Chapter 1

A UNIFORM METHOD FOR AUTOMATICALLY EXTRACTING STOCHASTIC LEXICALIZED TREE GRAMMARS FROM TREEBANKS AND HPSG

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 with respect to 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.

Keywords: Grammar extraction, treebanks, Head-Driven Phrase Structure Grammars, general learning method

1. INTRODUCTION

In this paper[1] 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 capture distributional and hierarchical information better than stochastic CFG (cf. Schabes, 1992; Schabes and Waters, 1996), and second, they allow the factorization of different kinds of recursion, viz. extraction of left, right, and wrapping auxiliary trees and possible combinations thereof. Processing of different SLTG is performed using a stochas-

tic version of the two-phase Early-based parsing strategy introduced in Schabes and Joshi, 1991.

Existing treebanks are used because they allow 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 to 2) ease adaptation of treebanks to new domains (see also 6.).

2. RELATED WORK

Before describing our method in detail, we will first discuss alternative approaches for automatically converting treebanks into tree grammars, namely the Data-oriented Parsing (DOP) framework and approaches based on applying Explanation-based Learning (EBL) to NL parsing.

The general strategy of our approach is similar to DOP (the original DOP model was presented in Bod, 1993, but see also Bod, 2000) 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), whereas our approach is polynomial since we consider only a subset of all possible decompositions. A similar approach can be found in van Genabith et al., 2000 who derive an LFG grammar from treebanks, and Xia, 1999; Chen and Vijay-Shanker, 2000; Chiang, 2000 who derive lexicalized tree adjoining grammars from treebanks (based on and partially extending the work presented in Neumann, 1998 and in this paper).

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, 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 incrementality) and that the generality of the induced subtrees depends much more on the size and variation of the treebank than ours. On the other hand, there approach seems to be more sensitive to the distribution of sequences of lexical anchors than ours, so that we will explore its integration in the future.

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 with respect to 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.

3. 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 2.) and that it renders possible a comparison and integration of a grammar which has been extracted from a treebank with a linguistically based competence grammar. Both aspects (but especially the latter one) are of importance because it is also possible to apply the same extraction strategy to a treebank computed by some competence grammar, and to investigate novel methods for combining treebanks and competence grammars (see section 6.).

A common extraction strategy. In the following we introduce a general method for the extraction of stochastic lexicalized tree grammars (SLTG) from treebanks and HPSG, before its specialization to the different data sources at hand are described in the sections 1.2 and 5.

The extraction method is applied on a set of sentences which have been annotated with their corresponding syntactic tree structure or *parse trees*. For the extraction method, it does not matter what source has been in order to determine the parse trees. Thus, the method abstracts

away from the fact whether the set of sentences has been decorated manually (a treebank) or whether a parser and a source grammar has been used to compute the parse trees automatically.

The major operation of the extraction method is a recursive tree decomposition operation starting from the root node of a parse tree. As already noted above, the tree decomposition operation should be guided by linguistically oriented decomposition principles. During the traversal of a tree these principles specify which subtrees of a current node should be cut off or not. A major aspect for the definition of meaningful cutting criteria is an assumed classification of the parse tree nodes into head and modifier nodes. We can now define a quite simple *head–driven decomposition principle*: cut off all non-head subtrees. Using HPSG this is (should be) quite a simple task directly making use of the HPSG-principles. In case of treebanks, we assume that a treebank comes with a notion of lexical and phrasal head, i.e., with a kind of *head principle*.

Now we can describe the performance of the extraction method more precisely as follows: Using the head–driven decomposition principle, each tree from the set of parse trees 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 $\sum_{t:root(t)=\alpha} p(t) = 1$, where α denotes the root label of a tree t.

Lexical anchors. Each extracted subtree will automatically be lexically anchored. The lexical anchors are later used as indices for the retrieval of subtrees during parsing. We have parameterized our extraction method with respect to the exact definition of what counts as a lexical anchor, e.g., a word, stem, part-of-speech (POS) and in case of HPSG the lexical type of a lexical element. This allows us to investigate the trade-off between space (i.e., size of grammar) and time (i.e., more combinatorial power). For example, if we use words as lexical anchors, then each word of an input sentence will be able to identify its individual set of trees. However, the size of the grammar will be quite large. On

the contrary, if we use POS as lexical anchors the size of the SLTG will be smaller, but different words will retrieve the same set of trees if they belong to the same POS (see also sec. 4.1).

