}\ \textsf{SubstUp}$$

(if right tree canonical: g_2 must be of form $g_3^- g_4^+ \mathsf{a}^+$)

$$\frac{f_0 \mathsf{A}^+ f_1 \bullet f_2^- \mathsf{A}^- f_3 \qquad \mathsf{A}^+ g_1^+ \mathsf{A}*^+ \mathsf{A}*^- g_2 \mathsf{A}^-}{f_0 \mathsf{A}^+ g_1^+ \mathsf{A}^+ f_1 \bullet f_2^- \mathsf{A}^+ g_2 \mathsf{A}^- f_3}\ \textsf{AdjDownR}$$

(if right tree canonical: g_2 must be of form $g_3^+ g_4^- g_5^+ \mathsf{a}^+$)

$$\frac{f_0 \bullet f_1^- f_2^+ \mathsf{A}^+ f_3 \mathsf{A}^- f_4 \qquad \mathsf{A}^+ g_1 \mathsf{A}*^+ \mathsf{A}*^- g_2^- \mathsf{A}^-}{f_0 \bullet f_1^- f_2^+ \mathsf{A}^+ g_1 \mathsf{A}^+ f_3 \mathsf{A}^- g_2^- \mathsf{A}^- f_4}\ \textsf{AdjDownL}$$

(if right tree canonical: g_1 must be of form $g_3^+ \mathsf{a}^+$)

$$\frac{\mathsf{A}^+ f_0 \bullet f_1^- f_2^+ \mathsf{A}*^+ \mathsf{A}*^- f_3 \mathsf{A}^- \qquad g_1 \mathsf{A}^+ g_2 \mathsf{A}^- g_3}{g_1 \mathsf{A}^+ f_0 \bullet f_1^- f_2^+ \mathsf{A}^+ g_2 \mathsf{A}^- f_3 \mathsf{A}^- g_3}\ \textsf{AdjUp}$$

(if right tree canonical: g_1 must be positive)

Figure A.1
Provisional rules for incremental parsing with PLTAG (without markers).

downward (upward) node visits; a stands for a visit to a node labeled with the terminal symbol a; and $\mathsf{A}\downarrow$ and $\mathsf{A}*$ represent visits to substitution and foot nodes. Notice that parse items are sequences of upward or downward node visits.

Any parse starts by an application of the Init rule, which initializes the prefix tree with a single elementary tree (represented by its node sequence f). Because we have not yet read any input, we place the dot before f. At any point at which the end of the current fringe (i.e., the next leaf of the prefix tree after the dot) is a terminal symbol a (usually right after a Subst or Adj rule has been applied), we can use the Scan rule to move the dot to just after the downward visit of a. The parse terminates successfully once we derive an item of the form $w \bullet f^-$ which contains all words in the input string; that is, an item in which all that remains to do for the depth-first search is to return to the root along the right frontier of the prefix tree.

The main work of the parsing algorithm is carried out by the five parsing rules for substitution and adjunction, which are illustrated in Figure A.2. Each of these rules has two premises; the left premise is the prefix tree and the right premise an elementary tree with which it is to be combined. Both substitution and adjunction rules are available in an Up and a Down version: Down is the case where the elementary tree is substituted or adjoined into the prefix tree, and Up the reverse. There are two versions of AdjDown depending on whether the foot node of the auxiliary tree is the leftmost or the rightmost leaf.

Consider the SubstDown operation, which substitutes the elementary tree into some substitution node u with label A of the prefix tree (again, see Figure A.2 for an illustration). Because we parse incrementally, this operation can only be applied if the substitution node is the next leaf of the prefix tree after the dot: Once we substitute into a node, we will never have another chance to substitute or adjoin into a node to its left. The fact that u is the next leaf is reflected in SubstDown by requiring that f_1 must consist only of upward traversals of edges of the prefix tree, and f_2 only of downward traversals. Furthermore, if the elementary tree is canonical, then we require that its first leaf is a lexical anchor by insisting that g_1 (the initial sequence of downward traversals to the first leaf) ends in a. Under these conditions, SubstDown manipulates the string of node visits as PLTAG's substitution operation requires.

The other parsing rules enforce and exploit the incremental derivation process in analogous ways. The AdjDown rules exist in two versions because auxiliary trees with a rightmost foot node must be read *before* the adjunction site is first visited, that is, before the dot is moved past its downward occurrence. On the other hand, auxiliary trees with a leftmost foot node must be read *after* the first visit to the adjunction site. As a general rule, after applying a Subst or Adj rule whose right-hand premise is a canonical tree, the string immediately after the dot will be of the form $f_1^- f_2^+ \mathsf{a}^+$. The Scan rule will move the dot over this substring, in preparation for the reading of the next word.

