HPSG–DOP: data–oriented parsing with HPSG

Günter Neumann

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH
Stuhlsatzenhausweg 3
66123 Saarbrücken, Germany
neumann@dfki.de

HPSG–DOP: data–oriented parsing with HPSG

1 Introduction

In recent years several approaches have been proposed to improve the performance of an HPSG system, e.g., [vN97], [MYTT98], [KKNVS95], [KKCM99], or [TNMT00]. 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. 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 [BC93, Sam94, RC96] or lexicalized TAGs [Sri97].

In this paper we apply the idea of data–oriented approaches for achieving domain-adaptation as an alternative approach to improve efficient processing of HPSG grammars. The basic idea of HPSG–DOP is to parse all sentences of a representative training corpus using an HPSG grammar and parser in order to automatically acquire from the parsing results a *stochastic lexicalized tree grammar* (SLTG) such that each resulting parse tree is recursively decomposed into a set of subtrees. The decomposition operation is guided by the head feature principle of HPSG. 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, cf. 2). For each extracted tree a frequency counter is used to estimate the probability of a tree, after all parse trees have been processed.

Processing of an SLTG is performed by a two level parser. 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. 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).
- Since the SLTG learning phase is driven by the HPSG principles of the source grammar, the whole extraction process is simple and transparent.
- 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. [Neu97]), as well as to efficient interleaved parsing and generation (cf. [Neu98]).

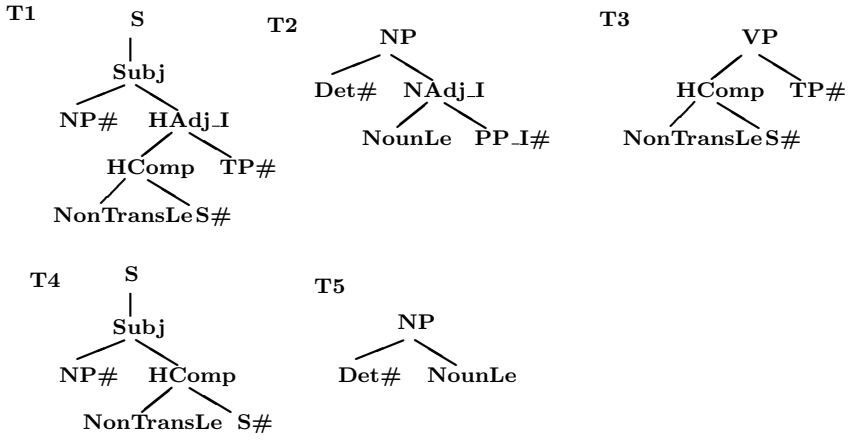

Figure 1: Some trees extracted from *I guess we need to figure out a day, within the next two months.* The symbols S, NP, VP, TP, Det, PP_I have been determined by means of specialization (see text). # denotes the substitution marker.

The approach has been implemented on top of the LKB (Linguistic Knowledge Building) system, a unification–based grammar development, parsing and generation platform currently being developed at CSLI [Cop99], and has been tested using the large HPSG–based English Resource Grammar being developed as part of the LinGO (Linguistic Grammars Online) project also at CSLI (see [FCS00, Fli00] for a detailed discussion of the grammar's architecture and coverage).

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 recursively decomposed into a set of subtrees such that each non-head subtree is cut off, and the cutting point is marked for substitution. 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 the LinGO 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. Note that these two additional operations applied on the training material can be considered as linguistically justified tree induction operations. Firstly, each subtree of the head-chain is copied and the copied tree is processed individually by the decomposition operation. 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 done by raising the head daughter of m into the position of m. Note, that unattaching a modifier actually means that a *copy* of a tree is made and that this copied tree is *destructively* changed by unattaching the modifier as described above.

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.

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., $\sum_{t_i:root(t_i)=\alpha} p(t_i) = 1$). Since we have only the substitution operation, the probability of a derivation will be the product of the probabilities of the trees which are involved in the derivation, and the probability of a parse tree is the sum of the probabilities of all derivations.

3 Parsing of SLTG

In HPSG–DOP, parsing of an SLTG is performed by a chart-based agenda-driven bottom-up parser (following [SJ91]). 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.

4 Experiments

We have conducted initial experiments using a VerbMobil corpus of 2000 utterances from dialogs about scheduling of meetings, with an average sentence length of 7.8 words. We

run the first 1000 sentences for training of an SLTG as described above using the LKB parser system. The LKB parser was able to find a valid analysis for 831 sentences which corresponds to a coverage of 83.1%. In each case the first reading is selected for the training which corresponds to a corpus size of 831 parse trees.