Additional operations. In order to automatically enrich the coverage of the extracted grammar, two additional operations are 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 1.3, 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 1.3, 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 $T2$). 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.

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 may 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 4.1). 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 different kinds of auxiliary trees .

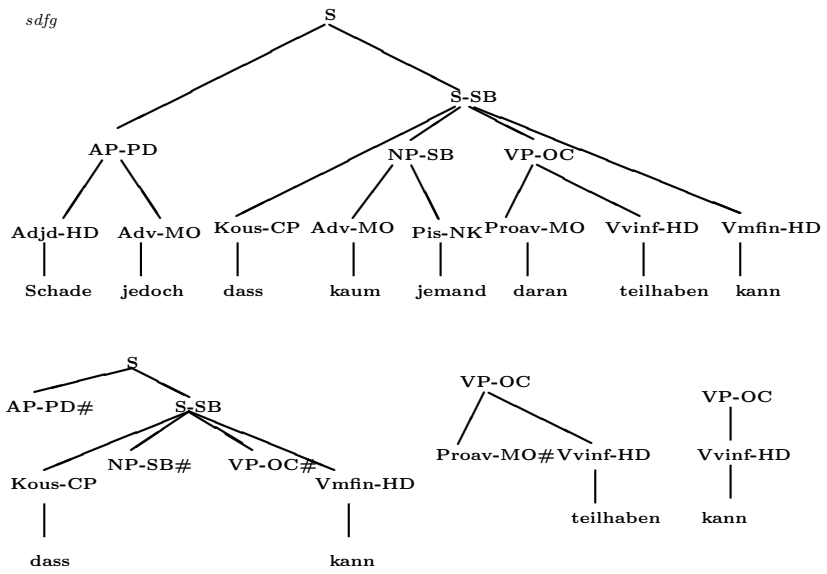

Figure 1.1 The Negra-parse tree of the sentence "Schade jedoch, daß kaum jemand daran teilhaben kann." (**Unfortunate however that almost nobody participate can*) and some of the extracted SLTG-trees. # denotes the substitution marker.

4. SLTG FROM TREEBANKS

We will now decribe how the general method is used for extracting an SLTG. 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 trees 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., 2000. We are using these transformed Negra-trees in our experiments.

We suggested that tree-decomposition should be guided by a head–driven decomposition principle and that we assumed that a treebank comes with a kind of a head principle. In the Negra-treebank, head

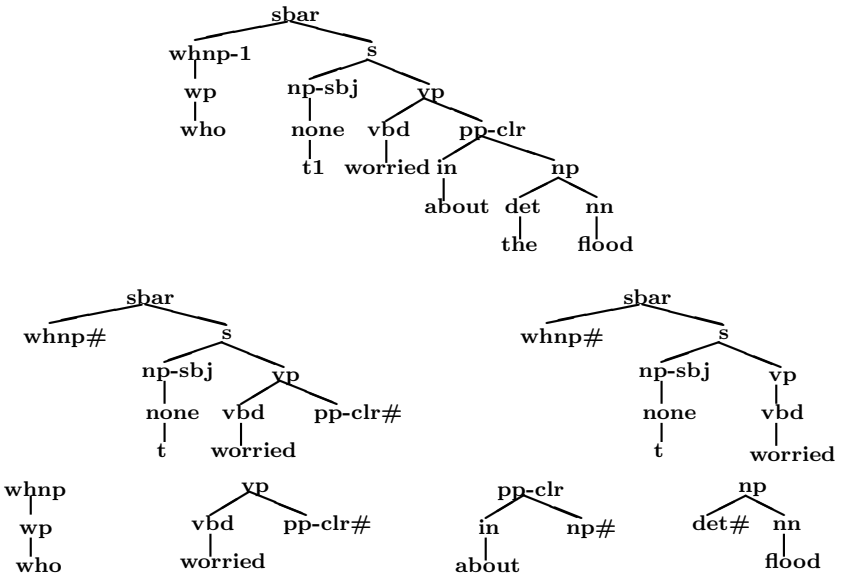

Figure 1.2 The Penn-parse tree of the sentence "who worried about the flood " and some of the extracted SLTG-trees. Here we used the word form as lexical anchor and POS for all other terminal elements. # denotes the substitution marker.

