Efficiency in Unification-Based N-Best Parsing

Yi Zhang♣, Stephan Oepen♢, and John Carroll♡

♣Saarland University, Department of Computational Linguistics, and DFKI GmbH (Germany)
♢University of Oslo, Department of Informatics (Norway)
♡University of Sussex, Department of Informatics (UK)

Abstract

We extend a recently proposed algorithm for n-best unpacking of parse forests to deal efficiently with (a) Maximum Entropy (ME) parse selection models containing important classes of non-local features, and (b) forests produced by unification grammars containing significant proportions of globally inconsistent analyses. The new algorithm empirically exhibits a linear relationship between processing time and the number of analyses unpacked at all degrees of ME feature non-locality; in addition, compared with agenda-driven best-first parsing and exhaustive parsing with post-hoc parse selection it leads to improved parsing speed, coverage, and accuracy.[†]

1 Background—Motivation

Technology for natural language analysis using linguistically precise grammars has matured to a level of coverage and efficiency that enables parsing of large amounts of running text. Research groups working within grammatical frameworks like CCG (Clark & Curran, 2004), LFG (Riezler et al., 2002), and HPSG (Malouf & van Noord, 2004; Oepen, Flickinger, Toutanova, & Manning, 2004; Miyao, Ninomiya, & Tsujii, 2005) have successfully integrated broad-coverage computational grammars with sophisticated statistical parse selection models. The former delineate the space of possible analyses, while the latter provide a probability distribution over competing hypotheses. Parse selection approaches for these frameworks often use discriminative Maximum Entropy (ME) models, where the probability of each parse tree, given an input string, is estimated on the basis of select properties (called features) of the tree (Abney, 1997; Johnson, Geman, Canon, Chi, & Riezler, 1999). Such features, in principle, are not restricted in their domain of locality, and enable the parse selection process to take into account properties that extend beyond local contexts (i.e. sub-trees of depth one).

There is a trade-off in this set-up between the accuracy of the parse selection model, on the one hand, and the efficiency of the search for the best solution(s), on the other hand. Extending the context size of ME features, within the bounds of available training data, enables increased parse selection accuracy. However, the interplay of the core parsing algorithm and the probabilistic ranking of alternate (sub-)hypotheses becomes considerably more complex and costly when the feature size exceeds the domain of locality (of depth-one trees) that is characteristic of phrase structure grammar-based formalisms. One current line of research focuses on finding the best balance between parsing efficiency and parse selection techniques of increasing complexity, aiming to identify the most probable solution(s) with minimal effort.

This paper explores a range of techniques, combining a broad-coverage, high-efficiency HPSG parser with a series of parse selection models with varying context size of features. We sketch three general scenarios for the integration: (a) a baseline sequential configuration, where all results are enumerated first, and subsequently ranked; (b) an interleaved but approximative solution, performing a greedy search for an n-best list of results; and (c) a two-phase approach, where a complete packed for-

[†]The first author warmly acknowledges the guidance of his PhD advisors, Valia Kordoni and Hans Uszkoreit. We are grateful to Ulrich Callmeier, Berthold Crysmann, Dan Flickinger, and Erik Velldal for many discussions and their support. We thank Ron Kaplan, Martin Kay, and Bob Moore for providing insightful information about related approaches, notably the XLE and CLE parsers.

Proceedings of the 10th Conference on Parsing Technologies, pages 48–59,
Prague, Czech Republic, June 2007.

est is created and combined with a specialized graph search procedure to selectively enumerate results in (globally) correct rank order. Although conceptually simple, the second technique has not previously been evaluated for HPSG parsing (to the best of our knowledge). The last of these techniques, which we call *selective unpacking*, was first proposed by Carroll & Oepen (2005) in the context of chart-based generation. However, they only provide an account of the algorithm for local ME properties and assert that the technique should generalize to larger contexts straightforwardly. This paper describes this generalization of selective unpacking, in its application to parsing, and demonstrates that the move from features that resemble a context-free domain of locality to features of, in principle, arbitrary context size can indeed be based on the same algorithm, but the required extensions are non-trivial.

The structure of the paper is as follows. Section 2 summarizes our formalism, grammars used, parse selection approach, and training and test data. Section 3 discusses the range of possibilities for structuring the process of statistical, grammar-based parsing, and Sections 4 to 6 describe our approach to efficient n-best parsing. We present experimental results in Section 7, compare our approach to previous ones (Section 8), and finally conclude.

2 Overall Set-up

While couched in the HPSG framework, the techniques explored here are applicable to the larger class of unification-based grammar formalisms. We make use of the DELPH-IN[1] reference formalism, as implemented by a variety of systems, including the LKB (Copestake, 2002) and PET (Callmeier, 2002). For the experiments discussed here, we adapted the open-source PET parsing engine in conjunction with two publicly available grammars, the English Resource Grammar (ERG; Flickinger, 2000) and the DFKI German Grammar (GG; Müller & Kasper, 2000, Crysmann, 2005). Our parse selection models were trained and evaluated on HPSG treebanks that are distributed with these grammars. The following paragraphs summarize relevant properties of the structures manipulated by the parser,

[1]Deep Linguistic Processing with HPSG, an open-source repository of grammars and processing tools; see '`http://www.delph-in.net/`'.

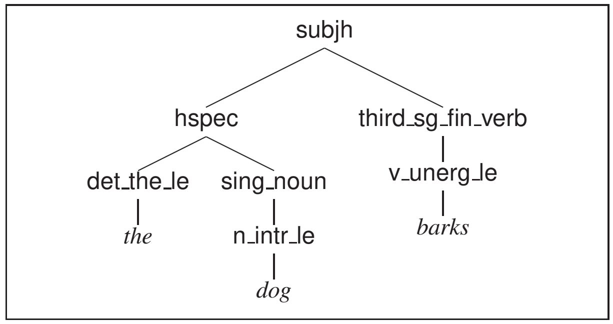


Figure 1: Sample HPSG derivation tree for the sentence *the dog barks*. Phrasal nodes are labeled with identifiers of grammar rules, and (pre-terminal) lexical nodes with class names for types of lexical entries.

followed by relevant background on parse selection.