A.2 Complete Parsing Rules

The rules in Figure A.1 gloss over the fact that elementary trees in PLTAG may be prediction trees, whose upper and lower node halves may carry markers. The two halves of a node in a prefix tree may therefore carry markers as well, possibly from two different prediction trees. In general, node labels may be of the form A_l^k.

Figure A.3 shows an extended version of these rules that takes the markers into account. We extend the notation for node visits in the following way. A visit that carries a superscript, for example, A^k, must be to a node whose upper half carries the marker

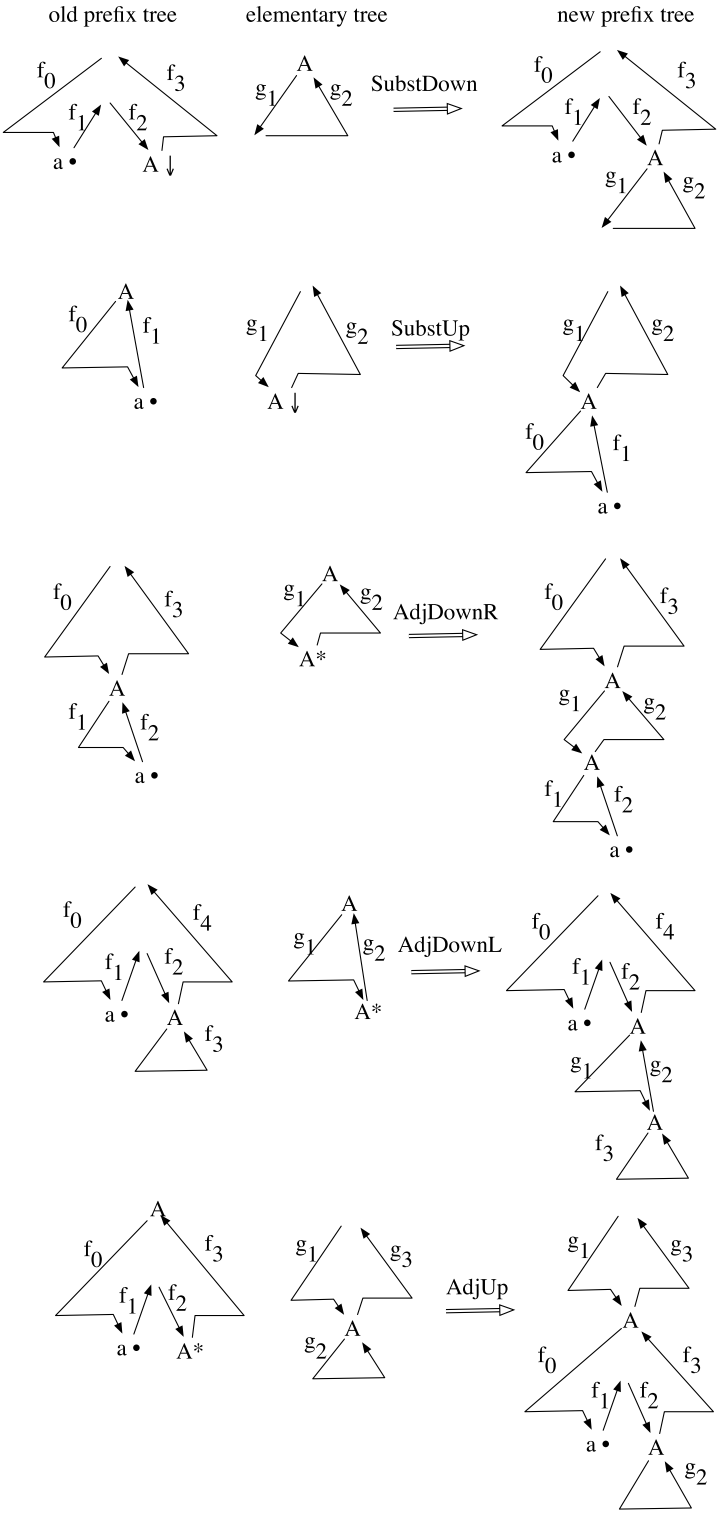


Figure A.2
Illustration of the parsing rules in Figure A.1.

