Learning Stochastic Lexicalized Tree Grammars from HPSG

Günter Neumann
DFKI, Saarbrücken
neumann@dfki.de

Dan Flickinger
CSLI, Stanford University
danf@csli.stanford.edu

June 23, 1999

Abstract

We[1] present a method for automatically extracting a Stochastic Lexicalized Tree Grammar (SLTG) from an HPSG source grammar and a given corpus. Processing of a SLTG is performed by a specialized fast parser. The approach has been tested on a large English grammar and has been shown to achieve a speed-up by a factor of better than 10 compared to parsing with a highly tuned HPSG parser. Our approach is simple and transparent, and comes with no magic tuning strategies. The extracted grammars are declaratively represented and have a high degree of practical applicability.

1 Introduction

Head Driven Phrase Structure Grammar (HPSG) has proven to be a quite successful formalism for specifying natural language grammars in a highly modular and compact manner [Pollard and Sag, 1994] supporting the definition of complex linguistic information and interactions between information

[1]Neumann was supported by a research grant from the German Federal Ministry of Education, Science, Research and Technology (BMBF) to the DFKI project PARADIME, FKZ:ITW 9704. Flickinger's contribution was funded both by BMBF in the framework of the Verbmobil Project under Grant FKZ:01IV7024, and by the U.S. National Science Foundation under grant number IRI-9612682. We would like to thank Tilman Becker, Gregor Erbach and Mark-Jan Nederhof for fruitful discussions about this work.

on different strata systematically using typed feature constraints. On the other hand, inefficiency in processing such grammars is the major obstacle for using the HPSG formalism in practical NL applications [Makino *et al.*, 1998].

In recent years several approaches have been proposed to improve the performance of an HPSG system. For example, Van Noord describes a selective memoization technique used in combination with head-corner parsing [van Noord, 1997]. Where Van Noord focuses on improved parsing strategies, [Makino *et al.*, 1998] focus on the compilation of typed feature structures into an abstract machine for attribute-value logics. Another approach is described in [Kasper *et al.*, 1995] which focuses on the compilation of HPSG into Tree Adjoining Grammar (TAG).

Common to all of these approaches is that the coverage of the original general grammars is not affected. Thus if the original grammar is defined domain-independently it might also define a great many theoretically valid analyses covering a wide range of plausible linguistic constructions, including the rarest cases. However, in building real-world applications it has been shown to be a fruitful compromise to focus on frequency and plausibility of linguistic structures wrt. a certain domain. Recently, a number of attempts have been made to automatically adapt a general grammar to a corpus in order to achieve efficiency through domain-adaptation, e.g., using PATR-style grammars [Briscoe and Carroll, 1993; Samuelsson, 1994; Rayner and Carter, 1996] or lexicalized TAGs [Srinivas, 1997].

In this paper we apply the idea of domain-adaptation as an alternative approach to improve efficient processing of HPSG grammars. Beyond efficiency, major goals of our approach are 1) to take full advantage of the HPSG formalism when constructing the specialized grammar, and 2) to favor simple and transparent strategies and interfaces, such that the method can be used without any details of the learning strategy, and such that the extracted grammars are declarative.

The basic idea of our approach is to parse all sentences of a representative training corpus using an HPSG grammar and parser in order to automatically extract from the parsing results a *stochastic lexicalized tree grammar* (SLTG) such that each resulting parse tree is decomposed from the top downwards into a set of subtrees such that each non-terminal non-headed subtree is cut off, and the cutting points are marked for substitution. The same process is then recursively applied to each extracted subtree. Each extracted tree is automatically lexically anchored and each node label of the extracted tree compactly represents a set of relevant features by means of a simple symbol (these are specific category labels defined as part of the HPSG source grammar). For each extracted tree a frequency counter is used to estimate the probability $p(t)$ of a tree t, after all parse trees have been processed.

Processing of an SLTG is performed by an LTAG-like parser (cf. [Schabes and Joshi, 1991]), such that in a first step a new sentence is completely parsed using the extracted SLTG, followed by a second step where the SLTG-valid parse trees are expanded *off-line* by applying the corresponding feature constraints of the original HPSG-grammar (giving our approach an LFG-style processing flavor). The performance of both steps directly depends on the number of SLTG-valid parse trees found for a new sentence. Our approach has many advantages:

- An SLTG has context-free power and meets the criteria of a lexicalized CFG.

