Parsing with Treebank Grammars: Empirical Bounds, Theoretical Models, and the Structure of the Penn Treebank

Dan Klein and **Christopher D. Manning**
Computer Science Department
Stanford University
Stanford, CA 94305-9040
{klein, manning}@cs.stanford.edu

Abstract

This paper presents empirical studies and closely corresponding theoretical models of the performance of a chart parser exhaustively parsing the Penn Treebank with the Treebank's own CFG grammar. We show how performance is dramatically affected by rule representation and tree transformations, but little by top-down vs. bottom-up strategies. We discuss grammatical saturation, including analysis of the strongly connected components of the phrasal nonterminals in the Treebank, and model how, as sentence length increases, the effective grammar rule size increases as regions of the grammar are unlocked, yielding super-cubic observed time behavior in some configurations.

1 Introduction

This paper originated from examining the empirical performance of an exhaustive active chart parser using an untransformed treebank grammar over the Penn Treebank. Our initial experiments yielded the surprising result that for many configurations empirical parsing speed was super-cubic in the sentence length. This led us to look more closely at the structure of the treebank grammar. The resulting analysis builds on the presentation of Charniak (1996), but extends it by elucidating the structure of non-terminal interrelationships in the Penn Treebank grammar. On the basis of these studies, we build simple theoretical models which closely predict observed parser performance, and, in particular, explain the originally observed super-cubic behavior.

We used treebank grammars induced directly from the local trees of the entire WSJ section of the Penn Treebank (Marcus et al., 1993) (release 3). For each length and parameter setting, 25 sentences evenly distributed through the treebank were parsed. Since we were parsing sentences from among those from which our grammar was derived, coverage was never an issue. Every sentence parsed had at least one parse – the parse with which it was originally observed.[1]

The sentences were parsed using an implementation of the probabilistic chart-parsing algorithm presented in (Klein and Manning, 2001). In that paper, we present a theoretical analysis showing an $O(n^3)$ worst-case time bound for exhaustively parsing arbitrary context-free grammars. In what follows, we do not make use of the probabilistic aspects of the grammar or parser.

2 Parameters

The parameters we varied were:

- Tree Transforms: **NoTransform**, NoEmpties, NoUnariesHigh, and NoUnariesLow.
- Grammar Rule Encodings: List, **Trie**, or Min
- Rule Introduction: TopDown or **BottomUp**

The default settings are shown above in bold face.

We do not discuss all possible combinations of these settings. Rather, we take the bottom-up parser using an untransformed grammar with trie rule encodings to be the basic form of the parser. Except where noted, we will discuss how each factor affects this baseline, as most of the effects are orthogonal. When we name a setting, any omitted parameters are assumed to be the defaults.

2.1 Tree Transforms

In all cases, the grammar was directly induced from (transformed) Penn treebank trees. The transforms used are shown in figure 1. For all settings, functional tags and crossreferencing annotations were stripped. For NoTransform, no other modification was made. In particular, empty nodes (represented as -NONE- in the treebank) were turned into rules that generated the empty string (ϵ), and there was no collapsing of categories (such as PRT and ADVP) as is often done in parsing work (Collins, 1997, etc.). For

[1] Effectively "testing on the training set" would be invalid if we wished to present performance results such as precision and recall, but it is not a problem for the present experiments, which focus solely on the parser load and grammar structure.

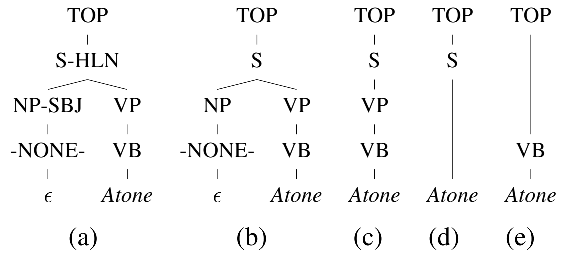

Figure 1: Tree Transforms: (a) The raw tree, (b) NoTransform, (c) NoEmpties, (d) NoUnariesHigh (e) NoUnariesLow

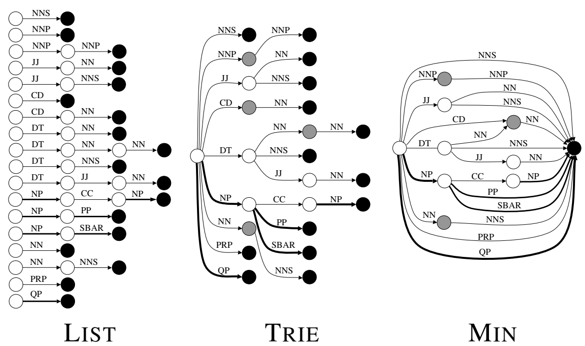

Figure 2: Grammar Encodings: FSAs for a subset of the rules for the category NP. Non-black states are active, non-white states are accepting, and bold transitions are phrasal.