$$\frac{f_0 \bullet f_1^- f_2^+ \mathsf{a}^+ \mathsf{a}^- f_3}{f_0 f_1^- f_2^+ \mathsf{a}^+ \bullet \mathsf{a}^- f_3}\ \mathsf{Scan} \qquad \frac{f \text{ is node sequence of an elementary tree}}{\bullet f}\ \mathsf{Init}$$

$$\frac{f_0 \bullet f_1^- f_2^+ \mathsf{A}\downarrow^{(k),+}\ \mathsf{A}\downarrow^{(k),-} f_3 \qquad \mathsf{A}^+_{(l)} g_1^+ g_2 \mathsf{A}^-_{(l)}}{f_0 \bullet f_1^- f_2^+ \mathsf{A}^{(k)+}_{(l)} g_1^+ g_2 \mathsf{A}^{(k)-}_{(l)} f_3}\ \mathsf{SubstDown}$$

(if right tree canonical: g_1 must be of form $g_3\mathsf{a}$)

$$\frac{\mathsf{A}^+_{(l)} f_0 \bullet f_1^- \mathsf{A}^-_{(l)} \qquad g_1^+ \mathsf{A}\downarrow^{(k),+}\ \mathsf{A}\downarrow^{(k),-} g_2}{g_1 \mathsf{A}^{(k)+}_{(l)} f_0 \bullet f_1^- \mathsf{A}^{(k)-}_{(l)} g_2}\ \mathsf{SubstUp}$$

(if right tree canonical: g_2 must be of form $g_3^- g_4^+ \mathsf{a}^+$)

$$\frac{f_0 \mathsf{A}^{(k)+}_{(l)} f_1 \bullet f_2^- \mathsf{A}^{(k)-}_{(l)} f_3 \qquad \mathsf{A}^+_{(m)} g_1^+ \mathsf{A}*^{(m),+}\ \mathsf{A}*^{(m),-} g_2 \mathsf{A}^-_{(m)}}{f_0 \mathsf{A}^{(k)+}_{(m)} g_1^+ \mathsf{A}^{(m)+}_{(l)} f_1 \bullet f_2^- \mathsf{A}^{(m)+}_{(l)} g_2 \mathsf{A}^{(k)-}_{(m)} f_3}\ \mathsf{AdjDownR}$$

(if right tree canonical: g_2 must be of form $g_3^+ g_4^- g_5^+ \mathsf{a}^+$)

$$\frac{f_0 \bullet f_1^- f_2^+ \mathsf{A}^{(k)+}_{(l)} f_3 \mathsf{A}^{(k)-}_{(l)} f_4 \qquad \mathsf{A}^+_{(m)} g_1 \mathsf{A}*^{(m),+}\ \mathsf{A}*^{(m),-} g_2^- \mathsf{A}^-_{(m)}}{f_0 \bullet f_1^- f_2^+ \mathsf{A}^{(k)+}_{(m)} g_1 \mathsf{A}^{(m)+}_{(l)} f_3 \mathsf{A}^{(m)-}_{(l)} g_2^- \mathsf{A}^{(k)-}_{(m)} f_4}\ \mathsf{AdjDownL}$$

(if right tree canonical: g_1 must be of form $g_3^+ \mathsf{a}^+$)

$$\frac{\mathsf{A}^+_{(k)} f_0 \bullet f_1^- f_2^+ \mathsf{A}*^{(k),+}\ \mathsf{A}*^{(k),-} f_3 \mathsf{A}^-_{(k)} \qquad g_1 \mathsf{A}^{(l)+}_{(l)} g_2 \mathsf{A}^{(l)-}_{(l)} g_3}{g_1 \mathsf{A}^{(l)+}_{(k)} f_0 \bullet f_1^- f_2^+ \mathsf{A}^{(k)+}_{(l)} g_2 \mathsf{A}^{(k)-}_{(l)} f_3 \mathsf{A}^{(l)-}_{(k)} g_3}\ \mathsf{AdjUp}$$

(if right tree canonical: g_1 must be positive)

$$\frac{f_0 \bullet f_1^- f_2^+ \mathsf{A}^{(k)+}_k \mathsf{A}^{(k)-}_k f_3 \qquad g_1 \mathsf{A}^+ g_2 \mathsf{A}^- g_3}{\mathit{verify}(f_0 \bullet f_1^- f_2^+ \mathsf{A}^{(k)+}_k, g_1 \mathsf{A}^+)\ g_2\ \mathit{extend}(\mathsf{A}^{(k)-}_k f_3, \mathsf{A}^- g_3)}\ \mathsf{Verify}(k)$$

Figure A.3
Incremental parsing rules for PLTAG.

k. To generalize over cases with and without markers, we may write the superscript in parentheses, for example, $\mathsf{A}^{(k)}$, to match a visit to a node that may or may not have a marker on the upper half. If any such node visit in the premise of the rule does have a marker, then all other uses of the superscript $^{(k)}$ in the premises and conclusion of the rule must also carry this marker. The use of subscripts A_l and $\mathsf{A}_{(l)}$ for lower markers is analogous.