Figure 1 shows an example ERG derivation tree. Internal tree nodes are labeled with identifiers of grammar rules, and leaves with lexical entries. The derivation tree provides complete information about the actual HPSG analysis, in the sense that it can be viewed as a recipe for computing it. Lexical entries and grammar rules alike are ultimately just feature structures, complex and highly-structured linguistic categories. When unified together in the configuration depicted by the derivation tree, the resulting feature structure yields an HPSG sign, a detailed representation of the syntactic and semantic properties of the input string. Just as the full derivation denotes a feature structure, so do its sub-trees, and for grammars like the ERG and GG each such structure will contain hundreds of feature – value pairs.

Because of the lexicalized nature of HPSG (and similar frameworks) our parsers search for well-formed derivations in a pure bottom-up fashion. Other than that, there are no hard-wired assumptions about the order of computation, i.e. the specific parsing strategy. Our basic set-up closely mimics that of Oepen & Carroll (2002), where edges indexed by sub-string positions in a chart represent the nodes of the tree, recording both a feature structure (as its category label) and the identity of the underlying lexical entry or rule in the grammar. Multiple edges derived for identical sub-strings can be 'packed' into a single chart entry in case their feature structures are compatible, i.e. stand in an equivalence or subsumption relation. By virtue of having each edge keep back-pointers to its daughter edges—the immediate sub-nodes in the tree whose combination resulted in the mother edge—the parse forest provides a complete and *explicit* encoding of all possible results in a maximally compact form.[2] A simple unpacking procedure is obtained from the cross-multiplication of all local combinatorics, which is directly amenable to dynamic programming.

Figure 2 shows a hypothetical forest (on the left), where sets of edges exhibiting local ambiguity have been packed into a single 'representative' edge, viz. the one in each set with one or more incoming dominance arcs. Confirming the findings of Oepen & Carroll (2002), in our experiments packing under feature structure subsumption is much more effective than packing under mere equivalence, i.e. for each pair of edges (over identical sub-strings) that stand in a subsumption relation, a technique that Oepen & Carroll (2002) termed retro-active packing ensures that the more general of the two edges remains in the chart. When packing under subsumption, however, some of the cross-product of local ambiguities in the forest may not be globally consistent. Assume for example that, in Figure 2, edges $\boxed{6}$ and $\boxed{8}$ subsume $\boxed{7}$ and $\boxed{9}$, respectively; combining $\boxed{7}$ and $\boxed{9}$ into the same tree during unpacking can in principle fail. Thus, unpacking effectively needs to deterministically replay unifications, but this extra expense in our experience is negligible when compared to the decreased cost of constructing the forest under subsumption. In Section 3 we argue that this very property, in addition to increasing parsing efficiency, interacts beneficially with parse selection and on-demand enumeration of results in rank order.

Following (Johnson et al., 1999), a conditional ME model of the probabilities of trees $\{t_1 \ldots t_n\}$ for a string s, and assuming a set of feature functions $\{f_1 \ldots f_m\}$ with corresponding weights $\{\lambda_1 \ldots \lambda_m\}$, is defined as:

$$p(t_i|s) = \frac{\exp \sum_j \lambda_j f_j(t_i)}{\sum_{k=1}^{n} \exp \sum_j \lambda_j f_j(t_k)} \qquad (1)$$

[2]This property of parse forests is not a prerequisite of the chart parsing framework. The basic CKY procedure (Kasami, 1965), for example, as well as many unification-based adaptations (e.g. the Core Language Engine; Moore & Alshawi, 1992) merely record the local category of each edge, which is sufficient for the recognition task and simplifies the search. However, reading out complete trees from the chart, then, amounts to a limited form of search, going back to the rules of the grammar itself to (re-)discover decomposition relations among chart entries.

Type	Sample Features
1	⟨0 subjh hspec third_sg_fin_verb⟩
1	⟨1 △ subjh hspec third_sg_fin_verb⟩
1	⟨0 hspec det_the_le sing_noun⟩
1	⟨1 subjh hspec det_the_le sing_noun⟩
1	⟨2 △ subjh hspec det_the_le sing_noun⟩
2	⟨0 subjh third_sg_fin_verb⟩
2	⟨0 subjh hspce⟩
2	⟨1 subjh hspec det_the_le⟩
2	⟨1 subjh hspec sing_noun⟩
3	⟨1 n_intr_le *dog*⟩
3	⟨2 det_the_le n_intr_le *dog*⟩
3	⟨3 ◁ det_the_le n_intr_le *dog*⟩

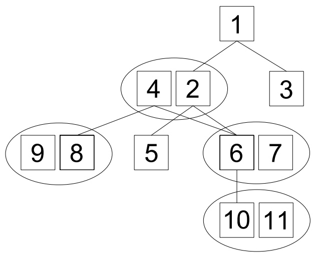
Table 1: Examples of structural features extracted from the derivation tree in Figure 1. The *Type* column indicates the template corresponding to each sample feature; the integer that starts each feature indicates the degree of grandparenting (in the case of type 1 and 2 features) or n-gram size (type 3 features). The symbols △ and ◁ denote the root of the tree and left periphery of the yield, respectively.

Feature functions f_j can test for arbitrary structural properties of analyses t_i, and their value typically is the number of times a specific property is present in t_i. Toutanova, Manning, Flickinger, & Oepen (2005) propose an inventory of features that perform well in HPSG parse selection; currently we restrict ourselves to the best-performing of these, of the form illustrated in Table 1, comprising depth-one sub-trees (or portions of these) with grammar-internal identifiers as node labels, plus optionally a chain of one or more dominating nodes (i.e. levels of grandparents). If a grandparents chain is present then the feature is non-local. For expository purposes, Table 1 includes another feature type, n-grams over leaf nodes of the derivation; in Section 5 below we speculate about the incorporation of these (and similar) features in our algorithm.

3 Interleaving Parsing and Ranking

At an abstract level, given a grammar and an associated ME parse selection model, there are three basic ways of combining them in order to find the single 'best' or small set of n-best results.

