Chapter 1

AUTOMATIC EXTRACTION OF STOCHASTIC LEXICALIZED TREE GRAMMARS FROM TREEBANKS

Günter Neumann
DFKI
66123 Saarbrücken, Germany
neumann@dfki.de

Abstract We present a uniform method for the extraction of stochastic lexicalized tree grammars (SLTG) of different complexities from existing treebanks as well as from competence-based grammars, which allows us to analyze the relationship of a grammar automatically induced from a treebank wrt. its size, its complexity, and its predictive power on unseen data. Processing of different SLTG is performed by a stochastic version of the two-step Early-based parsing strategy introduced in Schabes and Joshi, 1991.

1. INTRODUCTION

In this paper we present a uniform method for the extraction of stochastic lexicalized tree grammars (SLTG) from existing treebanks as well as from competence-based grammars, especially Head-Driven Phrase Structure grammars HPSG, Pollard and Sag, 1994. The use of SLTGs is motivated for two reasons. First, it is assumed that SLTGs better capture distributional and hierarchical information than stochastic CFG (cf. Schabes, 1992; Schabes and Waters, 1996), and second, they allow the factorization of recursion of different kinds, viz. extraction of left, right, and wrapping auxiliary trees and possible combinations. Processing of different SLTG is performed by a stochastic version of the two-phase Early-based parsing strategy introduced in Schabes and Joshi, 1991.
Existing treebanks are used because they allow a corpus-based analysis of grammars of realistic size. HPSG is used in order to extract a domain-independent, phenomena-oriented subgrammar. The ultimate goal then is to merge SLTGs extracted from both sources in order to 1) improve the coverage of treebank grammars on unseen data, and 2) in order to ease adaptation of treebanks to new domains (see also 6.).

This abstract describes work in progress. So far, we have concentrated on the automatic extraction of SLTGs of different kinds. This phase is completed and we will report on first experiments using the PennTreebank, Marcus et al., 1993, Negra, a treebank for German, Skut et al., 1997; Brants et al., 1999, and a broad coverage English HPSG-based grammar (see section 4.). A first version of the two-phase parser is implemented for both types of grammars, and we have started first tests concerning its performance.

2. GRAMMAR EXTRACTION

Given a set of parse trees grammar extraction is the process of decomposing each parse tree into smaller units called subtrees. In our approach, the underlying decomposition operation

1. should yield lexically anchored subtrees, and

2. should be guided by linguistic principles.

The motivation behind (1) is the observation that in practice stochastic CFGs perform worse than non-hierarchical approaches, and that lexicalized tree grammars may be able to capture both distributional and hierarchical information, Schabes and Waters, 1996. Concerning (2) we want to take advantage of the linguistic principles explicitly or implicitly used to define a treebank. This is motivated by the hypothesis that it will better support the development of on-line or incremental learning strategies (the cutting criteria are less dependent from the quantity and quality of the existing treebank than purely statistically based approaches, see also section 5.) and that it renders possible a comparison and integration of an induced grammar with a linguistically based competence grammar. Both aspects (but especially the latter one) are of importance because it is possible to apply the same learning strategy also to a treebank computed by some competence grammar, and to investigate methods for combining treebanks and competence grammars (see section 6.).
Figure 1.1 The Negra parse tree of the sentence “Schade jedoch, daß kaum jemand daran teilhaben kann.” (*Unfortunately however that almost nobody participate could) and some of the extracted SLTG-trees. # denotes the substitution marker.

3. SLTG FROM TREEBANKS

First we will focus on the use of existing treebanks using the Penn-Treebank (Marcus et al., 1993) and the Negra-treebank, a treebank for German, Skut et al., 1997. In the Negra-treebank, dependence theory has been chosen in order to account for the free word order property of German. The Negra-treebank follows an hybrid framework that combines the advantage of phrase-structure and dependency grammars: They do employ phrasal nodes, but try to keep the structure flat such that a phrasal node mostly corresponds to one lexical head (see figure 1.1). The branches of such tree may cross in order to treat non-local dependencies. Negra comes with a tool that transforms the Negra-format to the Penn-format by transforming crossing edges into non-crossing edges and by the introduction of corresponding gap nodes, Skut et al., 1997; Brants et al., 1999. We are using these transformed Negra-trees in our experiments.

