Explanation-Based Learning of Data-Oriented Parsing

K. Sima'an
Research Institute for Language and Speech,
Utrecht University,
Trans 10, 3512 JK Utrecht, The Netherlands
khalil.simaan@let.ruu.nl

Abstract

This paper presents a new view of Explanation-Based Learning (EBL) of natural language parsing. Rather than employing EBL for specializing parsers by inferring new ones, this paper suggests employing EBL for learning how to reduce ambiguity only partially.We exemplify this by presenting a new EBL method that learns parsers that avoid spurious overgeneration, and we show how the same method can be used for reducing the sizes of stochastic grammars learned from tree-banks, e.g. (Bod, 1995, Charniak, 1996, Sekine and Grishman, 1995).

The present method consists of an EBL algorithm for learning partial-parsers, and a parsing algorithm which combines partial-parsers with existing "full-parsers". The learned partial-parsers, implementable as Cascades of Finite State Transducers (CFSTs), recognize and combine constituents efficiently, prohibiting spurious overgeneration. The parsing algorithm combines a learned partial-parser with a given full-parser such that the role of the full-parser is limited to combining the constituents, recognized by the partial-parser, and to recognizing unrecognized portions of the input sentence. Besides the reduction of the parse-space prior to disambiguation, the present method provides a way for refining existing disambiguation models that learn stochastic grammars from tree-banks e.g. (Bod, 1995, Charniak, 1996, Sekine and Grishman, 1995).

We exhibit encouraging empirical results using a pilot implementation: parse-space is reduced substantially with minimal loss of coverage. The speedup gain for disambiguation models is exemplified by experiments with the DOP model (Bod, 1995).

1 Introduction

Current work on natural language parsing is in large part directed towards eliminating overgeneration of grammars by employing stochastic models for disambiguation (e.g. (Bod, 1995, Sekine and Grishman, 1995, Charniak, 1996)). For many applications (e.g. Speech Understanding), probabilistic evaluation of the full parse-space using such models is NP-hard (Sima'an, 1996b), and even when it is deterministic polynomial-time, then grammar size is prohibitive. Therefore, it is necessary to develop methods that, on the one hand, reduce the space of analyses, as much as possible prior to disambiguation, and on the other hand, reduce the sizes of grammars used for disambiguation. This paper presents a method aimed at these two forms of reduction of time and space costs.

In recent work on speeding up parsing, effort is directed towards *specializing* broad-coverage grammar by EBL (e.g. (Rayner, 1988, Samuelsson, 1994, Rayner and Carter, 1996, Srinivas and Joshi, 1995)). Grammar-specialization, in these works, amounts to replacing a given parser by a fresh efficient parser learned from the tree-bank. The learned parser trades coverage for efficiency. Inspired by these works, we present a new method based on EBL for learning efficient parsers. Rather than specializing a given full-parser by inferring a new one, the present method learns a partial-parser and combines it with the full-parser in a way that reduces ambiguity. The combination is a serial construction in which the partial-parser is employed first for recognizing and combining constituents. The partial-parser is learned such that it parses only those portions of the sentence that are "safe" to parse, i.e. at the points where there is clear bias in the tree-

Khalil Sima'an (1997) **Explanation-Based Learning of Data-Oriented Parsing.** In T.M. Ellison (ed.) *CoNLL97: Computational Natural Language Learning,* ACL pp 107-116.

bank. These constituents are then passed through, together with unrecognized portions of the input, to the full-parser, that completes the space only where necessary.

For disambiguation models such as (Bod, 1995, Sekine and Grishman, 1995, Charniak, 1996), the present method refines the cutting criteria which these models employ for inferring stochastic grammars. This refinement results in the inference of smaller, yet no less powerful, statistical grammars.

