Automatic Extraction of Stochastic Lexicalized Tree Grammars from Treebanks

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

Abstract

We present a method for the extraction of stochastic lexicalized tree grammars (S-LTG) of different complexities from existing treebanks, 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 S-LTG 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 method for the extraction of stochastic lexicalized tree grammars (S-LTG) of different complexities from existing treebanks, 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. The use of S-LTGs is motivated for two reasons. First, it is assumed that S-LTG 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. Existing treebanks are used because they allow a corpus-based analysis of grammars of realistic size. Processing of different S-LTG is performed by a stochastic version of the two-phase Early-based parsing strategy introduced in (Schabes and Joshi, 1991).

This abstract describes work in progress. So far, we have concentrated on the automatic extraction of S-LTGs of different kinds (actually S-LTSG, S-LTIG, and S-LTAG). This phase is completed and we will report on first experiments using the Penn-Treebank (Marcus et al., 1993) and Negra, a treebank for German (Skut et al., 1997). A first version of the two-phase parser is implemented, and we have started first tests concerning its performance.

2 Grammar extraction

Given a treebank, 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 CFG 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 sec. 5) and that it renders possible a comparison 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 sec. 6).

However, in this paper we will focus on the use of existing treebanks using the Penn-Treebank (Marcus et al., 1993) and Negra, a treebank for German (Skut et al., 1997). First, it is assumed that the treebank comes with a notion of lexical and phrasal head, i.e., with a kind of *head principle* (see also (Charniak, 1997)). In the Negra treebank, head elements are explicitly tagged. For the Penn treebank, the head relation has been determined manually. 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-headed 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. 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.

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 S-LTIG, 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 substitution.

3 Two-phase parsing of S-LTG

The resulting S-LTG 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 sec. 4). 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 S-LTIG 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.

4 First 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 S-LTG (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:

Negra	**words**	**pos**	**:term**
elem. trees:	26553	10384	6515
leftaux trees	184	60	40
rightaux trees	54	35	25
wrapping trees	39	36	29

Penn	**words**	**pos**	**:term**
elem. tree:	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 S-LTIG 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, however ignoring every parse which took longer than 30 sec. (about 20 %). 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 (around 10 %). 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).[1] 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.

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. 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 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 generealized 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

The work described here is certainly in its early phase. The next future steps (partly already started) will be: (1) measuring the coverage of an extracted S-LTG, (2) incremental grammar induction, (3) combination of a competence grammar and a treebank. I already applied the same learning strategy on derivation trees obtained from a large HPSG-based English grammar in order to speed up parsing of HPSG (extending the work described in (Neumann, 1994)). Now I am exploring methods for merging such an "HPSG-based" S-LTG with one extracted from a treebank. The same will also be explored wrt. a competence-based LTAG, like the one which comes with the XTAG system (Doran et al., 1994).

7 Acknowledgment

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, FKZ ITW 9704. I would like to thank Tilman Becker for many fruitful discussions.

References

R. Bod. 1995. *Enriching Linguistics with Statistics: Performance Models of Natural Language.* Ph.D.

[1] 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.

thesis, University of Amsterdam. ILLC Dissertation Series 1995-14.

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

C. Doran, D. Egedi, B. Hockey, B. Srinivas, and M. Zeidel. 1994. Xtag system - a wide coverage grammar for english. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING)*, Kyoto, Japan.

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

G. Neumann. 1994. Application of explanation based learning for efficient processing of constraint-based grammars. In *Proceedings of the Tenth IEEE Conference on Artifical Intelligence for Applications*, pages 208–215, San Antonio, Texas, March.

C. Samuelsson. 1994. Grammar specialization through entropy thresholds. In *Proceedings of the 32nd Annual Meeting of the Association forComputational Linguistics*, pages 188–195.

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

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

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

Y. Schabes. 1992. Stochastic lexicalized tree-adjoining grammars. In *Proceedings of the 14th International Conference on Computational Linguistics (COLING)*, pages 426–432, Nantes.

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

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

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