We suggested that tree-decomposition should be guided by linguistic principles. A major aspect here is an assumed classification of the parse tree nodes into head and modifier nodes. Using HPSG this is (should be) a quite simple task directly making use of the HPSG-principles. In
case of treebanks we assume that the treebank comes with a notion of lexical and phrasal head, i.e., with a kind of head principle. In the Negra-treebank, head and modifier elements are explicitly tagged. For example, each head is marked by the suffix HD or NK. In case of the Penn-treebank, the head relation is determined manually and stored in a head-perlocation table (see also Charniak, 1997). In case it is not possible to uniquely identify one head element there exists a parameter called direction which specifies whether the left or right candidate should be selected. Note that by means of this parameter we can also specify whether the resulting grammar should prefer a left or right branching.

Using the head information, each tree from the treebank is decomposed from the top downwards into a set of subtrees, such that each non-terminal non-head subtree is cut off, and the cutting point is marked for substitution. The same process is then recursively applied to each extracted subtree. Due to the assumed head notion each extracted tree will automatically be lexically anchored (and the path from the lexical anchor to the root can be seen as a head-chain). Furthermore, every terminal element which is a sister of a node of the head-chain will also remain in the extracted tree. Thus, the yield of the extracted tree might contain several terminal substrings, which gives interesting patterns of word or POS sequences (see also figure 1.1). For each extracted tree a frequency counter is used to compute the probability $p(t)$ of a tree $t$, after the whole treebank has been processed, such that $\sum_{t: \text{root}(t) = \alpha} p(t) = 1$, where $\alpha$ denotes the root label of a tree $t$.

After a tree has been decomposed completely we obtain a set of lexicalized elementary trees where each nonterminal of the yield is marked for substitution. In a next step the set of elementary trees is divided into a set of initial and auxiliary trees. The set of auxiliary trees is further subdivided into a set of left, right, and wrapping auxiliary trees following Schabes and Waters, 1995 (using special foot note labels, like :lfoot, :rfoot, and :wfoot). Note that the identification of possible auxiliary trees is strongly corpus-driven. Using special foot note labels allows us to trigger carefully the corresponding inference rules. For example, it might be possible to treat the :wfoot label as the substitution label, which means that we consider the extracted grammar as a SLTIG, or only highly frequent wrapping auxiliary trees will be considered. It is also possible to treat every foot label as the substitution label, which means that the extracted grammar only allows for substitution. At that point we have to stress that we do not factor out modifier recursion explicitly from the Penn-treebank. The major reason is that arguments and modifiers for the same head are both sisters of the head. In the Negra-treebank, modifiers are explicitly marked by means of the suffix
However, because the parse trees are flat we cannot simply factor out recursion without changing the topological structure of the parse trees. For that reason we “re-do” modifier attachment by iteratively visiting all modifier nodes of an elementary tree $etree$. In each iteration, $etree$ is copied and the current modifier is destructively deleted from $etree$ (the same approach is also performed using the HPSG-based strategy, see section 4.).

### 3.1 TWO-PHASE PARSING OF SLTG

The resulting SLTG will be processed by a two-phase stochastic parser along the line of Schabes and Joshi, 1991. In a first step the input string is used for retrieving the relevant subset of elementary trees. Note that the yield of an elementary tree might consist of a sequence of lexical elements. Thus in order to support efficient access, the deepest leftmost chain of lexical elements is used as index to an elementary tree. Each such index is stored in a decision tree. The first step is then realized by means of a recursive tree traversal which identifies all (longest) matching substrings of the input string (see also section 3.2). Parsing of lexically triggered trees is performed in the second step using an Earley-based strategy. In order to ease implementation of different strategies, the different parsing operations are expressed as inference rules and controlled by a chart-based agenda strategy along the line of Shieber et al., 1995. So far, we have implemented a version for running SLTIG which is based on Schabes and Waters, 1995. The inference rules can be triggered through boolean parameters, which allows flexible hiding of auxiliary trees of different kinds.