The first way is a naïve sequential set-up, in which the parser first enumerates the full set of analyses, computes a score for each using the model, and returns the highest-ranking n results. For carefully



$$\boxed{1} \rightarrow \langle \boxed{2}\,\boxed{3} \rangle \mid \langle \boxed{4}\,\boxed{3} \rangle$$
$$\boxed{2} \rightarrow \langle \boxed{5}\,\boxed{6} \rangle \mid \langle \boxed{5}\,\boxed{7} \rangle$$
$$\boxed{4} \rightarrow \langle \boxed{8}\,\boxed{6} \rangle \mid \langle \boxed{8}\,\boxed{7} \rangle \mid \langle \boxed{9}\,\boxed{6} \rangle \mid \langle \boxed{9}\,\boxed{7} \rangle$$
$$\boxed{6} \rightarrow \langle \boxed{10} \rangle \mid \langle \boxed{11} \rangle$$

Figure 2: Sample forest and sub-node decompositions: ovals in the forest (on the left) indicate packing of edges under subsumption, i.e. edges $\boxed{4}$, $\boxed{7}$, $\boxed{9}$, and $\boxed{11}$ are *not* in the chart proper. During unpacking, there will be multiple ways of instantiating a chart edge, each obtained from cross-multiplying alternate daughter sequences locally. The elements of this cross-product we call *decomposition*, and they are pivotal points both for stochastic scoring and dynamic programming in selective unpacking. The table on the right shows all non-leaf decompositions for our example packed forest: given two ways of decomposing $\boxed{6}$, there will be three candidate ways of instantiating $\boxed{2}$ and six for $\boxed{4}$, respectively, for a total of nine full trees.

crafted grammars and inputs of average complexity the approach can perform reasonably well.

Another mode of operation is to organize the parser's search according to an agenda (i.e. priority queue) that assigns numeric scores to parsing moves (Erbach, 1991). Each such move is an application of the fundamental rule of chart parsing, combining an active and a passive edge, and the scores represent the expected 'figure of merit' (Caraballo & Charniak, 1998) of the resulting structure. Assuming a parse selection model of the type sketched in Section 2, we can determine the agenda priority for a parsing move according to the (unnormalized) ME score of the derivation (sub-)tree that would result from its successful execution. Note that, unlike in probabilistic context-free grammars (PCFGs), ME scores of partial trees do not necessarily decrease as the tree size increases; instead, the distribution of feature weights is in the range $(-\infty, +\infty)$, centered around 0, where negative weights intuitively correspond to dis-preferred properties.

This lack of monotonicity in the scores associated with sub-trees, on the one hand, is beneficial, in that performing a greedy best-first search becomes practical: in contrast, with PCFGs and their monotonically decreasing probabilities on larger sub-trees, once the parser finds the first full tree the chart necessarily has been instantiated almost completely. On the other hand, the same property prohibits the application of exact best-first techniques like A* search, because there is no reliable future cost estimate; in this respect, our set-up differs fundamentally from that of Klein & Manning (2003) and related PCFG parsing work. Using the unnormalized sum of ME weights on a partial solution as its agenda score, effectively, means that sub-trees with low scores 'sink' to the bottom of the agenda; highly-ranked partial constituents, in turn, instigate the immediate creation of larger structures, and ideally the bottom-up agenda-driven search will greedily steer the parser towards full analyses with high scores. Given its heuristic nature, this procedure cannot guarantee that its n-best list of results corresponds to the globally correct rank order, but it may in practice come reasonably close to it. While conceptually simple, greedy best-first search does not combine easily with ambiguity packing in the chart: (a) at least when packing under subsumption, it is not obvious how to accurately compute the agenda score of packed nodes, and (b) to the extent that the greedy search avoids exploration of dis-preferred local ambiguity, the need for packing should be greatly reduced. Unfortunately, in scoring bottom-up parsing moves, ME features involving grandparenting are not applicable, leading to a second potential source of reduced parse selection accuracy. In Section 7 below, we provide an empirical evaluation of both the naïve sequential and greedy best-first approaches.

4 Selective Unpacking

Carroll & Oepen (2005) observe that, at least for grammars like the ERG, the construction of the parse forest can be very efficient (with observed polynomial complexity), especially when packing edges under subsumption. Their selective unpacking procedure, originally proposed for the forest created by a chart *generator*, aims to unpack the n-best set

```
procedure selectively-unpack-edge(edge, n) ≡
  results ← ⟨ ⟩; i ← 0;
  do
    hypothesis ← hypothesize-edge(edge, i); i ← i + 1;
    if (new ← instantiate-hypothesis(hypothesis)) then
      n ← n − 1; results ← results ⊕ ⟨new⟩;
  while (hypothesis and n ≥ 1)
  return results;

procedure hypothesize-edge(edge, i) ≡
  if (edge.hypotheses[i]) return edge.hypotheses[i];
  if (i = 0) then
    for each (decomposition in decompose-edge(edge)) do
      daughters ← ⟨ ⟩; indices ← ⟨ ⟩
      for each (edge in decomposition.rhs) do
        daughters ← daughters ⊕ ⟨hypothesize-edge(edge, 0)⟩;
        indices ← indices ⊕ ⟨0⟩;
      new-hypothesis(edge, decomposition, daughters, indices);
  if (hypothesis ← edge.agenda.pop()) then
    for each (indices in advance-indices(hypothesis.indices)) do
      if (indices ∈ hypothesis.decomposition.indices) then continue
      daughters ← ⟨ ⟩;
      for each (edge in hypothesis.decomposition.rhs) each (i in indices) do
        daughter ← hypothesize-edge(edge, i);
        if (not daughter) then daughters ← ⟨ ⟩; break
        daughters ← daughters ⊕ ⟨daughter⟩;
      if (daughters) then new-hypothesis(edge, hypothesis.decomposition, daughters, indices)
    edge.hypotheses[i] ← hypothesis;
    return hypothesis;

procedure new-hypothesis(edge, decomposition, daughters, indices) ≡
  hypothesis ← new hypothesis(decomposition, daughters, indices);
  edge.agenda.insert(score-hypothesis(hypothesis), hypothesis);
  decomposition.indices ← decomposition.indices ∪ {indices};
```

Figure 3: Selective unpacking procedure, enumerating the n best realizations for a top-level result *edge* from a packed forest. An auxiliary function decompose-edge() performs local cross-multiplication as shown in the examples in Figure 2. Another utility function not shown in pseudo-code is advance-indices(), a 'driver' routine searching for alternate instantiations of daughter edges, e.g. advance-indices($\langle 0\ 2\ 1 \rangle$) $\rightarrow \{\langle 1\ 2\ 1 \rangle\ \langle 0\ 3\ 1 \rangle\ \langle 0\ 2\ 2 \rangle\}$. Finally, instantiate-hypothesis() is the function that actually builds result trees, replaying the unifications of constructions from the grammar (as identified by chart edges) with the feature structures of daughter constituents.

of full trees from the forest, guaranteeing the globally correct rank order according to the probability distribution, with a minimal amount of search. The basic algorithm is a specialized graph search through the forest, with local contexts of optimization corresponding to packed nodes.