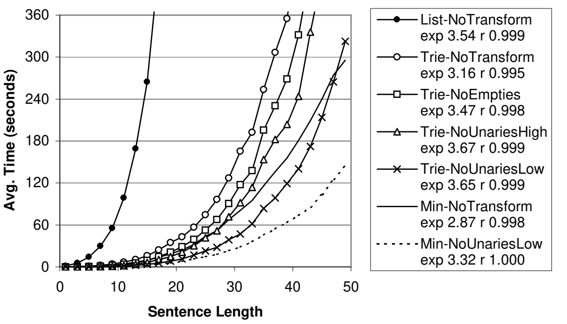

Figure 3: The average time to parse sentences using various parameters.

NoEmpties, empties were removed by pruning nonterminals which covered no overt words. For NoUnariesHigh, and NoUnariesLow, unary nodes were removed as well, by keeping only the tops and the bottoms of unary chains, respectively.[2]

2.2 Grammar Rule Encodings

The parser operates on Finite State Automata (FSA) grammar representations. We compiled grammar rules into FSAs in three ways: Lists, Tries, and MINimized FSAs. An example of each representation is given in figure 2. For List encodings, each local tree type was encoded in its own, linearly structured FSA, corresponding to Earley (1970)-style dotted rules. For Trie, there was one FSA per category, encoding together all rule types producing that category. For Min, state-minimized FSAs were constructed from the trie FSAs. Note that while the rule encoding may dramatically affect the efficiency of a parser, it does not change the actual set of parses for a given sentence in any way.[3]

3 Observed Performance

In this section, we outline the observed performance of the parser for various settings. We frequently speak in terms of the following:

- **span:** a range of words in the chart, e.g., [1,3][4]
- **edge:** a category over a span, e.g., NP:[1,3]
- **traversal:** a way of making an edge from an active and a passive edge, e.g., NP:[1,3] $\Leftarrow$ (NP$\rightarrow$DT.NN:[1,2] + NN:[2,3])

3.1 Time

The parser has an $O(SCn^3)$ theoretical time bound, where n is the number of words in the sentence to be parsed, C is the number of nonterminal categories in the grammar and S is the number of (active) states in the FSA encoding of the grammar. The time bound is derived from counting the number of traversals processed by the parser, each taking $O(1)$ time.

In figure 3, we see the average time[5] taken per sentence length for several settings, with the empirical exponent (and correlation r-value) from the best-fit simple power law model to the right. Notice that most settings show time growth greater than $O(n^3)$.

Although, $O(n^3)$ is simply an asymptotic bound, there are good explanations for the observed behavior. There are two primary causes for the super-cubic time values. The first is theoretically uninteresting. The parser is implemented in Java, which uses garbage collection for memory management. Even when there is plenty of memory for a parse's primary data structures, "garbage collection thrashing" can occur when

[2] In no case were the nonterminal-to-word or TOP-to-nonterminal unaries altered.

[3] FSAs are not the only method of representing and compacting grammars. For example, the prefix compacted tries we use are the same as the common practice of ignoring items before the dot in a dotted rule (Moore, 2000). Another logical possibility would be trie encodings which compact the grammar states by common suffix rather than common prefix, as in (Leermakers, 1992). The savings are less than for prefix compaction.

[4] Note that the number of words (or *size*) of a span is equal to the difference between the endpoints.

[5] The hardware was a 700 MHz Intel Pentium III, and we used up to 2GB of RAM for very long sentences or very poor parameters. With good parameter settings, the system can parse 100+ word treebank sentences.