The Subst and Adj rules pass markers onto the result item appropriately; note that the upper and lower markers of a node into which another tree is adjoined are split over two nodes in the result. The Scan rule does not allow a to carry markers. This is intentional: Trees with two lexical anchors (such as the one for *either ... or*) may contain predicted lexical nodes, which may not be processed by Scan as read before they have been verified.

Furthermore, the revised rule schema includes the rule Verify(k), which removes all occurrences of the marker k from the prefix tree by performing a verification operation. Verify(k) can be applied if the current fringe ends in a visit to some node u with lower marker k. It cuts the node visit sequence for the prefix tree τ into two parts: the string up to the downward visit to u, and the string starting at the upward visit. The first part consists of the region of the prefix tree to the left of the node, plus the downward visits to the spine of u, that is, the nodes on the path from the root to u. We call this part F_1 for brevity. The second part F_2 consists of the nodes to the right of u and the upward visits to the spine. $\mathsf{Verify}(k)$ chooses a verification tree τ_v that contains a visit to a node u' that has the same label as u, and splits it into the part $G_1 = g_1\mathsf{A}^+$ up to the downward visit of u', the part $G_2 = \mathsf{A}^- g_3$ starting at the upward visit of u', and the part g_2 between these two visits.

$\mathsf{Verify}(k)$ then attempts to compute an admissible correspondence and performs a verification step by calling the functions *verify* and *extend*. The function *verify* tries to establish a correspondence between the k-marked node visits in F_1 and the node visits in G_1. If such a correspondence exists, it is unique and can be found in linear time by matching the tree structure of the k-marked nodes in τ against the nodes in τ_v. If such a correspondence cannot be found, *verify* fails; otherwise, it returns F_1 with all k-markers removed.

The function *extend* performs the same task for the F_2 and G_2, that is, the "right" parts of τ and τ_v, except that G_2 may now contain complete subtrees that are not present in F_2. Just like *verify*, *extend* looks for the unique correspondence between the k-marked node visits on F_2 and the node visits in G_2 in linear time. If such a correspondence exists, *extend* adds the unmatched subtrees in G_2 to the correspondents of their parent nodes in F_2 and removes all k-markers from F_2. $\mathsf{Verify}(k)$ then obtains the node visit sequence for the result of the verification operation by concatenating $\mathit{verify}(F_1, G_1)$ with g_2 (i.e., the nodes below u') and $\mathit{extend}(F_2, G_2)$.

Acknowledgments

Portions of this work have benefited from presentations at the 2008 Architectures and Mechanisms for Sentence Processing Conference and at the 2008 CUNY Sentence Processing Conferences. We are grateful to and Aravind Joshi, Roger Levy, Mark Steedman, Patrick Sturt, and the four anonymous reviewers for feedback. This research was supported by EPSRC research grant EP/C546830/1: Prediction in Human Parsing.

References

Altmann, Gerry T. M. and Yuki Kamide. 1999. Incremental interpretation at verbs: Restricting the domain of subsequent reference. *Cognition*, 73:247–264.

Aoshima, Sachiko, Masaya Yoshida, and Colin Phillips. 2009. Incremental processing of coreference and binding in Japanese. *Syntax*, 12:93–134.

Arai, Manabu and Frank Keller. 2013. The use of verb-specific information for prediction in sentence processing. *Language and Cognitive Processes*, 28:525–560.

Bangalore, Srinivas and Aravind K. Joshi. 1999. Supertagging: an approach to almost parsing. *Computational Linguistics*, 25:237–265.

Barr, Dale J., Roger Levy, Christoph Scheepers, and Harry J. Tily. 2013. Random effects structure for confirmatory hypothesis testing: Keep it maximal. *Journal of Memory and Language*, 68(3):255–278.