Each such node represents local combinatorics, and two key notions in the selective unpacking procedure are the concepts of (a) *decomposing* an edge locally into candidate ways of instantiating it, and of (b) nested contexts of local search for ranked *hypotheses* (i.e. uninstantiated edges) about candidate subtrees. See Figure 2 for examples of the decomposition of edges. Given one decomposition—i.e. a vector of candidate daughters for a particular rule—there can be multiple ways of instantiating each daughter: a parallel index vector $\vec{I} = \langle i_0 \ldots i_n \rangle$ serves to keep track of 'vertical' search among daughter hypotheses, where each index i_j denotes the i-th best instantiation (hypothesis) of the daughter at position j. If we restrict ME features to a depth of one (i.e. without grandparenting), then given the additive nature of ME scores on complete derivations, it can be guaranteed that hypothesized trees including an edge e as an immediate daughter must use the best instantiation of e in their own best instantiation. Assuming a binary rule, the corresponding hypothesis would use daughter indices of $\langle 0\,0 \rangle$. The second-best instantiation, in turn, can be obtained from moving to the second-best hypothesis for *one* of the elements in the (right-hand side of the) decomposition, e.g. indices

$\langle 0\ 1 \rangle$ or $\langle 1\ 0 \rangle$ in the binary example. Hypotheses are associated with ME scores and ordered within each nested context by means of a local priority queue (stored in the original representative edge, for convenience). Therefore, nested local optimizations result in a top-down, breadth-first, exact n-best search through the packed forest, while avoiding exhaustive cross-multiplication of packed nodes.

Figure 3 shows the unchanged pseudo-code of Carroll & Oepen (2005). The main function hypothesize-edge() controls both the 'horizontal' and 'vertical' search, initializing the set of decompositions and pushing initial hypotheses onto the local agenda when called on an edge for the first time (lines $11-17$). For each call, the procedure retrieves the current next-best hypothesis from the agenda (line 18), generates new hypotheses by advancing daughter indices (while skipping over configurations seen earlier) and calling itself recursively for each new index (lines $19-26$), and, finally, arranging for the resulting hypothesis to be cached for later invocations on the same *edge* and *i* values (line 27). Note that unification (in instantiate-hypothesis()) is only invoked on complete, top-level hypotheses, as our structural ME features can actually be evaluated *prior* to building each full feature structure. However, as Carroll & Oepen (2005) suggest, the procedure could be adapted to perform instantiation of sub-hypotheses within each local search, should additional features require it. For better efficiency, the instantiate-hypothesis() routine applies dynamic programming (i.e. memoization) to intermediate results.

5 Generalizing the Algorithm

Carroll & Oepen (2005) offer no solution for selective unpacking with larger context ME features. Yet, both Toutanova et al. (2005) and our own experiments (described in Section 7 below) suggest that properties of larger contexts and especially grandparenting can greatly improve parse selection accuracy. The following paragraphs outline how to generalize the basic selective unpacking procedure, while retaining its key properties: exact n-best enumeration with minimal search. Our generalization of the algorithm distinguishes between 'upward' contexts, with grandparenting with dominating nodes as a representative feature type, and 'downward' extensions, which we discuss for the example of lexical n-gram features.

A naïve approach to selective unpacking with grandparenting might be extending the cross-multiplication of local ambiguity to trees of more than depth one. However, with multiple levels of grandparenting this approach would greatly increase the combinatorics to be explored, and it would pose the puzzle of overlapping local contexts of optimization. Choices made among the alternates for one packed node would interact with other ambiguity contexts in their internal nodes, rather than merely at the leaves of their decompositions. However, it is sufficient to keep the depth of decompositions to minimal sub-trees and rather contextualize each decomposition as a whole. Assuming our sample forest and set of decompositions from Figure 2, let $\langle \boxed{1}\boxed{4} \rangle : \boxed{6} \rightarrow \langle \boxed{10} \rangle$ denote the decomposition of node $\boxed{6}$ in the context of $\boxed{4}$ and $\boxed{1}$ as its immediate parents. When descending through the forest, hypothesize-edge() can, without significant extra cost, maintain a vector $\vec{P} = \langle p_n \ldots p_0 \rangle$ of parents of the current node, for n-level grandparenting. For each packed node, the bookkeeping elements of the graph search procedure need to be contextualized on $\vec{P}$, viz. (a) the edge-local priority queue, (b) the record of index vectors hypothesized already, and (c) the cache of previous instantiations. Assuming each is stored in an associative array, then all references to edge.agenda in the original procedure can be replaced by edge.agenda[$\vec{P}$], and likewise for other slots. With these extensions in place, the original control structure of nested, on-demand creation of hypotheses and dynamic programming of partial results can be retained, and for each packed node with multiple parents ($\boxed{6}$ in our sample forest) there will be parallel, contextualized partitions of optimization. Thus, extra combinatorics introduced in this generalized procedure are confined to only such nodes, which (intuitively at least) appears to establish the lower bound of added search needed—while keeping the algorithm non-approximative. Section 7 provides empirical data on the degradation of the procedure in growing levels of grandparenting and the number of n-best results to be extracted from the forest.

Finally, we turn to enlarged feature contexts that capture information from nodes *below* the elements of a local decomposition. Consider the example of feature type 3 in Table 1, n-grams (of various size) over properties of the yield of the parse tree. For now we only consider lexical *b*i-grams. For an edge e dominating a sub-string of n words $\langle w_i \ \ldots \ w_{i+n-1} \rangle$ there will be $n-1$ bi-grams internal to e, and two bi-grams that interact with w_{i-1} and w_{i+n}—which will be determined by the left- and right-adjacent edges to e in a complete tree. The internal bi-grams are unproblematic, and we can assume that ME weights corresponding to these features have been included in the sum of weights associated to e. Seeing that e may occur in multiple trees, with different sister edges, the selective unpacking procedure has to take this variation into account when evaluating local contexts of optimization.