2 Terminology

A Context-Free Grammar (CFG) derivation is a sequence of one or more rewriting steps, starting with the start non-terminal of the grammar, employing the grammar productions. A *subderivation* is a subsequence of a derivation. A string of symbols which results from a CFG-derivation (of zero or more rewriting steps) is called a *sentential-form.* A *partial-tree (also subtree)* of a given tree t is a tree-structure which is the result of a subderivation of a derivation represented by t. A partial-tree which has as its root the start non-terminal is called a *sentential partial-tree.* The string obtained from the ordered sequence of leaf nodes of a partial-tree is called the *frontier* of the partial-tree; the leaf nodes are called the *frontier nodes* of the partial-tree. A Context-Free rule (CF-rule) $R{=}A \rightarrow A_1 \ldots A_n$ is said to *appear* in a tree t if there is a node in t, labeled A, and that node has n children labeled with (maintaining order from left to right) $A_1 \cdots A_n$. The CFG (V_N, V_T, S, $\mathcal{R}$) is called the *CFG underlying a given tree-bank* iff $\mathcal{R}$ is the set $\{R \mid$ rule R appears in a tree in the tree-bank$\}$ (and the start non-terminal S, non-terminal set V_N, terminal set V_T are exactly those of the tree-bank). A parser based on the CFG-underlying a tree-bank is called the Tree-bank parser (denoted **T-parser**).

3 Explanation-Based Learning

EBL (Mitchel et al., 1986, DeJong and Mooney, 1986, van Harmelen and Bundy, 1988) is the name of a unifying framework for methods that learn from previously explained examples of a certain concept. EBL assumes a domain theory (or background theory) which provides explanations to and enables the definition of concepts. In existing literature, the main goal of EBL is much faster recognition of concepts than the domain-theory does; EBL learns "shortcuts" in computation (called macro-operators or "chunks"), or directives for changing the thread of computation. EBL stores the learned chunks in the form of partial-explanations to previously seen input instances, in order to apply them in the future to "similar" input instances (in EBL, also "similarity" is assumed provided by the domain-theory).

The specification of EBL consists of four preconditions and one postcondition. The preconditions are: *1) A domain theory: A description language for the domain at hand together with rules and facts about the domain. 2) A target concept: A formal description, over the alfabet of the domain-theory, of the to-be-learned relation. 3) An Operationality criterion: A requirement on the form of the target concept. And 4) training examples: A history which makes explicit the explanations given by the domain-theory to examples that occurred in the past; the explanations consist of instances of the target concept.* The postcondition is: *Find a generalization of the instances of the target concept given in the training-examples that satisfies the operationality criterion.*

Past experience in Machine Learning cast doubts on the feasibility of improving performance by using EBL (Minton, 1990). Minton explains that EBL does not guarantee better performance, since the cost of applying the learned knowledge might outweigh the gain. Minton discusses a formula for computing the utility of knowledge during learning. Generally speaking, this formulae is neither part of EBL nor part of the domain-theory; it is an extension to the EBL scheme by e.g. statistical inference over large sets of training examples.

4 Learning partial-parsers

We assume a tree-bank representing a certain domain of application. The tree-bank forms the training-examples of our EBL-based method, and the linguistic annotation employed for annotating the sentences represents the domain-theory. For the sake of presentation we delay the discussion of detail of the algorithm and concentrate on a simplified version of it. The simplest instances of the target-concept of our algorithm are called *probably always subsentential-forms* (PA-SSFs).

Subsentential-form *A subsentential-form (SSF) is a sequence of grammar-symbols which forms the frontier of a partial-tree.*

Probably always SSF *An SSF* $ssf = N_1 \cdots N_m$ *is called* Probably Always SSF (PA-SSF) with respect to the tree-bank *if the frequency of occurrence of* $N_1 \cdots N_m$ *in the tree-bank as SSF (denoted* $fc(N_1 \cdots N_m)$*) is equal to the total frequency of its occurrence in the tree-bank (denoted* $f(N_1 \cdots N_m)$*).*

The concept PA-SSF formalizes the intuitive concept "probably always constituent". In reality, as discussed below, this concept is refined to become context-sensitive and less rigid; it becomes "probably *almost* always constituent *in some local-context*". Moreover, to avoid sparse-data problems we exclude the words of the language from the SSFs which we consider; the SSFs may consist of both part-of-speech tags (PoSTags) as well as phrase-symbols.

Associated subtree *A partial-tree which has the sequence* ssf *as its frontier is called a* subtree associated *with* ssf. *The set of subtrees associated with* ssf, *with respect to a tree-bank, consists of all partial-trees of the tree-bank trees, which are subtrees associated with* ssf.