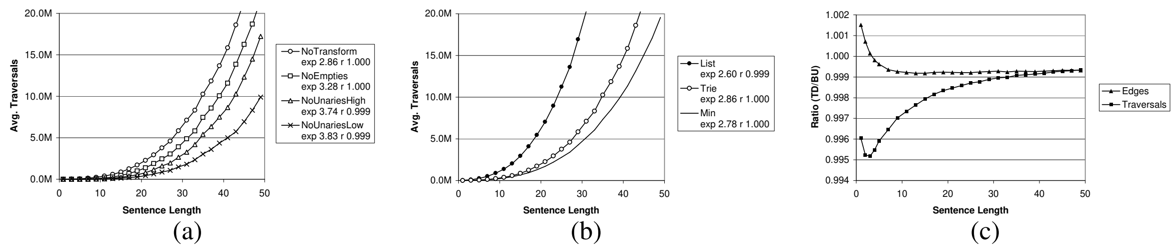

Figure 4: (a) The number of traversals for different grammar transforms. (b) The number of traversals for different grammar encodings. (c) The ratio of the number of edges and traversals produced with a top-down strategy over the number produced with a bottom-up strategy (shown for TRIE-NOTRANSFORM, others are similar).

parsing longer sentences as temporary objects cause increasingly frequent reclamation. To see past this effect, which inflates the empirical exponents, we turn to the actual traversal counts, which better illuminate the issues at hand. Figures 4 (a) and (b) show the traversal curves corresponding to the times in figure 3.

The interesting cause of the varying exponents comes from the "constant" terms in the theoretical bound. The second half of this paper shows how modeling growth in these terms can accurately predict parsing performance (see figures 9 to 13).

3.2 Memory

The memory bound for the parser is $O(Sn^2)$. Since the parser is running in a garbage-collected environment, it is hard to distinguish required memory from utilized memory. However, unlike time and traversals which in practice can diverge, memory requirements match the number of edges in the chart almost exactly, since the large data structures are all proportional in size to the number of edges $E = O(Sn^2)$.[6]

Almost all edges stored are active edges ($\geq 98\%$ for sentences longer than 30 words), of which there can be $O(Sn^2)$: one for every grammar state and span. Passive edges, of which there can be $O(Cn^2)$, one for every category and span, are a shrinking minority. This is because, while C is bounded above by 27 in the treebank[7] (for spans ≥ 2), S numbers in the thousands (see figure 12). Thus, required memory will be implicitly modeled when we model active edges in section 4.3.

3.3 Tree Transforms

Figure 4 (a) shows the effect of the tree transforms on traversal counts. The NOUNARIES settings are much more efficient than the others, however this efficiency comes at a price in terms of the utility of the final parse. For example, regardless of which NOUNARIES transform is chosen, there will be NP nodes missing from the parses, making the parses less useful for any task requiring NP identification. For the remainder of the paper, we will focus on the settings NOTRANSFORM and NOEMPTIES.

3.4 Grammar Encodings

Figure 4 (b) shows the effect of each tree transform on traversal counts. The more compacted the grammar representation, the more time-efficient the parser is.

3.5 Top-Down vs. Bottom-Up

Figure 4 (c) shows the effect on total edges and traversals of using top-down and bottom-up strategies. There are some extremely minimal savings in traversals due to top-down filtering effects, but there is a corresponding penalty in edges as rules whose left-corner cannot be built are introduced. Given the highly unrestrictive nature of the treebank grammar, it is not very surprising that top-down filtering provides such little benefit. However, this is a useful observation about real world parsing performance. The advantages of top-down chart parsing in providing grammar-driven prediction are often advanced (e.g., Allen 1995:66), but in practice we find almost no value in this for broad coverage CFGs. While some part of this is perhaps due to errors in the treebank, a large part just reflects the true nature of broad coverage grammars: e.g., once you allow adverbial phrases almost anywhere and allow PPs, (participial) VPs, and (temporal) NPs to be adverbial phrases, along with phrases headed by adverbs, then there is very little useful top-down control left. With such a permissive grammar, the only real constraints are in the POS tags which anchor the local trees (see section 4.3). Therefore, for the remainder of the paper, we consider only bottom-up settings.

4 Models

In the remainder of the paper we provide simple models that nevertheless accurately capture the varying magnitudes and exponents seen for different grammar encodings and tree transformations. Since the n^3 term of $O(SCn^3)$ comes directly from the number of start,

[6] A standard chart parser might conceivably require storing more than $O(E)$ traversals on its agenda, but ours provably never does.

[7] This count is the number of phrasal categories with the introduction of a TOP label for the unlabeled top treebank nodes.

split, and end points for traversals, it is certainly not responsible for the varying growth rates. An initially plausible possibility is that the quantity bounded by the C term is non-constant in n in practice, because longer spans are more ambiguous in terms of the number of categories they can form. This turns out to be generally false, as discussed in section 4.2. Alternately, the effective S term could be growing with n, which turns out to be true, as discussed in section 4.3.