Beuck, Niels, Arne Köhn, and Wolfgang Menzel. 2011. Incremental parsing and the evaluation of partial dependency analyses. In *Proceedings of the 1st International Conference on Dependency Linguistics*, pages 290–299, Barcelona.

Boston, Marisa Ferrara, John Hale, Reinhold Kliegl, Umesh Patil, and Shravan Vasishth. 2008. Parsing costs as predictors of reading difficulty: An evaluation using the Potsdam Sentence Corpus. *Journal of Eye Movement Research*, 2(1):1–12.

Brants, Thorsten. 2000. TnT—A statistical part-of-speech tagger. In *Proceedings of the 6th Conference on Applied Natural Language Processing*, pages 224–231, Seattle, WA.

Chen, John. 2001. *Towards Efficient Statistical Parsing using Lexicalized Grammatical Information*. Ph.D. thesis, University of Delaware.

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

Collins, Michael. 2003. Head-driven statistical models for natural language parsing. *Computational Linguistics*, 29(4):589–637.

Collins, Michael and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, pages 111–120, Barcelona.

Demberg, Vera. 2012. Incremental derivations in CCG. In *Proceedings of the 11th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+11)*, pages 198–206, Paris.

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

Demberg, Vera and Frank Keller. 2008b. A psycholinguistically motivated version of TAG. In *Proceedings of the 9th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+9)*, 8 pages, Tübingen.

Demberg, Vera and Frank Keller. 2009. A computational model of prediction in human parsing: Unifying locality and surprisal effects. In *Proceedings of the 29th Annual Meeting of the Cognitive Science Society*, pages 1,888–1,893, Amsterdam.

Demberg-Winterfors, Vera. 2010. *A Broad-Coverage Model of Prediction in Human Sentence Processing*. Ph.D. thesis, University of Edinburgh.

Fossum, Victoria and Roger Levy. 2012. Syntactic vs. hierarchical models of human incremental sentence processing. In *Proceedings of the 3rd Workshop on Cognitive Modeling and Computational Linguistics*, pages 61–69, Montreal.

Frank, Stefan L. 2009. Surprisal-based comparison between a symbolic and a connectionist model of sentence processing. In Niels Taatgen and Hedderik van Rijn, editors, *Proceedings of the 31st Annual Conference of the Cognitive Science Society*, pages 1,139–1,144, Amsterdam.

Gibson, Edward. 1998. Linguistic complexity: Locality of syntactic dependencies. *Cognition 68*, pages 1–76.

Gibson, Edward. 2000. Dependency locality theory: A distance-based theory of linguistic complexity. In Alec Marantz, Yasushi Miyashita, and Wayne O'Neil, editors, *Image, Language, Brain: Papers from the First Mind Articulation Project Symposium*. MIT Press, Cambridge, MA, pages 95–126.

Hale, John. 2001. A probabilistic Earley parser as a psycholinguistic model. In *Proceedings of the 2nd Conference of the North American Chapter of the Association for Computational Linguistics*, volume 2, pages 159–166, Pittsburgh, PA.

Hockenmaier, Julia and Mark Steedman. 2007. CCGbank: A corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396.

Joshi, Aravind K., Leon Levy, and M. Takahashi. 1975. Tree adjunct grammars. *Journal of the Computer and System Sciences*, 10(1):136–163.

Joshi, Aravind K. and Yves Schabes. 1992. Tree adjoining grammars and lexicalized grammars. In Maurice Nivat and Andreas Podelski, editors, *Tree Automata and Languages*. North-Holland, Amsterdam, pages 409–432.

Kamide, Yuki, Christoph Scheepers, and Gerry T. M. Altmann. 2003. Integration of syntactic and semantic information in predictive processing: Cross-linguistic evidence from German and English. *Journal of Psycholinguistic Research*, 32:37–55.

Kashammer, Miriam and Vera Demberg. 2012. German and English treebanks and lexica for tree-adjoining grammars. In *Proceedings of the 8th International Conference on Language Resources and Evaluation*, pages 1,880–1,887, Istanbul.

Kato, Yoshihide, Shigeki Matsubara, and Yasuyoshi Inagaki. 2004. Stochastically evaluating the validity of partial parse trees in incremental parsing. In *Proceedings of the ACL Workshop Incremental Parsing*, pages 9–15, Barcelona.

Keller, Frank. 2010. Cognitively plausible models of human language processing. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics: Short Papers*, pages 60–67, Uppsala.

Kennedy, Alan and Joel Pynte. 2005. Parafoveal-on-foveal effects in normal reading. *Vision Research*, 45:153–168.