- The off-line step guarantees that no information of the original grammar is lost (including semantics). Furthermore, because the number of valid SLTG-trees computed before the off-line process is started is usually small (about 12) complete SLTG-parsing is fast (better than factor 12 compared with the original HPSG system).

- Since the SLTG learning phase is driven by the HPSG principles of the source grammar, the whole extraction process is simple, transparent and non-redundant wrt. grammatical information.

- Our approach allows a proper embedding of statistical information into HPSG structures, which supports preference-based and robust processing.

- The whole approach has an important application impact, e.g., for controlled language processing or information extraction, making use of rich and linguistically motivated information from HPSG. Furthermore, the same mechanism can also be applied to NL generation (cf. [Neumann, 1997]), as well as to efficient interleaved parsing and generation (cf. [Neumann, 1998b]).

The approach has been fully implemented and tested with an advanced HPSG-system. The grammar used in our study is the English Resource

Grammar being developed as part of the LinGO (Linguistic Grammars On-line) project at CSLI, Stanford University. The grammar consists of about 7000 types, arranged in a multiple-inheritance hierarchy which defines the properties of lexical entries, lexical rules, and syntactic phrase structure rules. The lexicon includes hand-built entries for about 5000 stems, along with the full set of inflectional lexical rules and 15 derivational rules which are executed at run time. Syntactic coverage of the grammar is relatively broad, with a central focus on providing precise semantic interpretations for each phenomenon that is assigned an analysis, using the Minimal Recursion Semantics framework of [Copestake *et al.*, 1995; Copestake *et al.*, 1997] (see [Oepen and Flickinger, 1998] for more detailed discussion of the grammar's coverage and of the issues related to such measurement).

The rest of the paper is organized as follows. In section 2 we describe details of the learning phase. In section 3, the parsing strategy is described, as well as some implementational issues. In section 4 we report on several experiments which have been performed in order to measure the performance of the SLTG parsing results. We relate our work to others in section 5 and give a conclusion and outline of future work in section 6.

2 Learning of SLTG

Learning of an SLTG starts by first parsing each sentence s_i of the training corpus with the source HPSG system. The resulting feature structure fs_{s_i} of each example also contains the parse tree pt_i, where each non-terminal node contains the label of the HPSG-rule schema (e.g, head-complement rule) that has been applied during the corresponding derivation step as well as a pointer to the feature structure of the corresponding sign. The label of each terminal node consists of the lexical type of the corresponding feature structure. Each parse tree pt_i is now processed by the following interleaved steps (see also figure 1).

Decomposition Each parse tree is decomposed from the top downwards into a set of subtrees such that each non-headed subtree is cut off[2], and

[2]Testing whether a phrase is a headed phrase or not can be done very easily by checking whether the top-level type of a rule's label feature structure is a subtype of a general *headed phrase* which is defined in an HPSG grammar. The same holds for adjunct phrases (see below).