The number of (possibly zero-size) spans for a sentence of length n is fixed: $(n+1)(n+2)/2$. Thus, to be able to evaluate and model the total edge counts, we look to the number of edges over a given span.

Definition 1 *The* passive *(or* active*)* saturation *of a given span is the number of passive (or active) edges over that span.*

In the total time and traversal bound $O(SCn^3)$, the effective value of S is determined by the active saturation, while the effective value of C is determined by the passive saturation. An interesting fact is that the saturation of a span is, for the treebank grammar and sentences, essentially independent of what size sentence the span is from and where in the sentence the span begins. Thus, for a given span size, we report the average over all spans of that size occurring anywhere in any sentence parsed.

4.1 Treebank Grammar Structure

The reason that effective growth is not found in the C component is that passive saturation stays almost constant as span size increases. However, the more interesting result is not that saturation is relatively constant (for spans beyond a small, grammar-dependent size), but that the saturation values are extremely *large* compared to C (see section 4.2). For the NoTransform and NoEmpties grammars, most categories are reachable from most other categories using rules which can be applied over a single span. Once you get one of these categories over a span, you will get the rest as well. We now formalize this.

Definition 2 *A category X is* empty-reachable *in a grammar G if X can be built using only empty terminals.*

The empty-reachable set for the NoTransform grammar is shown in figure 5.[8] These 23 categories plus the tag -NONE- create a passive saturation of 24 for zero-spans for NoTransform (see figure 9).

Definition 3 *A category Y is* same-span-reachable *from a category X in a grammar G if Y can be built from X using a parse tree in which, aside from at most one instance of X, every node not dominating that instance is an instance of an empty-reachable category.*

[8]The set of phrasal categories used in the Penn Treebank is documented in Manning and Schütze (1999, 413); Marcus et al. (1993, 281) has an early version.

ADJP	ADVP	FRAG	INTJ	NAC	NP
NX	PP	PRN	QP	RRC	S
SBAR	SBARQ	SINV	SQ	TOP	UCP
VP	WHADVP	WHNP	WHPP	X	

Figure 5: The empty-reachable set for the NoTransform grammar.

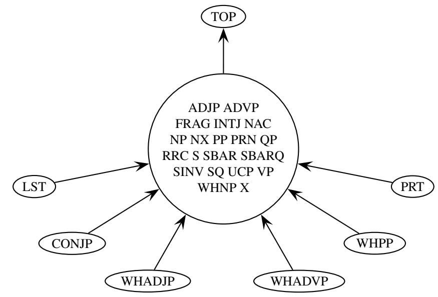

Figure 6: The same-span reachability graph for the NoTransform grammar.

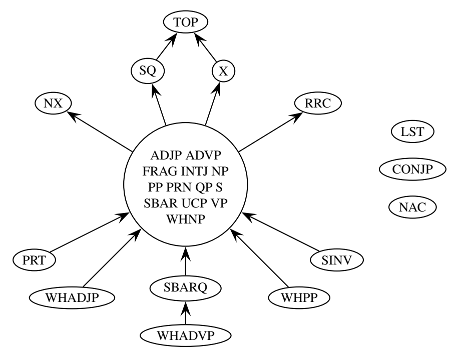

Figure 7: The same-span-reachability graph for the NoEmpties grammar.

The same-span-reachability relation induces a graph over the 27 non-terminal categories. The strongly-connected component (SCC) reduction of that graph is shown in figures 6 and 7.[9] Unsurprisingly, the largest SCC, which contains most "common" categories (S, NP, VP, PP, etc.) is slightly larger for the NoTransform grammar, since the empty-reachable set is non-empty. However, note that even for NoTransform, the largest SCC is smaller than the empty-reachable set, since empties provide direct entry into some of the lower SCCs, in particular because of WH-gaps.

Interestingly, this same high-reachability effect occurs even for the NoUnaries grammars, as shown in the next section.

4.2 Passive Edges

The total growth and saturation of passive edges is relatively easy to describe. Figure 8 shows the total num-

[9]Implied arcs have been removed for clarity. The relation is in fact the transitive closure of this graph.