and modifier elements are explicitly tagged. For example, each head is marked by the suffix HD or NK which allows straightforward definitions of the decomposition principles. In case of the Penn-treebank (see figure 1.2), the head relation has been determined manually and stored in a head-percolation table.[2] 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 as described in section 3. 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 applying the standard definitions as known from the TAG literature. 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 node labels, like :lfoot, :rfoot, and :wfoot). Note that the identification of possible auxiliary trees is strongly corpus-driven. Using special foot node 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 node label as the substitution label, which means that the extracted grammar only allows for substitution. At this point we must 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 MO (see figure 1.1). However, since the parse trees are flat, we cannot simply factor out recursion without changing the topological structure of the parse trees. For this 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* (as described in 3.).[3]

4.1 EXPERIMENTS

We will briefly report on first results of our method using the Negra-treebank (4270 sentences) and the sections 02, 03, 04 from the Penn-treebank (the first 4270 sentences). In both cases we extracted three different versions of SLTGs (note that no normalization of the tree-banks was 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 obtained for the different sorts of lexical anchors. The following two tables summarize the results:

Extracted number of trees for the Negra-treebank.

# trees	**anchor=word**	**anchor=pos**	**anchor=:term**
elem. trees	26553	10384	6515
leftaux trees	184	60	40
rightaux trees	54	35	25
wrapping trees	39	36	29

Extracted number of trees for the Penn-treebank.

# trees	**anchor=word**	**anchor=pos**	**anchor=:term**
elem. trees	31944	11979	8132
leftaux trees	701	403	293
rightaux trees	649	246	153
wrapping trees	386	306	249

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) is 6.18 sec. This is promising, since the parser has not yet been optimized.

We also tried initial blind tests, but it turned out that the current size of the considered 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 parsing result, very often due to 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 slower by factor of about 1/5 (due to the restricted annotation scheme).[4] A better strategy seems to be the use of words for lexical anchors only 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).

5. SLTG FROM HPSG

The same approach[5] has been applied to 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, Copestake et al., 2000, 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

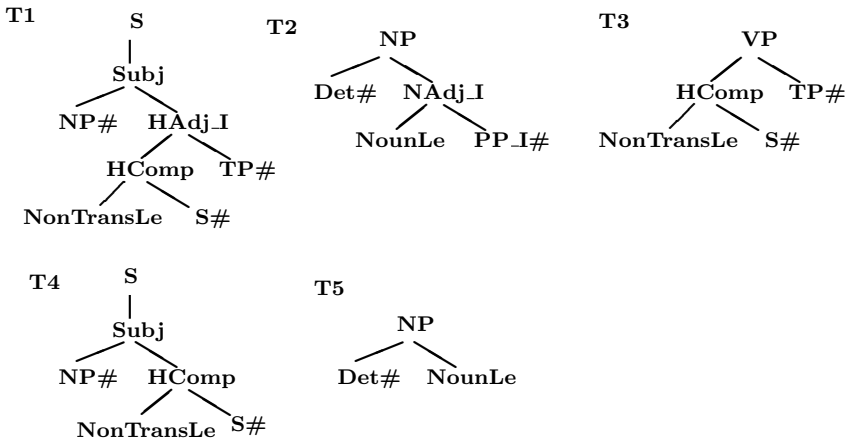

Figure 1.3 Some trees extracted from *I guess we need to figure out a day, within the next two months.* The symbols S, NP, VP, TP, Det, PP_I have been determined by means of specialization (see text). # denotes the substitution marker. Note that the lexical type of a lexical element is used as lexical anchor.

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., 1997 (see Oepen and Flickinger, 1998 for more detailed discussion of the grammar's coverage and of the issues related to such measurement).

Training of an SLTG starts by first parsing each sentence s_i of the training corpus with the source HPSG system.[6] 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 1.3).

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 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 section 3.). 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 this case, $HeadAdjunct$ may 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 1.2

Experiments. We trained the HPSG-system on a corpus of 2000 English sentences from Verbmobil dialogs (with an average length of 11.5 words per sentence). The size of the extracted SLTG before applying the additional operations (copy of trees of the head-chain and unattachment) is 1922 elementary trees. Copying of trees of the head chain yields 3828 trees, considering only unattachment as additional operation gives 2427 elementary trees. Applying both operations gives an SLTG with a total of 4195 trees.