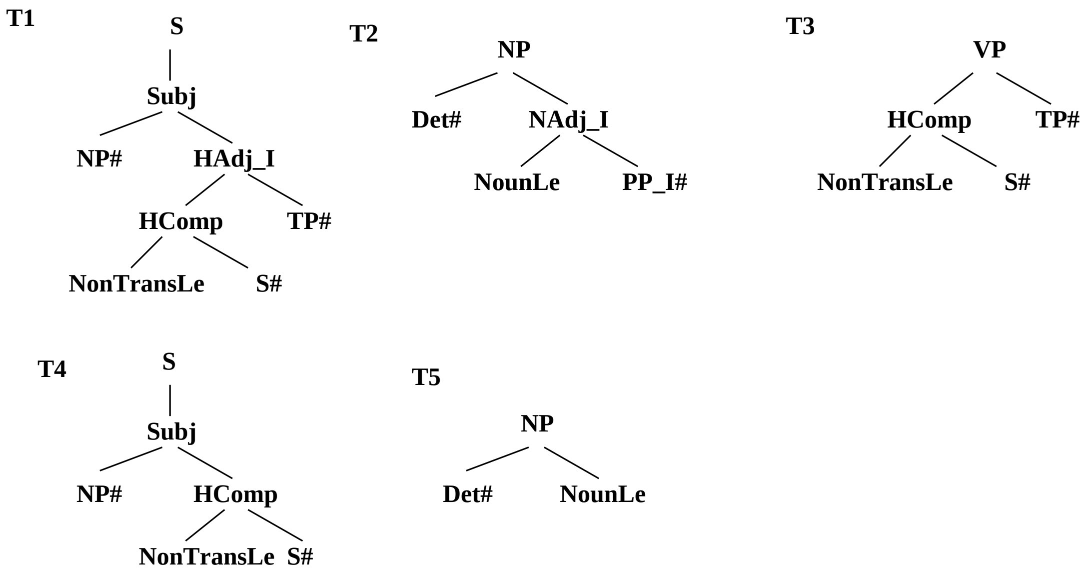

Figure 1: Some trees extracted from *I guess we need to figure out a day, within the next two months.* The symbols S, NP, VP, TP, Det, PP_I have been determined by means of specialization (see text). # denotes the substitution marker.

the cutting point is marked for substitution. The decomposition process is recursively applied on each extracted subtree. Each extracted tree is automatically lexically anchored, where the anchor consists of a lexical type and the path from the lexical anchor to the root defines a head-chain. For each extracted tree a frequency counter is used to estimate its probability after all parse trees of the training set have been processed.

In order to automatically enrich the coverage of the extracted grammar, two additional operations will be performed during decomposition. Firstly, each subtree of the head-chain is copied and the copied tree is processed individually by the decomposition operation (e.g., in figure 1, the tree $T3$ is copied from the head chain of $T1$.). This means that a phrase which occurs only in a head-position in the training corpus can now also be used in nonhead-positions by the SLTG-parser when parsing new sentences. Secondly, if the SLTG-tree has a modifier phrase attached, then a new tree is created with the modifier "unattached" (applied recursively, if the tree has more than one modifier). Unattachment of a modifier m is simply done by raising the head daughter into the position of m (e.g, in figure 1, the tree $T4$ is obtained by replacing the subtree rooted at $HAdj\_I$ of tree $T1$ with the subtree rooted at $HComp$. In a similar way, $T5$ is created from $T2$).[3] The advantage of unattaching modifiers is that we will be able to also recognize sentences with fewer or no modifiers using our extracted grammar. Note that the possible maximum number n of modifier sequences is constrained by the training corpus, i.e., modifiers are implicitly represented as iterations from 0 to n.

Specialization The root node as well as all substitution nodes of an extracted tree are further processed by replacing the rule label with a corresponding *category label*. The possible set of category labels is defined in the type hierarchy of the HPSG source grammar. They express equivalence classes for different phrasal signs. For example, phrasal signs whose value of the LOCAL.CAT.HEAD feature is of type *noun*, and whose value of the LOCAL.CAT.VAL.SUBJ feature is the empty list, are classified as NPs. Now, if the associated feature structure of a rule label $HeadAdjunct$ of the current training sentence is subsumed by the NP type, then $HeadAdjunct$ is replaced by NP. Note that this step actually performs a specialization of the current tree, because the same tree might occur in another tree in verbal position. In that case, $HeadAdjunct$ might be replaced by the type VP.

The definition of category labels is declarative. Thus it is possible to define more fine-grained labels directly as part of the source grammar leading to more specific SLTG trees. This can be done by the grammar writer without knowing any details of the learning strategy.

Probability For each extracted tree a frequency counter is maintained. After all parse trees of the training set have been decomposed and specialized, we estimate a tree's probability as follows: First we count the total number n of all trees which have the same root label α. Then we divide the frequency number m of a tree t with root α by n. This gives the probability $p(t)$. In doing so, the sum of all probabilities of trees t_i with root α is 1 (i.e., $\sum_{t_i:root(t_i)=\alpha} p(t_i) = 1$).

[3] The current grammar is binary branching, so it suffices to raise the head daughter. If a flat representation of modifiers is used one would remove the modifier daughters first, before the head daughter would be raised.

Properties of an SLTG Compared to the original HPSG source grammar, the SLTG has the following properties:

- An SLTG has context-free power, since it allows center embedding (note that it has been extracted from a finite set of parse trees). Recursion of modifiers is actually "spelled out" where the training sentence with the largest number of modifiers defines the upper bound.[4]

- A SLTG cannot derive the empty element, because every SLTG tree has a lexical anchor and they are finitely ambiguous, because every tree introduces a lexical item into the resulting string. Thus an SLTG meats the criteria of lexicalized CFG.

- SLTG trees have a larger domain of locality than the HPSG rule schemata or CF rules. Thus parsing can benefit from a high degree of top-down information. Furthermore, since every tree is lexically anchored, parsing will produce less spurious ambiguities than would be the case for P-CFG.