Let ${}_xe_y$ denote an edge e, with x and y as the lexical types of its leftmost and rightmost daughters, respectively. Returning to our sample forest, assume lexicalizations ${}_\beta\boxed{10}_\beta$ and ${}_\gamma\boxed{11}_\gamma$ (each spanning only one word), with $\beta \neq \gamma$. Obviously, when decomposing $\boxed{4}$ as $\langle \boxed{8}\boxed{6} \rangle$, its ME score, in turn, will depend on the choice made in the expansion of $\boxed{6}$: the sequences $\left\langle {}_\alpha\boxed{8}_\alpha \, {}_\beta\boxed{6}_\beta \right\rangle$ and $\left\langle {}_\alpha\boxed{8}_\alpha \, {}_\gamma\boxed{6}_\gamma \right\rangle$ will differ in (at least) the scores associated with the bigrams $\langle \alpha \, \beta \rangle$ vs. $\langle \alpha \, \gamma \rangle$. Accordingly, when evaluating candidate decompositions of $\boxed{4}$, the number of hypotheses that need to be considered is doubled; as an immediate consequence, there can be up to eight distinct lexicalized variants for the decomposition $\boxed{1} \rightarrow \langle \boxed{4}\boxed{3} \rangle$ further up in the tree. It may look as if combinatorics will cross-multiply throughout the tree—in the worst case returning us to an exponential number of hypotheses—but this is fortunately not the case: regarding the external bi-grams of $\boxed{1}$, node $\boxed{6}$ no longer participates in its left- or rightmost periphery, so variation internal to $\boxed{6}$ is not a multiplicative factor at this level. This is essentially the observation of Langkilde (2000), and her bottom-up factoring of n-gram computation is easily incorporated into our top-down selective unpacking control structure. At the point where hypothesize-edge() invokes itself recursively (line 23 in Figure 3), its return value is now a set of lexicalized alternates, and hypothesis creation (in line 26) can take into account the local cross-product of all such alternation. Including additional properties from non-local subtrees (for example higher-order n-grams and head lexicalization) is a straightforward extension of this scheme, replacing our per-edge left- and rightmost periphery symbols with a generalized vector of externally relevant, internal properties. In addition to traditional (head) lexicalization as we have just discussed it, such extended 'downward' properties on decompositions—percolated from daughters to mothers and cross-multiplied as appropriate—could include metrics of constituent weight too, for example to enable the ME model to prefer 'balanced' coordination structures.

However, given that Toutanova et al. (2005) obtain only marginally improved parse selection accuracy from the inclusion of n-gram (and other lexical) ME features, we have left the implementation of lexicalization and empirical evaluation for future work.

6 Failure Caching and Propagation

As we pointed out at the end of Section 4, during the unpacking phase, unification is only replayed in instantiate-hypothesis() on the top-level hypotheses. It is only at this step that inconsistencies in the local combinatorics are discovered. However, such a discovery can be used to improve the unpacking routine by (a) avoiding further unification on hypotheses that have already failed to instantiate, (b) avoiding creating new hypotheses based on failed sub-hypotheses. This requires some changes to the routines instantiate-hypothesis() and hypothesize-edge(), as well as an extra boolean marker for each hypothesis.

The extended instantiate-hypothesis() starts by checking whether the hypothesis is already marked as failed. If it is not so marked, the routine recursively instantiates all sub-hypotheses. Any failure will again lead to instant return. Otherwise, unification is used to create a new edge from the outcome of the sub-hypothesis instantiations. If this unification fails, the current hypothesis is marked. Moreover, all its ancestor hypotheses are also marked (by recursively following the pointers to the direct parent hypotheses) as they are also guaranteed to fail.

Correspondingly, hypothesize-edge() needs to check the instantiation failure marker to avoid returning hypotheses that are guaranteed to fail. If a hypothesis coming out of the agenda is already marked as failed, it will be used to create new hypotheses (with advance-indices()), but dropped afterward. Subsequent hypotheses will be popped from the agenda until either a hypothesis that is not marked as failed is returned, or the agenda is empty.

Moreover, hypothesize-edge() also needs to avoid creating new hypotheses based on failed sub-hypotheses. When a failed sub-hypothesis is found, the creation of the new hypothesis is skipped. But the index vector $\vec{I}$ may not be simply discarded. Otherwise hypotheses based on advance-indices($\vec{I}$) will not be reachable in the search. On the other hand, simply adding every advance-indices($\vec{I}$) on to the pending creation list is not efficient either in the case where multiple sub-hypotheses fail.

To solve the problem, we compute a failure vector $\vec{F} = \langle f_0 \ldots f_n \rangle$, where f_j is 1 when the sub-hypothesis at position j is known as failed, and 0 otherwise. If a sub-hypothesis at position j is failed then all the index vectors having value i_j at position j must also fail. By putting the result of $\vec{I} + \vec{F}$ on the pending creation list, we can safely skip the failed rows of sub-hypotheses, while not losing the reachability of the others. As an example, suppose we have a ternary index vector $\langle 3\ 1\ 2 \rangle$ for which a new hypothesis is to be created. By checking the instantiation failure marker of the sub-hypotheses, we find that the first and the third sub-hypotheses are already marked. The failure recording vector will then be $\langle 1\ 0\ 1 \rangle$. By putting $\langle 4\ 1\ 3 \rangle = \langle 3\ 1\ 2 \rangle + \langle 1\ 0\ 1 \rangle$ on to the pending hypothesis creation list, the failed sub-hypotheses are skipped.

We evaluate the effects of instantiation failure caching and propagation below in Section 7.

7 Empirical Results

To evaluate the performance of the selective unpacking algorithm, we carried out a series of empirical evaluations with the ERG and GG, in combination with a modified version of the PET parser. When running the ERG we used as our test set the *JH4* section of the LOGON treebank[3], which contains 1603 items with an average sentence length of 14.6 words. The remaining LOGON treebank (of around 8,000 items) was used in training the various ME parse disambiguation models. For the experiment with GG, we designated a 2825-item portion of the DFKI Verb*mobil* treebank[4] for our tests, and trained ME models on the remaining 10,000 utterances. At only 7.4 words, the average sentence length is much shorter in the Verb*mobil* data.