Using the extracted SLTG-grammar we ran initial performance tests on the training corpus. The average *run-time* of the SLTG-parser (i.e., without off-line expansion, but including morphological and lexical preprocessing) 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 of the corresponding feature structure of the found SLTG-parse tree by unifying lexical information and the HPSG principles of the source grammar) is improved by a factor of 12 compared to parsing with the original highly tuned HPSG parser at the time of our study. Figure 1.4 shows the number of readings found by the SLTG parser. 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).

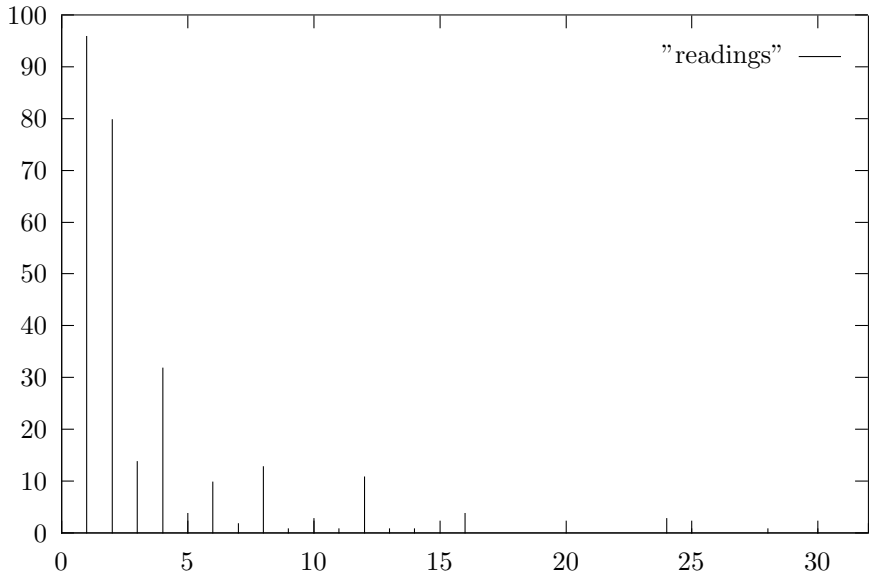

Figure 1.4 In almost all cases the SLTG-parser computes less than 16 readings per sentence.

6. FUTURE STEPS: TOWARDS MERGING SLTG

In the near future, we wish to investigate methods for merging SLTGs extracted from treebanks and competence grammar in order to 1) improve the coverage of treebank grammars on unseen data, and to 2) 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 treebank grammars can achieve an accuracy of about 87% (see Charniak, 1997). We wish to explore whether integrating knowledge from a competence grammar can improve the accuracy. We believe that SLTGs are well suited since they capture distributional and hierarchical information better than stochastic CFGs. The major obstacle for merging the current grammars is the different nature of syntactic constituent levels. For example, the Penn-treebank modifier structure is flat compared to that of an HPSG-based SLTG. Recently, Xia, 1999; Chen and Vijay-Shanker, 2000, have shown how the Penn-treebank can be fully bracketed in order to factor out the recursive structures for elementary trees. This is actually done by inserting additional non-terminal nodes into the treebank trees on the basis 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 similar trees found in the treebank-SLTG such that the HPSG-SLTG serves as "building plans". Xia and Palmer, 2000, have very recently presented a method for comparing the hand-crafted lexicalized XTAG English grammar (cf. Doran et al., 1994) with one extracted from the Penn-treebank. The core idea is to find out how many trees in one grammar match trees in the other. We will explore its use for our proposed HPSG-based approach.

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 making use of similar tree matching methods to those as described above. Now, it would be possible to adapt an HPSG-SLTG to a new domain following a minimally supervised learning approach by automatically creating new trees using an evolutionary strategy and to measure the fitness of such trees through the application of HPSG principles. Implementation of this method has already been started and we hope to be able to report on first results soon.