Processing of this HPSG–based tree–bank yields an SLTG containing 3818 different lexicalized trees anchored with a lexical type. There were 180 different lexical types computed which gives an average of about 21 trees per lexical anchor.

From the original 2000 sentences, the next 1000 sentences were used for testing the coverage of the extracted SLTG. We first run the LKB parser to measure its coverage on this test corpus, which resulted in a coverage of 803 analyzed sentences. We used this subset of parsed sentences to test the coverage of the SLTG. For the computation of the first reading of each sentence, the LKB parser needed 1.40 seconds on average .[1] Actually this shows that the LKB HPSG parser (including rule filters and unification quick checks) is already quite fast, at least for this sort of corpus.

Then, the SLTG parser was called in off–line unification mode such that for each sentence it stopped after the first complete SLTG-tree could successfully be "expanded" by unifying the feature constraints of the HPSG source grammar. Thus seen, we consider a sentence analysis as valid, if it is consistent with respect to the HPSG constraints (including all lexical constraints, of course). Following this way, 704 sentences were recognized which corresponds to a coverage of 87.67%. On average, the SLTG parser needed 0.48 seconds per sentence.

We then run the SLTG-parser without the off-line unification step, but in exhaustive mode (computing an average of 58 alternative parse trees for each sentence). For each sentence, we checked whether the set of possible analysis also contained the first reading of the HPSG parser considering only the parse tree, i.e., no constraints other than those conflated into the category labels (cf. 2). This was the case for 551 sentences which means that we obtained an exact derivation accuracy (exact match of the resulting parse trees) of 68.61%.

Concerning efficiency, the current speed up is by a factor of 2.92. This is already a quite promising result. On the one hand side, the parser of the LKB system is a highly tuned one which already uses a kind of data–driven approach through the quick check mechanism which results in an performance improvement of 75%. On the other hand side, our SLTG parser is still not optimized. For example, we assume that the number of trees selected for an input sentence can be reduced substantially using more specific lexical types (which would have to be defined as part of the grammar), making use of pruning strategies based on [RC96] in order to filter out unplausible intermediate tree structures or to induce more word-based statistical information, e.g., exploring strategies to statistically "connect" word forms or word stems with certain tree structures, or to make use of anchored–based n–grams.

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. [KKNVS95]), 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.

[1] Processing was done on a SUN Ultra Enterprise 450, 4 CPUs (400MHz) and 4GB RAM.

Our approach is also related to approaches of grammar specialization based on Explanation-based Learning (EBL), cf. [Sam94, SJ95, RC96]. Rayner and Carter 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. It should be not difficult to integrate these or similar 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.

References

[BC93] T. Briscoe and J. Carroll. Generalized probabilistic lr parsing of natural language (corpora) with unification-based grammars. *Computational linguistics*, 19(1):25–59, 1993.

[Cop99] A. Copestake. The (new) lkb system. CSLI, Stanford University: http://www-csli.stanford.edu/~acc/doc5-2.pdf, 1999.

[FCS00] D. Flickinger, A. Copestake, and I. Sag. Hpsg analysis of english. In *W. Wahlster (ed.) Verbmobil: Foundations of Speech-to-Speech Translation*, pages 254–263, Springer–Verlag Berlin Heidelberg New York, 2000.

[Fli00] D. Flickinger. On building a more efficient grammar by exploiting types. *Journal of Natural Language Engineering*, 6(1):15–28, 2000.

[KKCM99] Bernd Kiefer, Hans-Ulrich Krieger, John Carroll, and Rob Malouf. A bag of useful techniques for efficient and robust parsing. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics, ACL-99*, pages 473–480, 1999.

[KKNVS95] 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.

[MYTT98] 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.

[Neu97] 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.

[Neu98] G. Neumann. Interleaving natural language parsing and generation through uniform processing. *Artifical Intelligence*, 99:121–163, 1998.

[RC96] 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.

[Sam94] C. Samuelsson. Grammar specialization through entropy thresholds. In *Proceedings of the 32nd Annual Meeting of the Association forComputational Linguistics*, pages 188–195, 1994.

[SJ91] 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.

[SJ95] 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.

[Sri97] B. Srinivas. *Complexity of Lexical Restrictions and Its Relevance to Partial Parsing*. PhD thesis, University of Pennsylvania, 1997. IRCS Report 97–10.

[TNMT00] K. Torisawa, N. Nishida, Y. Miyao, and J. Tsujii. An hpsg parser with cfg filtering. *Journal of Natural Language Engineering*, 6(1):63–80, 2000.

[vN97] G. van Noord. An efficient implementation of the head-corner parser. *Computational Linguistics*, 23:425–456, 1997.