Configuration	**GP**	**Coverage**	**Time (s)**
greedy best-first	0	91.6%	3889
exhaustive unpacking	0	84.5%	4673
selective unpacking	0	94.3%	2245
	1	94.3%	2529
	2	94.3%	3964
	3	94.2%	3199
	4	94.2%	3502

Table 2: Coverage on the ERG for different configurations, with fixed resource consumption limits (of 100k passive edges or 300 seconds). In all cases, up to ten 'best' results were searched, and *Coverage* shows the percentage of inputs that succeed to parse within the available resource. *Time* shows the end-to-end processing time for each batch.

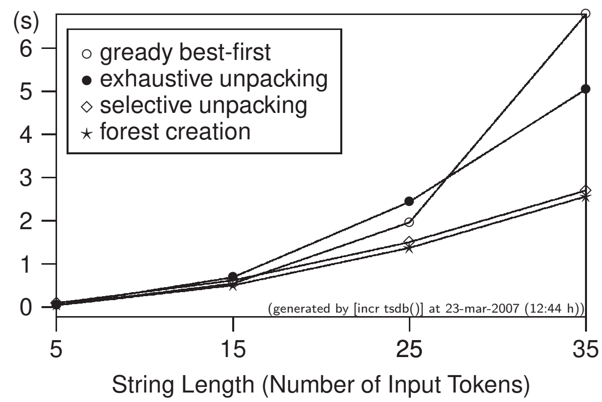


Figure 4: Parsing times for different configurations using the ERG, in all three cases searching for up to ten results, without the use of grandparenting.

We ran seven different configurations of the parser with different search strategies and (un-)packing mechanisms:

- Agenda driven greedy n-best parsing using the ME score without grandparenting features; no local ambiguity packing;
- Local ambiguity packing with exhaustive unpacking, without grandparenting features;

[3]The treebank is comprised of several booklets of edited, instructional texts on backcountry activities in Norway. The data is available from the LOGON web site at 'http://www.emmtee.net'.

[4]The data in this treebank is taken from transcribed appointment scheduling dialogues; see 'http://gg.dfki.de/' for further information on GG and its treebank.

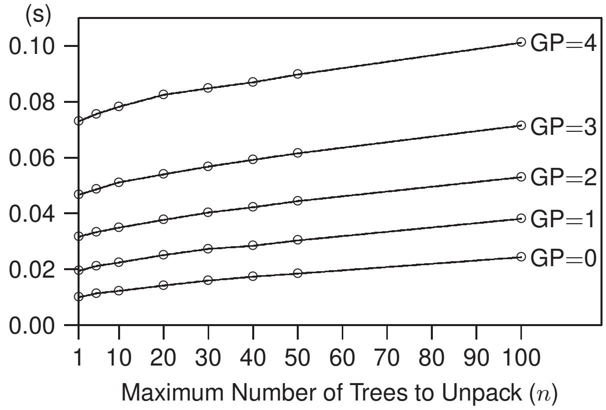


Figure 5: Mean times for selective unpacking of all test items for n-best parsing with the ERG, for varying n and grandparenting (GP) levels

Configuration	Exact Match	Top Ten
random choice	11.34	43.06
no grandparenting	52.52	68.38
greedy best-first	51.79	69.48
grandparenting[1]	56.83	85.33
grandparenting[2]	56.55	84.14
grandparenting[3]	56.37	84.14
grandparenting[4]	56.28	84.51

Table 3: Parse selection accuracy for various levels of grandparenting. The *exact match* column shows the percentage of cases in which the correct tree, according to the treebank, was ranked highest by the model; conversely, the *top ten* column indicates how often the correct tree was among the ten top-ranking results.

- Local ambiguity packing and selective unpacking for n-best parsing, with 0 through 4 levels of grandparenting (GP) features.

As a side-effect of differences in efficiency, some configurations could not complete parsing all sentences given reasonable memory constraints (which we set at a limit of 100k passive edges or 300 seconds processing time per item). The overall coverage and processing time of different configurations on *JH4* are given in Table 2.

The correlation between processing time and coverage is interesting. However, it makes the efficiency comparison difficult as parser behavior is not clearly defined when the memory limit is exceeded. To circumvent this problem, in the following experiments we average only over those 1362 utterances from *JH4* that complete parsing within the resource limit in all seven configurations. Nevertheless, it must be noted that this restriction potentially reduces efficiency differences between configurations, as some of the more challenging inputs (which typically lead to the largest differences) are excluded.

Figure 4 compares the processing time of different configurations. The difference is much more significant for longer sentences (i.e. with more than 15 words). If the parser unpacks exhaustively, the time for unpacking grows with sentence length at a quickly increasing rate. In such cases, the efficiency gain with ambiguity packing in the parsing phase is mostly lost in the unpacking phase. The graph shows that greedy best-first parsing without packing outperforms exhaustive unpacking for sentences of less than 25 words. With sentences longer than 25 words, the packing mechanism helps the parser to overtake greedy best-first parsing, although the exhaustive unpacking time also grows fast.

With the selective unpacking algorithm presented in the previous sections, unpacking time is reduced, and grows only slowly as sentence length increases. Unpacking up to ten results, when contrasted with the timings for forest creation (i.e. the first parsing phase) in Figure 4, adds a near-negligible extra cost to the total time required for both phases. Moreover, Figure 5 shows that with selective unpacking, as n is increased, unpacking time grows roughly linearly for all levels of grandparenting (albeit always with an initial delay in unpacking the first result).

Table 4 summarizes a number of internal parser measurements using the ERG with different packing/unpacking settings. Besides the difference in processing time, we also see a significant difference in *"space"* between exhaustive and selective unpacking. Also, the difference in *"unifications"* and *"copies"* indicates that with our selective unpacking algorithm, these expensive operations on typed feature structures are significantly reduced.