### 3.2 EXPERIMENTS

We will briefly report on first results of our method using the Negra treebank (4270 sentences) and the section 02, 03, 04 from the Penn treebank (the first 4270 sentences). In both cases we extracted three different versions of SLTG (note that no normalization of the treebanks has been performed): (a) lexical anchors are words, (b) lexical anchors are part-of-speech, and (c) all terminal elements are substituted by the constant :term, which means that lexical information is ignored. For each grammar we report the number of elementary trees, left, right, and wrapping auxiliary trees. The following table summarizes the results:
In a second experiment we evaluated the performance of the implemented SLTIG parser using the extracted Penn treebank with words as lexical anchors. We applied all sentences on the extracted grammar and computed the following average values for the first phase: sentence length: 27.54, number of matching substrings: 15.93, number of elementary trees: 492.77, number of different root labels: 33.16. The average run-time for each sentence (measured on a Sun Ultra 2 (200 mhz): 0.0231 sec. In a next step we tested the run-time behaviour of the whole parser on the same input. The average run-time for each sentence (exhaustive mode): 6.18 sec. This is promising, since the parser is still not optimized.

We also tried first blind tests, but it turned out that the current considered size of the treebanks is too small to get reliable results on unseen data (randomly selecting 10 % of a treebank for testing; 90 % for training). The reason is that if we consider only words as anchors then we rarely get a complete parse result very often because of unknown words and different punctuation. If we consider only POS then the number of elementary trees retrieved through the first phase increases causing the current parser prototype to be slow (due to the restricted annotation schema). A better strategy seems to be the use of words only for lexical anchors and POS for all other terminal nodes, or to use only closed-class words as lexical anchors (assuming a head principle based on functional categories). In that case it would also be possible to adapt the strategies described in Srinivas, 1997 wrt. supertagging in order to reduce the set of retrieved trees before the second phase is called.

4. SLTG FROM HPSG

Basically the same approach has been applied on a set of parse trees computed by using an English HPSG-grammar. The grammar used in our study is the English Resource Grammar being developed as part of the LinGO (Linguistic Grammars Online) 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).

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 $f_{s_i}$ 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 1.2).

Each parse tree is decomposed from the top downwards into a set of subtrees such that each non-head subtree is cut off as described above. In case of the HPSG-grammar testing whether a phrase is a head phrase or not can be done very easily by checking whether the top-level type of

![Figure 1.2 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.](image-url)
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).

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 1.2, the tree $T_3$ is copied from the head chain of $T_1$). 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 1.2, the tree $T_4$ is obtained by replacing the subtree rooted at $H_{Adj,J}$ of tree $T_1$ with the subtree rooted at $H_{Comp}$. In a similar way, $T_5$ is created from $T_2$). 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 $n$.

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.

After all parse trees of the training set have been decomposed and specialized, we compute a tree’s probability as described in section 3.
Experiments. We trained the HPSG-system on a corpus of 2000 sentences from dialogs (with an average length of 11.5 words per sentence). The size of the extracted SLT of the head-chain and unattachment) is 1922 elementary trees. Performing copy of trees of the head chain gives 3828 trees, considering only unattachment as additional operation gives 2427 elementary trees. Applying both operations gives an SLT with a total of 4195 trees.