- Since decomposition is linguistically guided, the resulting SLTG-trees are linguistically oriented. On the other hand, the statistical information expresses a relationship between lexical types and their probable syntactic context.

Representation of an SLTG Once an SLTG has been learned all trees are stored in an array, and a lexicon is computed using all lexical anchors (types) from the SLTG. Each lexical element keeps a list of indexes for the corresponding trees in the array. In addition a table is compiled from the HPSG source grammar, which assigns to each lexical type its corresponding phrase type. This table is used for speeding up parsing of an SLTG. All data

[4] It would have been possible to express modifier recursion through adjunction following [Schabes and Waters, 1995]. In that case the size of the grammar would be smaller; however parsing could become more complex on average, at least if we would adapt the strategy described in [Schabes and Waters, 1995]. This claim is only intuitive. However it is made on the basis that our parser directly operates on the SLTG-trees, where [Schabes and Waters, 1995] first "explode" a tree by mapping it to a set of local trees of depth one. These local trees define the items in the chart parser. Compared to that, our system treats whole trees as items, and thus has to process significantly fewer items.

structures of the SLTG are stored in a grammar file in readable form, so that the complete grammar can easily be loaded and processed when needed.[5]

3 Parsing of SLTG

Parsing of an SLTG is performed by a chart-based agenda-driven bottom-up parser (following [Schabes and Joshi, 1991]). The two major steps are:

- tree selection: selection of a set of SLTG-trees associated with the lexical items in the input sentence,
- tree combination: parsing of the sentence with respect to this set.

After parsing, each SLTG-parse tree is "expanded" by unifying the feature constraints of the HPSG source grammar into the parse trees. If successful, it determines a complete valid feature structure wrt. the HPSG grammar (e.g., all agreement and semantic information). If unification fails, the SLTG parse tree is rejected, and the next most likely tree is expanded.

Tree selection All SLTG-trees are lexically anchored and have at least one substitution node. A lexical anchor is a lexical type. Since different words can have the same lexical type, they are comparable to pre-terminal elements in a CFG. Thus for each word w_i of a new input sentence we determine its lexical type l_i (in general we will have a set of types) and use this information to retrieve the corresponding set of SLTG-trees T_i with anchor l_i.[6] All extracted trees and all pairs $\langle w_i, l_i \rangle$ are used to initialize the SLTG-parser's chart, where $\langle w_i, l_i \rangle$ is interpreted as a passive item (i.e., an item which has no substitution node; thus they serve as terminal elements), and a selected

[5] The interested reader might inspect the example grammar accessible from "http://www.dfki.de/ ˜neumann/sltg/vmgram.sltg.1112c" which has been computed using the training set found in file "vmsents.txt".

[6] In the HPSG source system, lexical processing consists of morphological processing followed by unification of lexical-specific syntactic and semantic information. In order to speed up processing and to be able to use the whole SLTG system without the HPSG system, we have compiled the lexicon into a full-form lexicon, which directly gives us the set of possible lexical types for a given word-form in the input sentence. Lexical access is a little bit more complicated than simple lookup because we have to deal with multi-word entries, e.g., "a little bit". The current full-form lexicon has more than 25,000 entries.

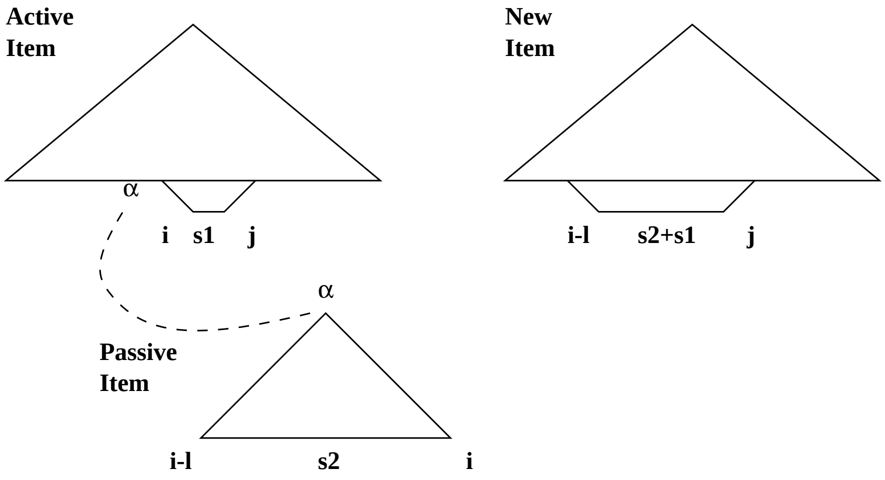

Figure 2: Left substitution of an active item with string $s1$ and span $\langle i,j \rangle$ with a passive item with string $s2$ and span $\langle i\text{-}l,i \rangle$ yields a new item with string $s2{+}s1$ and span $\langle i\text{-}l,j \rangle$.