The learning algorithm

The goal of the algorithm is to learn the set of PA-SSFs that represents the tree-bank trees in the *fastest and least ambiguous* way possible. The predicate "least ambiguous" is instantiated in two ways: 1) the learned (almost) PA-SSFs imply brackets which are most probably useful. And 2) the set of subtrees associated with a learned PA-SSF is assumed complete, i.e. no more structures are necessary for future sentences containing that PA-SSF. The second goal "fastest" is implemented by selecting the PA-SSFs that reduce the tree-bank trees in the fastest way. To achieve this we employ an operationality criterion which measures the utility of a PA-SSF. A measure of how much a single PA-SSF contributes to reducing a sentential-form is the *Reduction Factor*, and the "expected utility" of a PA-SSF is estimated as the *Global Reduction Factor*:

Reduction Factor *The* Reduction Factor *(RF) of a given SSF* ssf *is* $RF(ssf) = L(ssf) - 1$, *where* $L(ssf)$ *is the number of symbols which constitute* ssf.

Global RF *The* global reduction factor *of a given PA-SSF* ssf *with respect to the tree-bank is defined as* $GRF(ssf) = fc(ssf) \times RF(ssf)$, *where* $fc(ssf)$ *is the frequency of* ssf *as a constituent. In case* ssf *is an SSF that is* not *a PA-SSF then* GRF(ssf) $= -\infty$.

The specification of the learning algorithm is in figure 2. The algorithm learns PA-SSFs by an iterative procedure which "eats" up the tree-bank trees from their leaves upwards. Beginning with the tree-bank at hand, after each iteration, the procedure outputs: the set of learned PA-SSFs and a new tree-bank obtained by reducing all subtrees associated with a learned PA-SSF in all trees of the tree-bank at hand. In the next iteration, the same procedure is applied to the tree-bank output by this iteration. The procedure stops when there is nothing to learn anymore i.e. either there are no PA-SSFs to learn, or all tree-bank trees are fully reduced to their roots.

Competitor SSF *Let N be a node in tree t and let ssf be the partial-tree with N as root. The frontier of a partial-tree with root node which is a descendent or ancestor of N in t is called a* competitor *of ssf.*

Operationality criterion *At each iteration of the algorithm, for each sentential partial-tree t in the tree-bank, for each SSF* ssf *in t,* ssf *is learned iff* ssf *is PA-SSF and* $GRF(ssf) \geq GRF(x)$, *for all x which is a competitor of* ssf *in t.*

Consider again the specification in figure 2. Let the algorithm be at a certain iteration i and let each node in each partial-tree of the current tree-bank ($\mathcal{TB}_i$) have a unique address. Also define the global reduction factor of an address N of a node, $GRF(N)$, to be equal to *GRF(ssf)*, where ssf is the SSF on the frontier of the partial-tree under the node with address N. The operationality criterion is implemented in the specification at step (2.), where nodes are marked. The learned PA-SSFs are those SSFs which form the frontiers of partial-trees of which the root is a node which was marked at some iteration.

Detail of learning algorithm

Now we present further detail of the algorithm. The term "PA-SSF" is redefined as follows:

> *ssf* is called PA-SSF if it fulfills $\frac{fc(ssf)}{f(ssf)} \geq \theta$, where $0 < \theta \leq 1$ is a threshold.

This definition of PA-SSF makes the target-concept of our EBL method become "with probability more than θ a constituent". The algorithm employs this definition as follows. A threshold is set on the values of θ, where θ is allowed to change during learning (the default value of this threshold is $\theta = 1.0$ unless stated otherwise). Suppose the threshold on θ is 0.75. The algorithm starts in the first iteration with learning PA-SSFs of $\theta = 1.0$. Each time there are no more PA-SSFs to learn, under the current value of θ, it reduces θ by a fixed amount (e.g. 0.05) until θ becomes equal to the threshold (0.75). Then the algorithm stops learning.

We also employ a threshold (τ) on the minimum frequency of SSFs; an SSF must be frequent enough in order to qualify for the PA-SSF test. Currently this threshold is set at the maximum of a fixed integer (e.g. 10) and a percentage of the number of trees