In return for increased processing time (and marginal loss in coverage) when using grandparenting features, Table 3 shows some large improvements in parse selection accuracy (although the picture is less clear-cut at higher-order levels of grandparenting[5]). A balance point between efficiency

[5]The models were trained using the open-source TADM package (Malouf, 2002), using default hyper-parameters for all configurations, viz. a convergence threshold of 10^{-8}, variance of the prior of 10^{-4}, and frequency cut-off of 5. It is likely that

	Configuration	GP	Unifications (#)	Copies (#)	Space (kbyte)	Unpack (s)	Total (s)
≤ 15 words	greedy best-first	0	1845	527	2328	–	0.12
	exhaustive unpacking	0	2287	795	8907	0.01	0.12
	selective unpacking	0	1912	589	8109	0.00	0.12
		1	1913	589	8109	0.01	0.12
		2	1914	589	8109	0.01	0.12
		3	1914	589	8110	0.01	0.12
		4	1914	589	8110	0.02	0.13
> 15 words	greedy best-first	0	25233	5602	24646	–	1.66
	exhaustive unpacking	0	39095	15685	80832	0.85	1.95
	selective unpacking	0	17489	4422	33326	0.03	1.17
		1	17493	4421	33318	0.05	1.21
		2	17493	4421	33318	0.09	1.25
		3	17495	4422	33321	0.13	1.27
		4	17495	4422	33320	0.21	1.34

Table 4: Contrasting the efficiency of various (un-)packing settings in use with ERG on short (top) and medium-length (bottom) inputs; in each configuration, up to ten trees are extracted. *Unification* and *Copies* is the count of top-level FS operations, where only successful unifications require a subsequent copy (when creating a new edge). *Unpack* and *Total* are unpacking and total parse time, respectively.

and accuracy can be made according to application needs.

Finally, we compare the processing time of the selective unpacking algorithm with and without instantiation failure caching and propagation (as described in Section 4 above). The empirical results for GG are summarized in Table 5, showing clearly that the technique reduced unnecessary hypotheses and instantiation failures. The design philosophy of the ERG and GG differ. During the first, forest creation phase, GG suppresses a number of features (in the HPSG sense, not the ME sense) that can actually constrain the combinatorics of edges. This move makes the packed forest more compact, but it implies that unification failures will be more frequent during unpacking. In a sense, GG thus moves part of the search for globally consistent derivations into the second phase, and it is possible for the forest to contain 'result' trees that ultimately turn out to be incoherent. Dynamic programming of instantiation failures makes this approach tractable, while retaining the general breadth-first characteristic of the selective unpacking regime.

further optimization of hyper-parameters for individual configurations would moderately improve model performance, especially for higher-order grandparenting levels with large numbers of features.

8 Discussion

The approach to n-best parsing described in this paper takes as its point of departure recent work of Carroll & Oepen (2005), which describes an efficient algorithm for unpacking n-best trees from a forest produced by a chart-based sentence generator and containing local ME properties with associated weights. In an almost contemporaneous study, but in the context of parsing with treebank grammars, Huang & Chiang (2005) develop a series of increasingly efficient algorithms for unpacking n-best results from a weighted hypergraph representing a parse forest. The algorithm of Carroll & Oepen (2005) and the final one of Huang & Chiang (2005) are essentially equivalent, and turn out to be reformulations of an approach originally described by Jiménez & Marzal (2000) (although expressed there only for grammars in Chomsky Normal Form).

In this paper we have considered ME properties that extend beyond immediate dominance relations, extending up to 4 levels of grandparenting. Previous work has either assumed properties that are restricted to the minimal parse fragments (i.e. subtrees of depth one) that make up the packed representation (Geman & Johnson, 2002), or has taken a more relaxed approach by allowing non-local prop-

Configuration	Unifications (#)	Copies (#)	Hypotheses (#)	Space (kbyte)	Unpack (ms)	Total (ms)
greedy best-first	5980	1447	–	9202	–	400
selective, no caching	5535	1523	1245	27188	70	410
selective, with cache	4915	1522	382	27176	10	350

Table 5: Efficiency effects of the instantiation failure caching and propagation with GG, without grandparenting. All statistics are averages over the 1941 items that complete within the resource bounds in all three configurations. *Unification*, *Copies*, *Unpack*, and *Total* have the same interpretation as in Table 4, and *Hypotheses* is the average count of hypothesized sub-trees.

erties but without addressing the problem of how to efficiently extract the top-ranked trees from a packed forest (Miyao & Tsujii, 2002).

Probably the work closest in spirit to our approach is that of Malouf & van Noord (2004), who use an HPSG grammar comparable to the ERG and GG, non-local ME features, and a two-phase parse forest creation and unpacking approach. However, their unpacking phase uses a beam search to find a good (single) candidate for the best parse; in contrast—for ME models containing the types of non-local features that are most important for accurate parse selection—we avoid an approximative search and *efficiently* identify *exactly* the n-best parses.

When parsing with context free grammars, a (single) parse can be retrieved from a parse forest in time linear in the length of the input string (Billot & Lang, 1989). However, as discussed in Section 2, when parsing with a unification-based grammar and packing under feature structure subsumption, the cross-product of some local ambiguities may not be globally consistent. This means that additional unifications are required at unpacking time. In principle, when parsing with a pathological grammar with a high rate of failure, extracting a single consistent parse from the forest could take exponential time (see Lang (1994) for a discussion of this issue with respect to Indexed Grammars). In the case of GG, a high rate of unification failure in unpacking is dramatically reduced by our instantiation failure caching and propagation mechanism.

9 Conclusions and Future Work

We have described and evaluated an algorithm for efficiently computing the n-best analyses from a parse forest produced by a unification grammar, with respect to a Maximum Entropy (ME) model containing two classes of non-local features. The algorithm is efficient in that it empirically exhibits a linear relationship between processing time and the number of analyses unpacked, at all degrees of ME feature non-locality. It improves over previous work in providing the only exact procedure for retrieving n-best analyses from a packed forest that can deal with features with extended domains of locality and with forests created under subsumption. Our algorithm applies dynamic programming to intermediate results and local failures in unpacking alike.

The experiments compared the new algorithm with baseline systems representing other possible approaches to parsing with ME models: (a) a single phase of agenda-driven parsing with on-line pruning based on intermediate ME scores, and (b) two-phase parsing with exhaustive unpacking and post-hoc ranking of complete trees. The new approach showed better speed, coverage, and accuracy than the baselines.

Although we have dealt with the non-local ME features that in previous work have been found to be the most important for parse selection (i.e. grandparenting and n-grams), this does not exhaust the full range of features that could possibly be useful. For example, it may be the case that accurately resolving some kinds of ambiguities can only be done with reference to particular parts—or combinations of parts—of the HPSG feature structures representing the analysis of a complete constituent. To deal with such cases we are currently designing an extension to the algorithms described here which would add a ‘controlled’ beam search, in which the size of the beam was limited by the interval of score adjustments for ME features that could only be evaluated once the full linguistic structure became available. This approach would involve a constrained amount of extra search, but would still produce the exact n-best trees.