SLTG-tree is interpreted as an active item, because it has at least one substitution node. The position of the word in the input sentence is used as initial span information for an item.

Note that once tree selection has been performed, s subset of the grammar has been selected, i.e., no more trees can enter the parsing step (first experiments seem to support the assumption made in [Schabes and Joshi, 1991] that the tree selection phase is the most significant factor for a two-phase parsing strategy; see also sec. 4). Furthermore, the anchor positions express restrictions on the possible combinations of trees, because an anchor actually splits the set of trees it can combine with into a left and right subset (see also [Schabes and Joshi, 1991]).

Tree combination This is the process of building a complete parse tree for the input sentence by combining active and passive items until one passive item for the whole input has been found. The main operation is tree substitution, which substitutes a substitution node with label α of an active item with span $\langle i,j \rangle$ with a passive item with root node α with span $\langle i\text{-}l,i \rangle$ (left substitution, where $1 \leq l \leq i$) or $\langle j,j{+}k \rangle$ (right substitution, where 1

$\leq k \leq n$-j & n = length(input)) (see also figure 2).[7] The probability of a new item is obtained by multiplying the probability of the passive and active items. Terminal elements, i.e. passive items created directly from the input words, receive probability one.

Thus described, tree combination extends the string of an active item from its lexical anchor in the left as well as the right direction. However, the whole input string is processed incrementally left-to-right controlled by a chart-based agenda-driven bottom-up control strategy. Both strategies together will automatically complete all left substitution nodes of an active item before the right substitution nodes are considered from left to right (we call this the *left completion condition*). Only items contained in the chart can be combined. If tree combination creates a new item it is inserted into the agenda from which it will eventually be inserted into the chart. The agenda mechanism sorts new items based on an item's probability p and its span $\langle i,j \rangle$ computing $p+(j\text{-}i)$, i.e. "most probable, longest matching".

Some pruning techniques are applied in order to keep the chart as small as possible (see also 5). Currently, pruning is used for filtering out impossible combinations of SLTG-trees, namely those which would lead to invalid spans of new items.

4 Experiments

We have conducted initial experiments using a corpus of 425 utterances from dialogs about scheduling of meetings, with an average sentence length of 7.8. Compared to purely statistical approaches the corpus size seems to be too small to be of value for any predictions. However, in comparison with related approaches to grammar specialization the corpus size is actually a good start. For example, [Srinivas, 1997] uses one corpus with 300 and one with 1500 sentences. Although [Rayner and Carter, 1996] use a final set of about 15,000 sentences, they also show the growth of their grammar wrt. a smaller corpus of some hundred sentences, which allows a first preliminary comparison. Of course, we need to test our approach on corpora with at least several thousand sentences to get reliable predictions. However, we do not expect to run it on million-word data sets.

[7] We are using the Gorn number for efficiently referring to a substitution node in an active item (see also [Schabes, 1990]).