Acknowledgments

The research carried out for this paper was supported by a research grant from the German Bundesministerium für Bildung, Wissenschaft, Forschung und Technologie (BMBF) to the DFKI project WHITEBOARD ("Multilevel Annotation for Dynamic Free Text Processing"), FKZ 01 IW 002. I would like to thank Thorsten Brants for providing me with the Negra Corpus, as well as Tilman Becker, Ann Copestake, Anoop Sarkar, and Ivan Sag for many helpful comments. Many thanks also to Dan Flickinger for the very fruitful cooperation. I also would like to thank the anonymous reviewers for their valuable 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. Of course, we are assuming that the Penn-treebank developers made use of a kind of head-principle, e.g., by assuming that the head of an NP is the main noun or that the head of a VP is the main verb, see also Charniak, 1997. Chen and Vijay-Shanker, 2000 describe a sophisticated method carefully considering the determination of a node's status as a complement or adjunct, which I plan to integrate in the near future.

3. I am aware of the fact that this might not be the best strategy. Johnson, 1999 presents an interesting tree transformation procedure which maps a flat modifier construction of trees in the Penn-treebank into Chomsky adjunction representations, and shows that such a representation actually gives better predictive information on unseen data than the original representation. Johnson also remarks that argument PPs are not systematically distinguished from adjunct PPs in the Penn-treebank, and states that "... reliably determining whether a particular PP is an argument or an adjunct is extremely difficult, even for trained linguists.", Johnson, 1999, page 624. It is clear that it is not plausible to check a large enough treebank manually, in order to completely "extract" the implicitly made linguistic assumptions. Thus, only if a treebank comes with an explicit declaration of the used general linguistic principles, will it be possible to predict and interpret the influence of the effect of different tree transformation and extraction methods on the quality of a statistically driven parser.

4. 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.

5. 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.

6. Kasper et al., 1995, describe a method for compiling an HPSG source grammar to an LTAG. The basic idea here is to construct elementary trees starting from lexical elements from the bottom-up applying the HPSG-principles. In some sense our approach performs in the opposite direction by decomposing an HPSG-parse tree from the top downwards applying the HPSG principles for guiding the decomposition operation. Furthermore, since our approach is data driven, we are able to capture statistical information into an HPSG framework. Note further that once we have built up an HPSG-SLTG parse tree we are able to reconstruct the whole feature structure by applying the lexical information and all HPSG principles including semantic information of the source grammar. Hence, the resulting feature structure of an input sentence is correct and compatible with respect to the HPSG source grammar.

References

Bod, R. (1993). Using an annotated language corpus as a virtual stochastic grammar. In *Proceedings of AAAI'93*, Washington, D.C.

Bod, R. (2000). Extracting stochastic grammars from treebanks. This volume.

Brants, T., Skut, W., and H.Uszkoreit (2000). Syntactic annotation of a german newspaper corpus. This volume.

Charniak, E. (1997). Statistical parsing with a context-free grammar and word statistics. In *AAAI-97*, Providence, Rhode Island.

Chen, J. and Vijay-Shanker, K. (2000). Automated extraction of tags from the penn treebank. In *6th International Workshop on Parsing Technologies (IWPT'2000)*, Trento, Italy.

Chiang, D. (2000). Statistical parsing with an automatically–extracted tree adjoining grammar. In *38th ACL*, Honk Kong.

Copestake, A., Flickinger, D., and Sag, I. (1997). Minimal recursive semantics: An introduction. Technical report, CSLI, Stanford University.

Copestake, A., Flickinger, D., and Sag, I. (2000). Linguistic grammars online (lingo) project. http://hpsg.stanford.edu/hpsg/lingo.html.

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.

Johnson, M. (1999). Pcfg models of linguistic tree representations. *Journal of Computational Linguistics*, 24(4):613–632.

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

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. (1998). Automatic extraction of stochastic lexicalized tree grammars from treebanks. In *4th workshop on tree-adjoining grammars and related frameworks*, Philadelphia, PA, USA.

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 ten years after. *Journal of Computer Speech and Language*, 12:411–435.

Pollard, C. J. and Sag, I. A. (1994). *Head-Driven Phrase Structure Grammar*. Studies in Contemporary Linguistics. University of Chicago Press, Chicago, London.

Samuelsson, C. (1994). Grammar specialization through entropy thresholds. In *Proceedings of the 32nd Annual Meeting of the Association forComputational 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 worder 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.

van Genabith, J., Sadler, L., and Way, A. (2000). Deriving an lfg grammar from treebanks. This volume.

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.

Xia, F. and Palmer, M. (2000). Comparing and integrating tree adjoining grammars. In *Proceedings of the 5th TAG+ workshop*, Paris, France.