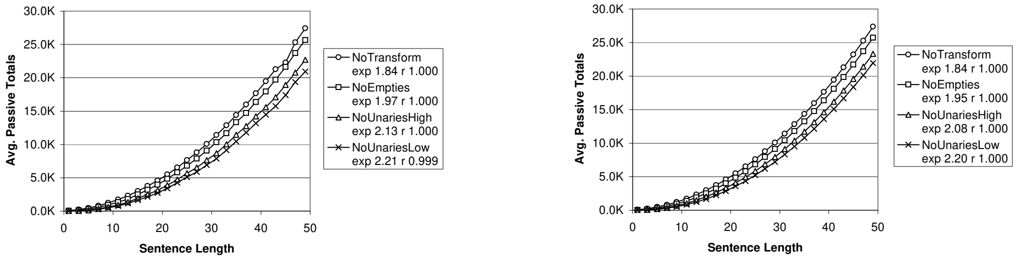

Figure 8: The average number of passive edges processed in practice (left), and predicted by our models (right).

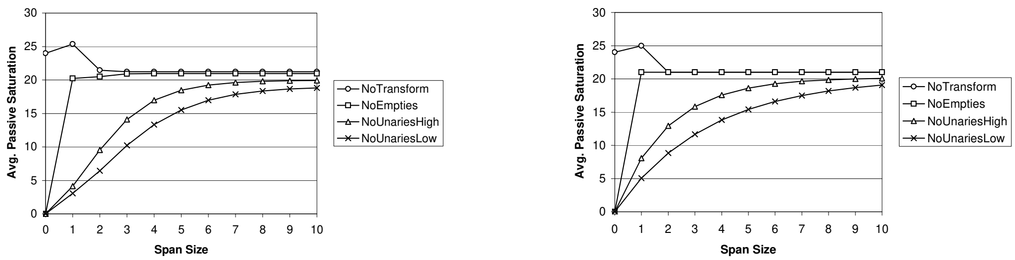

Figure 9: The average passive saturation (number of passive edges) for a span of a given size as processed in practice (left), and as predicted by our models (right).

ber of passive edges by sentence length, and figure 9 shows the saturation as a function of span size.[10] The grammar representation does not affect which passive edges will occur for a given span.

The large SCCs cause the relative independence of passive saturation from span size for the NoTransform and NoEmpties settings. Once any category in the SCC is found, all will be found, as well as all categories reachable from that SCC. For these settings, the passive saturation can be summarized by three saturation numbers: zero-spans (empties) $psat_0$, one-spans (words) $psat_1$, and all larger spans (categories) $psat_2$. Taking averages directly from the data, we have our first model, shown on the right in figure 9.

For the NoUnaries settings, there will be no same-span reachability and hence no SCCs. To reach a new category always requires the use of at least one overt word. However, for spans of size 6 or so, enough words exist that the same high saturation effect will still be observed. This can be modeled quite simply by assuming each terminal unlocks a fixed fraction of the nonterminals, as seen in the right graph of figure 9, but we omit the details here.

Using these passive saturation models, we can directly estimate the total passive edge counts by summation:

$$ptot(n) = \sum_{i=0}^{n}(n+1-i)psat_i$$

The predictions are shown in figure 8. For the NoTransform or NoEmpties settings, this reduces to:

$$ptot(n) = \tfrac{(n-1)n}{2}psat_2 + (n)psat_1 + (n+1)psat_0$$

We correctly predict that the passive edge total exponents will be slightly less than 2.0 when unaries are present, and greater than 2.0 when they are not. With unaries, the linear terms in the reduced equation are significant over these sentence lengths and drag down the exponent. The linear terms are larger for NoTransform and therefore drag the exponent down more.[11] Without unaries, the more gradual saturation growth increases the total exponent, more so for NoUnariesLow than NoUnariesHigh. However, note that for spans around 8 and onward, the saturation curves are essentially constant for all settings.

4.3 Active Edges

Active edges are the vast majority of edges and essentially determine (non-transient) memory requirements. While passive counts depend only on the grammar transform, active counts depend primarily on the encoding for general magnitude but also on the transform for the details (and exponent effects). Figure 10 shows the total active edges by sentence size for three settings chosen to illustrate the main effects. Total active growth is sub-quadratic for List, but has an exponent of up to about 2.4 for the Trie settings.

[10] The maximum possible passive saturation for any span greater than one is equal to the number of phrasal categories in the treebank grammar: 27. However, empty and size-one spans can additionally be covered by POS tag edges.

[11] Note that, over these values of n, even a basic quadratic function like the simple sum $\sum n = (n+1)n/2$ has a best-fit simple power curve exponent of only ≈ 1.95 for the same reason. Moreover, note that $n^2/2$ has a higher best-fit exponent, yet will never actually outgrow it.