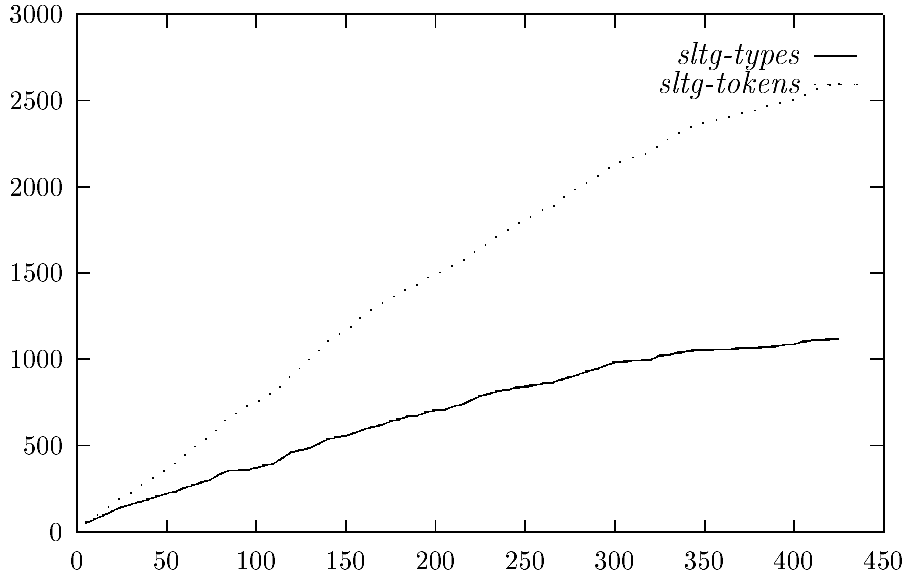

Figure 3: Growth of the extracted grammar during training. *sltg-types* refers to the number of different SLTG-trees, and *sltg-token* to the total number of trees extracted during training (including multiple occurences).

Training We run the whole corpus during the training phase. The size of the extracted SLTG is 1116 trees and the average number of trees with the same lexical anchor is 9.55. The growth of the extracted grammar as well as the total number of trees extracted during training is shown in figure 3 . So far the figure displays the behaviour that we expect, namely that the number of different constructions (or new SLTG-trees) decreases when the training set increases, whereas the total number of trees increases.

Performance Using the extracted SLTG-grammar we ran initial performance tests using the training corpus. The average *run-time* of the SLTG-parser (i.e., without off-line expansion, but including morphological and lexical pre-processing) is 170 msec for all readings and 20 msec for the best reading. The overall speed (i.e., including lexical lookup and off-line expansion) is improved by a factor of 12 compared to parsing with the original highly tuned HPSG parser at the time of our study.

Figure 4 shows the number of readings the SLTG-parser has found. From

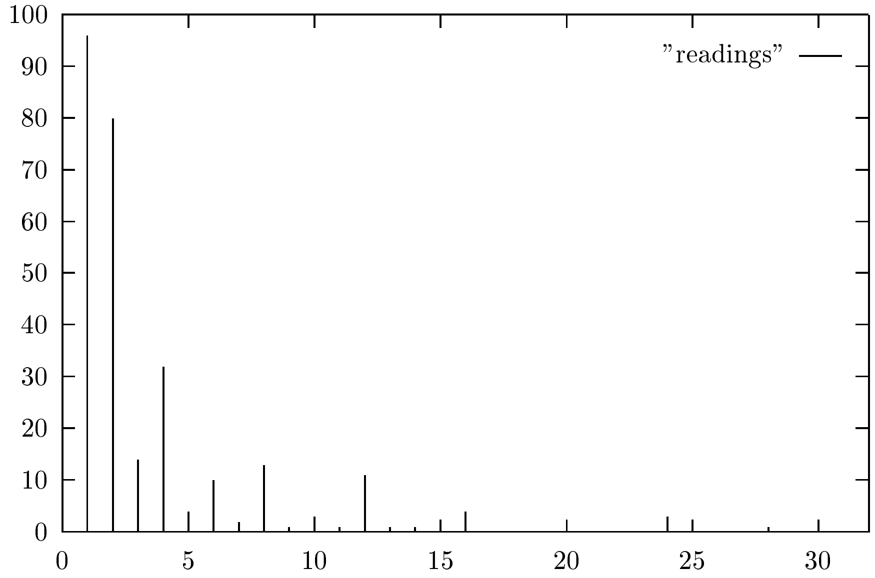

Figure 4: In almost all cases the SLTG-parser computes less than 16 readings per sentence.

the curve we can see that for most sentences the number of readings lies between 1 and 12 and that only very few sentences have extreme numbers of readings (in one pathalogical case we had 1024).

The average number of SLTG-trees retrieved in the first phase of the SLTG-parser (i.e., before tree combination is performed) is 229. This means that only about 200 trees have to be considered during tree combination. We expect that this number actually will not increase very much as the training size increases. This assumption is supported by the results on experiments performed for a similar learning approach applied on tree banks (see [Neumann, 1998a]). Here, stochastic lexicalized tree grammars were also extracted from tree banks. For example, applying the method on 4270 sentences from the Penn treebank [Marcus *et al.*, 1993], a grammar was obtained containing about 31944 trees with words as anchor. However, the average number of trees per word was about 500 when running the training corpus.