References

Abney, S. P. (1997). Stochastic attribute-value grammars. *Computational Linguistics*, *23*, 597 – 618.

Billot, S., & Lang, B. (1989). The structure of shared forests in ambiguous parsing. In *Proceedings of the 27th Meeting of the Association for Computational Linguistics* (pp. 143 – 151). Vancouver, BC.

Callmeier, U. (2002). Preprocessing and encoding techniques in PET. In S. Oepen, D. Flickinger, J. Tsujii, & H. Uszkoreit (Eds.), *Collaborative language engineering. A case study in efficient grammar-based processing.* Stanford, CA: CSLI Publications.

Caraballo, S. A., & Charniak, E. (1998). New figures of merit for best-first probabilistic chart parsing. *Computational Linguistics*, *24*(2), 275 – 298.

Carroll, J., & Oepen, S. (2005). High-efficiency realization for a wide-coverage unification grammar. In R. Dale & K. F. Wong (Eds.), *Proceedings of the 2nd International Joint Conference on Natural Language Processing* (Vol. 3651, pp. 165 – 176). Jeju, Korea: Springer.

Clark, S., & Curran, J. R. (2004). Parsing the WSJ using CCG and log-linear models. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics* (pp. 104 – 111). Barcelona, Spain.

Copestake, A. (2002). *Implementing typed feature structure grammars.* Stanford, CA: CSLI Publications.

Crysmann, B. (2005). Relative clause extraposition in German. An efficient and portable implementation. *Research on Language and Computation*, *3*(1), 61 – 82.

Erbach, G. (1991). A flexible parser for a linguistic development environment. In O. Herzog & C.-R. Rollinger (Eds.), *Text understanding in LILOG* (pp. 74 – 87). Berlin, Germany: Springer.

Flickinger, D. (2000). On building a more efficient grammar by exploiting types. *Natural Language Engineering*, *6 (1)*, 15 – 28.

Geman, S., & Johnson, M. (2002). Dynamic programming for parsing and estimation of stochastic unification-based grammars. In *Proceedings of the 40th Meeting of the Association for Computational Linguistics.* Philadelphia, PA.

Huang, L., & Chiang, D. (2005). Better k-best parsing. In *Proceedings of the 9th International Workshop on Parsing Technologies* (pp. 53 – 64). Vancouver, Canada.

Jiménez, V. M., & Marzal, A. (2000). Computation of the n best parse trees for weighted and stochastic context-free grammars. In *Proceedings of the Joint International Workshops on Advances in Pattern Recognition* (pp. 183 – 192). London, UK: Springer-Verlag.

Johnson, M., Geman, S., Canon, S., Chi, Z., & Riezler, S. (1999). Estimators for stochastic 'unification-based' grammars. In *Proceedings of the 37th Meeting of the Association for Computational Linguistics* (pp. 535 – 541). College Park, MD.

Kasami, T. (1965). *An efficient recognition and syntax algorithm for context-free languages* (Technical Report # 65-758). Bedford, MA: Air Force Cambrige Research Laboratory.

Klein, D., & Manning, C. D. (2003). A* parsing. Fast exact Viterbi parse selection. In *Proceedings of the 4th Conference of the North American Chapter of the ACL.* Edmonton, Canada.

Lang, B. (1994). Recognition can be harder than parsing. *Computational Intelligence*, *10*(4), 486 – 494.

Langkilde, I. (2000). Forest-based statistical sentence generation. In *Proceedings of the 1st Conference of the North American Chapter of the ACL.* Seattle, WA.

Malouf, R. (2002). A comparison of algorithms for maximum entropy parameter estimation. In *Proceedings of the 6th Conference on Natural Language Learning.* Taipei, Taiwan.

Malouf, R., & van Noord, G. (2004). Wide coverage parsing with stochastic attribute value grammars. In *Proceedings of the IJCNLP workshop Beyond Shallow Analysis.* Hainan, China.

Miyao, Y., Ninomiya, T., & Tsujii, J. (2005). Corpus-oriented grammar development for acquiring a Head-Driven Phrase Structure Grammar from the Penn Treebank. In K.-Y. Su, J. Tsujii, J.-H. Lee, & O. Y. Kwong (Eds.), *Natural language processing* (Vol. 3248, pp. 684 – 693). Hainan Island, China.

Miyao, Y., & Tsujii, J. (2002). Maximum entropy estimation for feature forests. In *Proceedings of the Human Language Technology Conference.* San Diego, CA.

Moore, R. C., & Alshawi, H. (1992). Syntactic and semantic processing. In H. Alshawi (Ed.), *The Core Language Engine* (pp. 129 – 148). Cambridge, MA: MIT Press.

Müller, S., & Kasper, W. (2000). HPSG analysis of German. In W. Wahlster (Ed.), *Verbmobil. Foundations of speech-to-speech translation* (Artificial Intelligence ed., pp. 238 – 253). Berlin, Germany: Springer.

Oepen, S., & Carroll, J. (2002). Efficient parsing for unification-based grammars. In S. Oepen, D. Flickinger, J. Tsujii, & H. Uszkoreit (Eds.), *Collaborative language engineering. A case study in efficient grammar-based processing* (pp. 195 – 225). Stanford, CA: CSLI Publications.

Oepen, S., Flickinger, D., Toutanova, K., & Manning, C. D. (2004). LinGO Redwoods. A rich and dynamic treebank for HPSG. *Journal of Research on Language and Computation*, *2*(4), 575 – 596.

Riezler, S., King, T. H., Kaplan, R. M., Crouch, R., Maxwell III, J. T., & Johnson, M. (2002). Parsing the Wall Street Journal using a Lexical-Functional Grammar and discriminative estimation techniques. In *Proceedings of the 40th Meeting of the Association for Computational Linguistics.* Philadelphia, PA.

Toutanova, K., Manning, C. D., Flickinger, D., & Oepen, S. (2005). Stochastic HPSG parse selection using the Redwoods corpus. *Journal of Research on Language and Computation*, *3*(1), 83 – 105.