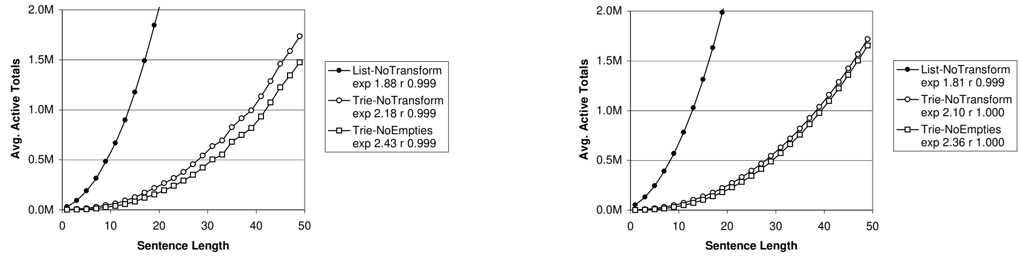

Figure 10: The average number of active edges for sentences of a given length as observed in practice (left), and as predicted by our models (right).

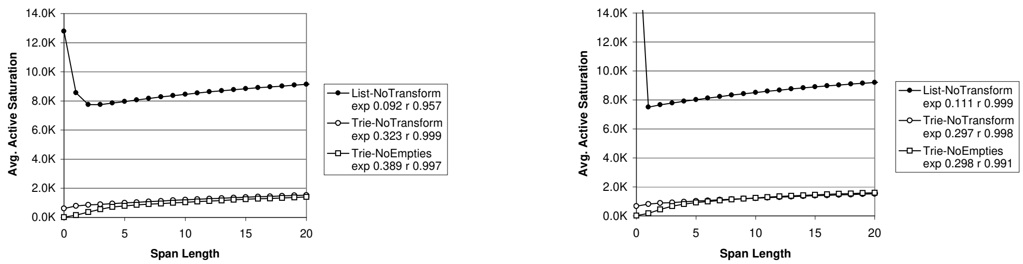

Figure 11: The average active saturation (number of active edges) for a span of a given size as processed in practice (left), and as predicted by our models (right).

	NoTrans	NoEmpties	NoUHigh	NoULow
List	80120	78233	81287	100818
Trie	17298	17011	17778	22026
Min	2631	2610	2817	3250

Figure 12: Grammar sizes: active state counts.

To model the active totals, we again begin by modeling the active saturation curves, shown in figure 11. The active saturation for any span is bounded above by S, the number of active grammar states (states in the grammar FSAs which correspond to active edges). For list grammars, this number is the sum of the lengths of all rules in the grammar. For trie grammars, it is the number of unique rule prefixes (including the LHS) in the grammar. For minimized grammars, it is the number of states with outgoing transitions (non-black states in figure 2). The value of S is shown for each setting in figure 12. Note that the maximum number of active states is dramatically larger for lists since common rule prefixes are duplicated many times. For minimized FSAs, the state reduction is even greater. Since states which are earlier in a rule are much more likely to match a span, the fact that tries (and min FSAs) compress early states is particularly advantageous.

Unlike passive saturation, which was relatively close to its bound C, active saturation is much farther below S. Furthermore, while passive saturation was relatively constant in span size, at least after a point, active saturation quite clearly grows with span size, even for spans well beyond those shown in figure 11. We now model these active saturation curves.

What does it take for a given active state to match a given span? For Trie and List, an active state corresponds to a prefix of a rule and is a mix of POS tags and phrasal categories, each of which must be matched, in order, over that span for that state to be reached. Given the large SCCs seen in section 4.1, phrasal categories, to a first approximation, might as well be wildcards, able to match any span, especially if empties are present. However, the tags are, in comparison, very restricted. Tags must actually match a word in the span.

More precisely, consider an active state a in the grammar and a span s. In the Trie and List encodings, there is some, possibly empty, list L of labels that must be matched over s before an active edge with this state can be constructed over that span.[12] Assume that the phrasal categories in L can match any span (or any non-zero span in NoEmpties).[13] Therefore, phrasal categories in L do not constrain whether a can match s. The real issue is whether the tags in L will match words in s. Assume that a random tag matches a random word with a fixed probability p, independently of where the tag is in the rule and where the word is in the sentence.[14] Assume further that, although tags occur more often than categories in rules (63.9% of rule items are tags in the NoTransform case[15]), given a

[12] The essence of the Min model, which is omitted here, is that states are represented by the "easiest" label sequence which leads to that state.