Once an SLTG-parse tree has been computed the off-line expansion process applies all HPSG-constraints. If successful the resulting feature structure is correct wrt. the HPSG source grammar, which means that our approach guarantees 100% precision (as long as we do not relax some of the HPSG-constraints). This implies that if for the first SLTG-parse tree t_i of some sentence that was expanded successfully, we know that the previous $(i-1)$ SLTG-parse trees are incorrect. Therefore, the number $(i-1)$ is an important measure for the accuracy of the SLTG-parser *without* off-line expansion. We can expect that the average number of readings is less than or equal to the number of readings found by the SLTG-parser (see 4). In the optimal case $(i-1)$ is just 0, which means that the SLTG-parser is guaranteed to compute valid SLTG-parse trees without the need of off-line expansion. Applying the extracted grammar on the training corpus showed that actually two-thirds of the corpus succeed on either the first or the second parse. This is a good promise, but still needs more data to achieve real reliability. However, if this number will not increase very much, then it would mean that an SLTG's overgeneration before off-line expansion is small, meaning that SLTG trees already have a high degree of accuracy.

5 Related work

An SLTG is structurally related to lexicalized TAGs. For that reason one might argue that an SLTG could also be obtained by first compiling an HPSG grammar into an LTAG (preserving full coverage, cf. [Kasper *et al.*, 1995]), and then to approximate the LTAG to a lexicalized CFG. Although we are not aware of any work describing such an approach, the advantage of our approach is that it combines both steps into one.

Our approach is also related to approaches of grammar specialization based on Explanation-based Learning (EBL), cf. [Samuelsson, 1994; Srinivas and Joshi, 1995; Rayner and Carter, 1996]. Samuelsson [1994] presents a method in which tree decomposition is completely automated using the information-theoretical concept of entropy, after the whole set of parse trees has been indexed in an and-or tree. This implies that a new grammar has to be computed if the tree bank changes (i.e., reduced incrementality) and that the generality of the induced subtrees depends much more on the size and variation of the tree banks than ours. On the other hand, this approach seems to be more sensitive to the distribution of sequences of lexical anchors than our approach, so that we will explore its integration. Rayner and Carter [1996] describe an approach in which EBL and constituent pruning is presented. They showed that constituent pruning (i.e. the removal of edges of a chart that are relatively unlikely to contribute to correct analysis) in combination with EBL increases system performance considerably, such that processing times of other components (lexical and morphological processing) become crucial. It is not difficult to integrate these pruning strategies into our parser (see sec. 3) which would help us to keep the chart as small as possible during parsing of one sentence.

In [Carroll and Weir, 1997] a classification of different methods (named S(1) to S(4)) for encoding frequency information in lexicalized grammars is given and applied to prominent existing approaches. The one with the most globally-dependent frequencies is S(4) and Bod's DOP framework [Bod, 1995] actually belongs to that class. We compute the frequency information in the same way as in DOP once a parse tree has been decomposed. The main difference here is that we only allow one lexical anchor per tree, where Bod also allows for multiple anchors. Although this implies that we might need less training data than DOP we have to pay the price of reduced lexical co-occurrence restrictions during application, which makes our approach weaker than DOP. However, it is not difficult to extend our approach to employ multiple lexical anchors (the "tree-bank" version described in [Neumann, 1998a] and mentioned above makes use of multiple lexical anchors). In that case it would be possible to evaluate the trade-off between training size and coverage of the extracted grammar.

6 Conclusion and Future Work

We have presented an approach for improving parsing with HPSG by automatically extracting a stochastic lexicalized tree grammar. Grammar extraction is based on linguistically guided tree decomposition taking full advantage of the HPSG principles. An extracted grammar is still "HPSG-based" in the sense that the structures computed by the SLTG-parser are fully compatible with HPSG. Thus our approach could be characterized as a corpus-driven method for "de-compressing" the search space of an HPSG grammar rather than for compiling it into some other format. Nevertheless, the extracted grammars have interesting relationships to lexicalized TAGs which allow us to adapt basic parsing technology from that area. We also showed how to integrate stochastic information into the extracted grammars which supports incorporation of statistically based parsing strategies into our HPSG-based approach. Our next steps involve more detailed evaluation, improved pruning, as well as further experimentation with an increased set of node labels, which can help to reduce the number of trees retrieved in parsing. We also have begun to explore strategies for robust processing, especially handling of unknown words. The main idea here is to classify unknown words according to the context of SLTG-trees which are anchored by known words.