Using the extracted SLT-grammar we ran initial performance tests using the training corpus. The average run-time of the SLT-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 1.3 shows the number of readings the SLT-parser has found. From 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 pathological case we had 1024).
5. RELATED WORK

Here we will discuss alternative approaches for converting treebanks into lexicalized tree grammars, namely the Data-oriented Parsing (DOP) framework Bod, 1995 and approaches based on applying Explanation-based Learning (EBL) to NL parsing (e.g., Samuelsson, 1994; Srinivas, 1997).

The general strategy of our approach is similar to DOP with the notable distinction that in our framework all trees must be lexically anchored and that in addition to substitution, we also consider adjunction and restricted versions of it. Furthermore, since DOP tries to compute all possible decompositions of a treebank, the training phase is very complex (actually it is exponential), where our approach is polynomial because we consider only a subset of all possible decompositions. In the EBL approach to NL parsing the core idea is to use a competence grammar and a training corpus to construct a treebank. The treebank is then used to obtain a specialized grammar which can be processed much faster than the original one at the price of a small loss in coverage. Samuelsson Samuelsson, 1994 presents a method in which tree decomposition is completely automatized using the information-theoretical concept of entropy, after the whole treebank has been indexed in an and-or tree. This implies that a new grammar has to be computed if the treebank changes (i.e., reduced incrementallity) and that the generality of the induced subtrees depends much more on the size and variation of the treebank than ours. On the other side, 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.

In Srinivas, 1997 the application of EBL to parsing of LTAG is presented. The core idea is to generalize the derivation trees generated by an LTAG and to allow for a finite state transducer representation of the set of generalized parses. The POS sequence of a training instance is used as the index to a generalized parse. Generalization wrt. recursion is achieved by introducing the Kleene star into the yield of an auxiliary tree that was part of the training example, which allows generalization about the length of the training sentences. This approach is an important candidate for improvements of our two-phase parser once we have acquired an S-LTAG.

6. FUTURE STEPS: TOWARDS MERGING SLTG

In the next future, we want to investigate methods for merging SLTGs extracted from both sources in order to 1) improve the coverage of tree-
bank grammars on unseen data, and 2) in order to ease adaptation of
treebanks to new domains. The core idea behind 1) is the extension of
an SLTG extracted from a treebank through the domain-independent
SLTG extracted from HPSG. It seems that current state-of-the-art tree-
bank grammars can achieve an accuracy of about 87% (see Charniak,
1997). We want to explore whether integrating knowledge from a com-
petence grammar can improve the accuracy. We believe that SLTGs
are suited since they better capture distributional and hierarchical in-
formation than stochastic CFGs. The major obstacle for merging the
current grammars are the different nature of syntactic constituent lev-
els. For example, the Penn-treebank modifier structure is flat compared
to that of an HPSG-based SLTG. Recently, Xia, 1999 has shown how
the Penn-treebank might be fully bracketed in order to factor out the
recursive structures for elementary trees. This is actually done by in-
serting additional non-terminal nodes into the treebank trees on the ba-
sis of Penn-treebank specific head-perlocation and argument table lists.
Using an HPSG-based SLTG it would be possible to use HPSG-based
trees as static tree-patterns and to create new trees from most similar
trees found in the treebank-SLTG such that the HPSG-SLTG serves as
“building plans”. 5

Another line of future research will be the use of an HPSG-SLTG in
order to initialize the induction of a domain-specific SLTG on the basis
of a small number of annotated parse trees or even on raw sentences.
Now, it would be possible to adapt a HPSG-SLTG to a new domain
following a minimally supervised learning approach.

Acknowledgments

The research underlying this paper was supported by a research grant from the
German Bundesministerium für Bildung, Wissenschaft, Forschung und Technologie
(BMBF) to the DFKI project paradime (“Parametrizable Domain-adaptive Infor-
mation and Message Extraction”), FKZ ITW 9704. I would like to thank Thorsten
Brants for providing me with the Negra Corpus, as well as Tilman Becker and Anoop
Sarkar for many helpful comments.

Notes

1. This paper is an extension of a previous version published in the *Proceedings of the
4th workshop on tree-adjoining grammars and related frameworks*, Philadelphia, PA, USA,
August, 1998.