[13] The model for the NoUnaries cases is slightly more complex, but similar.

[14] This is of course false; in particular, tags at the end of rules disproportionately tend to be punctuation tags.

[15] Although the present model does not directly apply to the NoUnaries cases, NoUnariesLow is significantly

fixed number of tags and categories, all permutations are equally likely to appear as rules.[16]

Under these assumptions, the probability that an active state a is in the treebank grammar will depend only on the number t of tags and c of categories in L. Call this pair $\sigma(a) = (t, c)$ the *signature* of a. For a given signature σ, let $count(\sigma)$ be the number of active states in the grammar which have that signature.

Now, take a state a of signature (t, c) and a span s. If we align the tags in a with words in s and align the categories in a with spans of words in s, then provided the categories align with a non-empty span (for NOEMPTIES) or any span at all (for NOTRANSFORM), then the question of whether this alignment of a with s matches is determined entirely by the t tags. However, with our assumptions, the probability that a randomly chosen set of t tags matches a randomly chosen set of t words is simply p^t.

We then have an expression for the chance of matching a specific alignment of an active state to a specific span. Clearly, there can be many alignments which differ only in the spans of the categories, but line up the same tags with the same words. However, there will be a certain number of unique ways in which the words and tags can be lined up between a and s. If we know this number, we can calculate the total probability that there is some alignment which matches. For example, consider the state NP $\rightarrow$ NP CC NP . PP (which has signature (1,2) – the PP has no effect) over a span of length n, with empties available. The NPs can match any span, so there are n alignments which are distinct from the standpoint of the CC tag – it can be in any position. The chance that some alignment will match is therefore $1-(1-p)^n$, which, for small p is roughly linear in n. It should be clear that for an active state like this, the longer the span, the more likely it is that this state will be found over that span.

It is unfortunately not the case that all states with the same signature will match a span length with the same probability. For example, the state NP $\rightarrow$ NP NP CC . NP has the same signature, but *must* align the CC with the final element of the span. A state like this will not become more likely (in our model) as span size increases. However, with some straightforward but space-consuming recurrences, we can calculate the expected chance that a random rule of a given signature will match a given span length. Since we know how many states have a given signature, we can calculate the total active saturation $asat(n)$ as

$$asat(n) = \sum_{\sigma} count(\sigma) E_{a \in \sigma}[P(match(a, n))]$$

This model has two parameters. First, there is p which we estimated directly by looking at the expected match between the distribution of tags in rules and the distribution of tags in the Treebank text (which is around 1/17.7). No factor for POS tag ambiguity was used, another simplification.[17] Second, there is the map $count$ from signatures to a number of active states, which was read directly from the compiled grammars.

This model predicts the active saturation curves shown to the right in figure 11. Note that the model, though not perfect, exhibits the qualitative differences between the settings, both in magnitudes and exponents.[18] In particular:

- The transform primarily changes the saturation over short spans, while the encoding determines the overall magnitudes. For example, in TRIE-NOEMPTIES the low-span saturation is lower than in TRIE-NOTRANSFORM since short spans in the former case can match only signatures which have both t and c small, while in the latter only t needs to be small. Therefore, the several hundred states which are reachable only via categories all match every span starting from size 0 for NOTRANSFORM, but are accessed only gradually for NOEMPTIES. However, for larger spans, the behavior converges to counts characteristic for TRIE encodings.
- For LIST encodings, the early saturations are huge, due to the fact that most of the states which are available early for trie grammars are precisely the ones duplicated up to thousands of times in the list grammars. However, the additive gain over the initial states is roughly the same for both, as after a few items are specified, the tries become sparse.
- The actual magnitudes and exponents[19] of the saturations are surprisingly well predicted, suggesting that this model captures the essential behavior.

These active saturation curves produce the active total curves in figure 10, which are also qualitatively correct in both magnitudes and exponents.

4.4 Traversals

Now that we have models for active and passive edges, we can combine them to model traversal counts as well. We assume that the chance for a passive edge and an active edge to combine into a traversal is a single probability representing how likely an arbitrary active state is to have a continuation with a label matching an arbitrary passive state. List rule states have only one continuation, while trie rule states in the branch-

more efficient than NOUNARIESHIGH despite having more active states, largely because using the bottoms of chains increases the frequency of tags relative to categories.