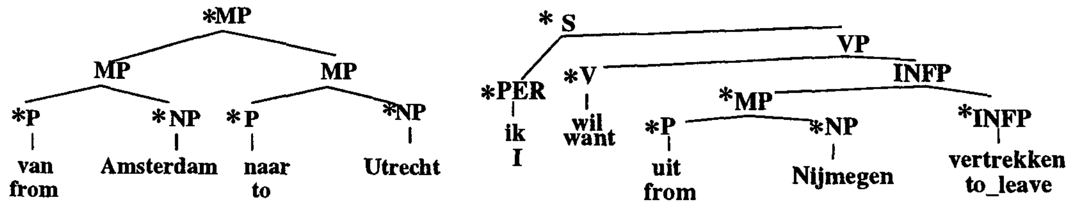

Figure 1: Two trees marked by the learning algorithm

```
/* Let $N$ denote a unique address of a node of a tree $t$. Also let $\mathcal{TB}_i$ denote the */
/* tree-bank obtained after $i$ iterations, where $\mathcal{TB}_0$ denotes the given tree-bank. */
/* Frontier_Of($N$) denotes the frontier (i.e. an SSF) of the partial-tree under $N$. */
/* Descendent(Nch,Np) denotes the predicate: the node addressed $Nch$ is a */
/* descendent of the node addressed $Np$. */

1. $i := 0$;
Repeat
2. $\forall t \in \mathcal{TB}_i$, $\forall$ node address $N$ in $t$: $N$ is marked iff
       Frontier_Of(N) is PA-SSF in $\mathcal{TB}_i$ and
       $\forall Nx \neq N$ in $t$: ( Descendent(Nx,N) or Descendent(N,Nx) ) $\rightarrow$ (GRF(N) $>$ GRF(Nx));
3. $i := i + 1$;
4. $\mathcal{TB}_i := (\mathcal{TB}_{i-1}$ after reducing all partial-trees under marked nodes);
until (($\mathcal{TB}_i == \emptyset$) or ($\mathcal{TB}_i == \mathcal{TB}_{i-1}$));
```

Figure 2: The Learning Algorithm

in the tree-bank (e.g. 0.3%). However, a more principled way to set the threshold is by letting it be a function of the distribution of SSFs in the tree-bank.

The algorithm also employs a definition of PA-SSF conditioned on local context, rather than fully context-free:

> A sequence of symbols is called PA-SSF in context C iff the ratio between its frequency as SSF in context C and its total frequency in context C is $\geq \theta$.

The local context that is employed consists of four fields: two grammar symbols to the left of and two to the right of an SSF. Since after the first round of the learning algorithm the training material consists of sentential partial-trees, this kind of local context may consist of PoSTags as well as *phrasal symbols*. The algorithm can use this local context in order to enhance learning and parsing. In the current implementation, however, we employ this local context only during learning and in a quite simplistic manner.

Since currently local-context is not employed during parsing, the learning algorithm is tuned to prefer as general local-contexts as possible. The learning algorithm assumes in the first place that all four fields of the local-context of an SSF are wild-cards. In case the SSF is not a PA-SSF in that context, then the algorithm *retreats* and assumes any three of the four fields to be wild-cards. In case an SSF is not a PA-SSF under three or more wild-cards of local-context then it is not learned, i.e. two or less wild-card local-contexts do not contribute to learning. Future implementations, however, shall have to take this local-context more seriously both in learning and in parsing.

Example: *In figure 1, two example trees are shown. The asterisks in the figure denote the borders of subtrees associated with PA-SSFs learned from the tree-bank. The sequences of symbols marked with an asterisk at the frontier of a subtree, which has a marked root, form the learned PA-SSFs. In the tree at the left-hand side of figure1, there is only one PA-SSF that reduces the tree: (p np p np), which corresponds to "from Amsterdam to Utrecht". In the right-hand side tree of figure 1, there are two PA-SSFs, (p np) and (per v mp infp).*

In the first iteration, the learning algorithm reduced the left tree totally and reduced the right tree only at the constituent "from Nijmegen". In the second iteration, the leftovers of the right-tree, a sentential partial-tree with frontier "ik/I wil/want mp vertrekken/to_leave", is reduced fully. If there are other partial-trees which are left over in the tree-

bank, after these two iterations, then the algorithm will attempt reducing them in subsequent iterations. If there are no more PA-SSFs to learn, the algorithm stops (possibly leaving some partial-trees not fully reduced).

The parsing algorithm

A Tree-Substitution Grammar (TSG) is a CFG with rules which are partial-trees called elementary-trees. Let the set of subtrees associated with the PA-SSFs, which the learning algorithm outputs, be the set of elementary-trees of a TSG; the TSG has the same start-symbol, terminal and non-terminal symbols as the CFG underlying the tree-bank. This TSG is employed as a partial-parser (other implementations are discussed below).