References

[Bod, 1995] R. Bod. *Enriching Linguistics with Statistics: Performance Models of Natural Language*. PhD thesis, University of Amsterdam, 1995. ILLC Dissertation Series 1995-14.

[Briscoe and Carroll, 1993] T. Briscoe and J. Carroll. Generalized probabilistic lr parsing of natural language (corpora) with unification-based grammars. *Computational linguistics*, 19(1):25–59, 1993.

[Carroll and Weir, 1997] J. Carroll and D. Weir. Encoding frequency information in lexicalized grammars. In *Proceedings of 5th ACL/SIGPARSE International Workshop on Parsing Technologies*, MIT, Cambridge, 1997.

[Copestake *et al.*, 1995] A. Copestake, D. Flickinger, R. Malouf, S. Riehemann, and I. Sag. Translation using minimal recursion semantics. In *Proceedings, 6th International Conference on Theoretical and Methodological Issues in Machine Translation*, 1995.

[Copestake *et al.*, 1997] A. Copestake, D. Flickinger, and I. Sag. Minimal recursion semantics: An introduction. CSLI, Stanford University., 1997.

[Kasper *et al.*, 1995] R. Kasper, B. Kiefer, K. Netter, and K. Vijay-Shanker. Compilation of hpsg into tag. In *33rd Annual Meeting of the Association for Computational Linguistics*, Cambridge, MA, 1995.

[Makino *et al.*, 1998] T. Makino, M. Yoshida, K. Torisawa, and J. Tsujii. Lilfes – towards a practical hpsg parser. In *Proceedings of the 36th ACL and 17th Coling*, pages 807 – 811, Montreal, 1998.

[Marcus *et al.*, 1993] M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19:313–330, 1993.

[Neumann, 1997] G. Neumann. Applying explanation-based learning to control and speeding-up natural language generation. In *35th Annual Meeting of the Association for Computational Linguistics/8th Conference of the European Chapter of the Association for Computational Linguistics*, Madrid, Spain, 1997.

[Neumann, 1998a] G. Neumann. Automatic extraction of stochastic lexicalized tree grammars from treebanks. In *4th workshop on tree-adjoining grammars and related frameworks*, Philadelphia, PA, USA, August 1998.

[Neumann, 1998b] G. Neumann. Interleaving natural language parsing and generation through uniform processing. *Artifical Intelligence*, 99:121–163, 1998.

[Oepen and Flickinger, 1998] S. Oepen and D. Flickinger. Towards systematic grammar profiling: Test suite technology 10 years after. *Computer Speech and Language*, 12:411–435, 1998.

[Pollard and Sag, 1994] C. Pollard and I. M. Sag. *Head-Driven Phrase Structure Grammar*. Center for the Study of Language and Information Stanford, 1994.

[Rayner and Carter, 1996] M. Rayner and D. Carter. Fast parsing using pruning and grammar specialization. In *34th Annual Meeting of the Association for Computational Linguistics*, Morristown, New Jersey, 1996.

[Samuelsson, 1994] C. Samuelsson. *Fast Natural-Language Parsing Using Explanation-Based Learning*. PhD thesis, Swedish Institute of Computer Science, Kista, Sweden, 1994.

[Schabes and Joshi, 1991] Y. Schabes and A. K. Joshi. Parsing with lexicalized tree adjoining grammar. In M. Tomita, editor, *Current Issues in Parsing Technology*, pages 25–48. Kluwer, Boston, 1991.

[Schabes and Waters, 1995] Y. Schabes and R. Waters. Tree insertion grammar: A cubic-time parsable formalism that lexicalizes context-free grammar without changing the trees produced. *Computational Linguistics*, 21:479–513, 1995.

[Schabes, 1990] Y. Schabes. *Mathematical and Computational Aspects of Lexicalized Grammars*. PhD thesis, University of Pennsylvania, Philadelphia, USA, 1990.

[Srinivas and Joshi, 1995] B. Srinivas and A. Joshi. Some novel applications of explanation-based learning to parsing lexicalized tree-adjoining grammars. In *33rd Annual Meeting of the Association for Computational Linguistics*, Cambridge, MA, 1995.

[Srinivas, 1997] B. Srinivas. *Complexity of Lexical Restrictions and Its Relevance to Partial Parsing*. PhD thesis, University of Pennsylvania, 1997. IRCS Report 97–10.

[van Noord, 1997] G. van Noord. An efficient implementation of the head-corner parser. *Computational Linguistics*, 23:425–456, 1997.