[16] This is also false; tags occur slightly more often at the beginnings of rules and less often at the ends.

[17] In general, the p we used was lower for not having modeled tagging ambiguity, but higher for not having modeled the fact that the SCCs are not of size 27.

[18] And does so without any "tweakable" parameters.

[19] Note that the list curves do not compellingly suggest a power law model.

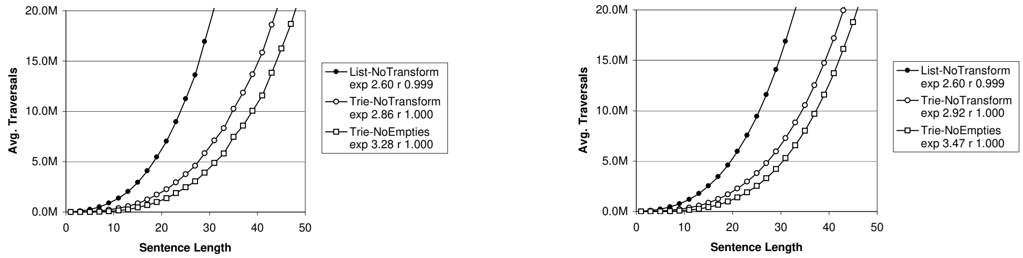

Figure 13: The average number of traversals for sentences of a given length as observed in practice (left), and as predicted by the models presented in the latter part of the paper (right).

ing portion of the trie average about 3.7 (min FSAs 4.2).[20] Making another uniformity assumption, we assume that this combination probability is the continuation degree divided by the total number of passive labels, categorical or tag (73).

In figure 13, we give graphs and exponents of the traversal counts, both observed and predicted, for various settings. Our model correctly predicts the approximate values and qualitative facts, including:

- For List, the observed exponent is lower than for Tries, though the total number of traversals is dramatically higher. This is because the active saturation is growing much faster for Tries; note that in cases like this the lower-exponent curve will never actually outgrow the higher-exponent curve.
- Of the settings shown, only Trie-NoEmpties exhibits super-cubic traversal totals. Despite their similar active and passive exponents, Trie-NoEmpties and Trie-NoTransform vary in traversal growth due to the "early burst" of active edges which gives Trie-NoTransform significantly more edges over short spans than its power law would predict. This excess leads to a sizeable quadratic addend in the number of transitions, causing the average best-fit exponent to drop without greatly affecting the overall magnitudes.

Overall, growth of saturation values in span size increases best-fit traversal exponents, while early spikes in saturation reduce them. The traversal exponents therefore range from List-NoTransform at 2.6 to Trie-NoUnariesLow at over 3.8. However, the final performance is more dependent on the magnitudes, which range from List-NoTransform as the worst, despite its exponent, to Min-NoUnariesHigh as the best. The single biggest factor in the time and traversal performance turned out to be the encoding, which is fortunate because the choice of grammar transform will depend greatly on the application.

[20] This is a simplification as well, since the shorter prefixes that tend to have higher continuation degrees are on average also a larger fraction of the active edges.

5 Conclusion

We built simple but accurate models on the basis of two observations. First, passive saturation is relatively constant in span size, but large due to high reachability among phrasal categories in the grammar. Second, active saturation grows with span size because, as spans increase, the tags in a given active edge are more likely to find a matching arrangement over a span. Combining these models, we demonstrated that a wide range of empirical qualitative and quantitative behaviors of an exhaustive parser could be derived, including the potential super-cubic traversal growth over sentence lengths of interest.

References

James Allen. 1995. *Natural Language Understanding*. Benjamin Cummings, Redwood City, CA.

Eugene Charniak. 1996. Tree-bank grammars. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 1031–1036.

Michael John Collins. 1997. Three generative, lexicalised models for statistical parsing. In *ACL 35/EACL 8*, pages 16–23.

Jay Earley. 1970. An efficient context-free parsing algorithm. *Communications of the ACM*, 6:451–455.

Dan Klein and Christopher D. Manning. 2001. An $O(n^3)$ agenda-based chart parser for arbitrary probabilistic context-free grammars. Technical Report dbpubs/2001-16, Stanford University.

R. Leermakers. 1992. A recursive ascent Earley parser. *Information Processing Letters*, 41:87–91.

Christopher D. Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press, Boston, MA.

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

Robert C. Moore. 2000. Improved left-corner chart parsing for large context-free grammars. In *Proceedings of the Sixth International Workshop on Parsing Technologies*.