2. Recently, Xia, 1999 has shown how the Penn-treebank might be fully bracketed in
order to factor out the recursive structures for elementary trees. In principle this is done
by using a modifier table and transforming the flat sequence of modifier trees into a binary
structure. Although this seems to be a quite treebank specific approach, the idea is worthwhile
for our future goal of merging treebanks (see section 6.).
3. Applying the same test as described above on POS, the average number of elementary
trees retrieved is 2292.86, i.e., the number seems to increase by a factor of 5.
4. This part of the work has been carried out together with Dan Flickinger, from CSLI,
Stanford. For a more detailed description of the approach see Neumann and Flickinger, 1999.
5. The same might also be explored wrt. a competence-based LTAG, like the one which
comes with the XTAG system Doran et al., 1994.

References

Bod, R. (1995). Enriching Linguistics with Statistics: Performance Models of Natural Language. PhD thesis, University of Amsterdam. ILLC Dissertation Series 1995-14.
Brants, T., Skut, W., and Uszkoreit, H. (1999). Syntactic annotation of a German newspaper corpus. In Proceedings of the ATALA Treebank Workshop, pages 69–76, Paris, France.
Charniak, E. (1997). Statistical parsing with a context-free grammar and word statistics. In AAAI-97, Providence, Rhode Island.
Copestake, A., Flickinger, D., Malouf, R., Riehemann, S., and Sag, I. (1995). Translation using minimal recursion semantics. In Proceedings, 6th International Conference on Theoretical and Methodological Issues in Machine Translation.
Copestake, A., Flickinger, D., and Sag, I. (1997). Minimal recursion semantics: An introduction. CSLI, Stanford University.
Doran, C., Egedi, D., Hockey, B., Srinivas, B., and Zeidel, M. (1994). Xtag system - a wide coverage grammar for english. In Proceedings of the 15th International Conference on Computational Linguistics (COLING), Kyoto, Japan.
Marcus, M. P., Santorini, B., and Marcinkiewicz, M. A. (1993). Building a large annotated corpus of english: The penn treebank. Computational Linguistics, 19:313–330.
Neumann, G. and Flickinger, D. (1999). Learning stochastic lexicalized tree grammars from hpsg. Technical report, DFKI, Saarbrücken.
Oepen, S. and Flickinger, D. (1998). Towards systematic grammar profiling: Test suite technology 10 years after. Computer Speech and Language, 12:411–435.
Pollard, C. and Sag, I. M. (1994). Head-Driven Phrase Structure Grammar. Center for the Study of Language and Information Stanford.
Samuelsson, C. (1994). Grammar specialization through entropy thresholds. In Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics, pages 188–195.
Schabes, Y. (1992). Stochastic lexicalized tree-adjoining grammars. In Proceedings of the 14th International Conference on Computational Linguistics (COLING), pages 426–432, Nantes.
Schabes, Y. and Joshi, A. K. (1991). Parsing with lexicalized tree adjoining grammar. In Tomita, M., editor, Current Issues in Parsing Technology, pages 25–48. Kluwer, Boston.

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

Schabes, Y. and Waters, R. (1996). Stochastic lexicalized tree-insertion grammar. In Bunt, H. and Tomita, M., editors, Recent Advances in Parsing Technology, pages 281–294. Kluwer Academic Press, London.

Shieber, S., Schabes, Y., and Pereira, F. (1995). Principles and implementation of deductive parsing. Journal of Logic and Computation, 24:3–36.

Skut, W., Krenn, B., Brants, T., and Uszkoreit, H. (1997). An annotation scheme for free word order languages. In 5th International Conference of Applied Natural Language, pages 88–94, Washington, USA.

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

Xia, F. (1999). Extracting tree adjoining grammars from bracketed corpora. In Proceedings of the 5th Natural Language Processing Pacific Rim Symposium(NLPRS-99), Beijing, China.