Konieczny, Lars. 2000. Locality and parsing complexity. *Journal of Psycholinguistic Research*, 29(6):627–645.

Lewis, Richard L. and Shravan Vasishth. 2005. An activation-based model of sentence processing as skilled memory retrieval. *Cognitive Science*, 29:1–45.

Magerman, David M. 1994. *Natural language parsing as statistical pattern recognition*. Ph.D. thesis, Stanford University.

Mazzei, Alessandro. 2005. *Formal and empirical issues of applying dynamics to Tree Adjoining Grammars*. Ph.D. thesis, Università di Torino.

Mazzei, Alessandro, Vincenzo Lombardo, and Patrick Sturt. 2007. Dynamic TAG and lexical dependencies. *Research on Language and Computation*, 5:309–332.

McRae, Ken, Michael J. Spivey-Knowlton, and Michael K. Tanenhaus. 1998. Modeling the influence of thematic fit (and other constraints) in on-line sentence comprehension. *Journal of Memory and Language*, 38(3):283–312.

Mitchell, Jeff, Mirella Lapata, Vera Demberg, and Frank Keller. 2010. Syntactic and semantic factors in processing difficulty: An integrated measure. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 196–206, Uppsala.

Nivre, Joakim. 2004. Incrementality in deterministic dependency parsing. In *Proceedings of the ACL Workshop on Incremental Parsing*, pages 50–57, Barcelona.

Palmer, Martha, Dan Gildea, and Paul Kingsbury. 2003. The Proposition Bank: an annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

Petrov, Slav and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 404–411, Rochester, NY.

Pinheiro, José C. and Douglas M. Bates. 2000. *Mixed-Effects Models in S and S-PLUS*. Springer, New York.

Resnik, Philip. 1992a. Left-corner parsing and psychological plausibility. In *Proceedings of the 15th International Conference on Computational Linguistics*, pages 191–197, Nantes.

Resnik, Philip. 1992b. Probabilistic tree-adjoining grammar as a framework for statistical natural language processing. In *Proceedings of the 15th International Conference on Computational Linguistics*, pages 418–424, Nantes.

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

Roark, Brian, 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 Conference on Empirical Methods in Natural Language Processing*, pages 324–333, Singapore.

Roland, Douglas, Gail Mauner, Carolyn O'Meara, and Hongoak Yun. 2012. Discourse expectations and relative clause processing. *Journal of Memory and Language*, 66(3):479–508.

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

Shen, Libin and Aravind K. Joshi. 2005. Incremental LTAG parsing. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 811–818, Vancouver.

Shieber, Stuart M., Yves Schabes, and Fernando C. N. Pereira. 1995. Principles and implementation of deductive parsing. *Journal of Logic Programming*, 24(1–2):3–36.

Staub, Adrian. 2010. Eye movements and processing difficulty in object relative clauses. *Cognition*, 116:71–86.

Staub, Adrian and Charles Clifton. 2006. Syntactic prediction in language comprehension: Evidence from either . . . or. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 32:425–436.

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

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

Tanenhaus, Michael K., Michael J. Spivey-Knowlton, Kathleen M. Eberhard, and Julie C. Sedivy. 1995. Integration of visual and linguistic information in spoken language comprehension. *Science*, 268:1,632–1,634.

Thompson, Henry S., Mike Dixon, and John Lamping. 1991. Compose-reduce parsing. In *Proceedings of the 29th Annual Meeting on Association for Computational Linguistics*, pages 87–97, Berkeley, CA.

Vadas, David and James Curran. 2007. Adding noun phrase structure to the Penn Treebank. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 240–247, Prague.

Vijay-Shanker, K. and Aravind K. Joshi. 1988. Feature structures based tree adjoining grammars. In *Proceedings of the 12th International Conference on Computational Linguistics*, pages 714–719, Morristown, NJ.

Witten, Ian H. and Timothy C. Bell. 1991. The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression. *IEEE Transaction on Information Theory*, 37(4):1,085–1,094.

Wu, Stephen, 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 1,189–1,198, Uppsala.

Xia, Fei, Martha Palmer, and Aravind Joshi. 2000. A uniform method of grammar extraction and its applications. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 53–62, Hong Kong.

Yoshida, Masaya, Michael Walsh-Dickey, and Patrick Sturt. 2013. Predictive processing of syntactic structure: Sluicing and ellipsis in real-time sentence processing. *Language and Cognitive Processes*, 28:272–302.