The new parsing algorithm combines the partial-parser with a given full-parser. It has two stages: firstly it employs the partial-parser for parsing the input sentence bottom-up, resulting in a space of partial-parses combined from subtrees associated with PA-SSFs. In the second phase it employs a given full-parser to complete these partial parses into full parses. Crucially, the second phase of the algorithm takes advantage of the construction of the learning algorithm. It makes two assumptions concerning the space of partial parses which the partial-parser constructed:

- If a sequence of symbols is recognized by the partial-parser then it is highly probable that all its subtrees are present in the chart (as these are either associated subtrees or combinations of associated subtrees). Thus it is not necessary to attempt reparsing portions of the sentence which were recognized as by the partial-parser.
- In the default case, $\theta = 1.0$, a PA-SSF implies "sure" constituent-borders; therefore, brackets placed by the full-parser are not allowed to cross the borders of a PA-SSF. In case two PA-SSFs cross each other, a highly unlikely case, then both PA-SSFs are removed from the partial-parser's output.

Thus, the task of the full-parser is limited to parsing *totally uncovered portions and combining them with the partial-trees provided by the partial-parser in ways that do not cross recognized PA-SSFs with* $\theta = 1.0$. In this paper we employ the CFG underlying the tree-bank (i.e. T-parser) as the full-parser.

Implementation of parsing algorithm

The current pilot implementation of the partial-parser does not take local context of PA-SSFs into consideration. The partial-parser is implemented as a parser for TSGs (Sima'an, 1996a), based on an extension to the CYK algorithm (Younger, 1967)). However, the partial-parser can be implemented as a Cascade of Finite State Transducers (CFSTs). A Finite State Transducer (FST) is learned at each iteration of the learning algorithm; the FST's language is the set of PA-SSFs learned at that iteration, and the output of the FST on recognition of a PA-SSF is the set of subtrees associated with that PA-SSF.

5 Existing related methods

EBL was introduced to NLP by Rayner (Rayner, 1988); Rayner employs EBL for specializing broad-coverage grammars to specific domains. In (Rayner and Samuelsson, 1994, Rayner and Carter, 1996) grammar specialization is conducted by chunking the trees of a tree-bank according to "chunking criteria" which are manually specified e.g. chunks correspond to trees with roots which correspond to full utterances, NPs, PPs or non-recursive NPs. Samuelsson (Samuelsson, 1994) is the first to depart from manual specification of chunking criteria in NLP; the chunking of the tree-bank trees employs the information theoretic measure of entropy. Samuelsson measures the entropy of a grammar non-terminal as the measure of how hard it is to decide on the choice of the next rule application given that non-terminal. Then he marks the nodes with the largest entropy as cutting nodes using an iterative algorithm. In (Srinivas and Joshi, 1995) the specific structure of the Lexicalized Tree-Adjoining Grammar (LTAG) derivations is exploited to result in an EBL method specific for LTAG. This differs from the other efforts in that the generalization which they employ is not limited only to *goal-regression* but allows generalizing the structure of explanations. Their method learns from the LTAG derivations of the training-examples all sequences of PoSTags and reduces those to regular-expressions by generalizing on sequences of adjunctions with a Kleene-star; the generalized LTAG-derivations are stored indexed by the PoSTag sequences.

Relation to Samuelsson's EBL: The present method is similar to Samuelsson's in that it learns "cutting criteria" from the data. Our method differs from Samuelsson's in that the cutting criteria are computed from an opposite direction. Samuelsson's maximum entropy is aimed at maximizing coverage, and his approach is derivational since the entropy is computed on steps of derivations starting from the start non-terminal. The target concept of our method is a PA-SSF not a non-terminal (i.e. "probably always constituent" vs. "constituent" resp.). Our method assumes a reductive approach and results in a partial-parser rather than a specialized parser.

Relation to LTAG's EBL: The concept of PA-SSF employed by our method is a generalization of the sentential PoSTag sequences employed in (Srinivas and Joshi, 1995). Our method can be easily extended to accommodate LTAG generalizations of derivations and of PA-SSFs; to this end it is necessary to have a tree-bank annotated with LTAG derivations. The subtrees associated with learned PA-SSFs are then generalized partial derivations of LTAG.

6 Application to DOP

This section relates the present EBL method to existing models of disambiguation that project stochastic grammars from tree-banks, e.g. (Bod, 1995, Charniak, 1996, Sekine and Grishman, 1995). To this end, we firstly relate these models to EBL, and then show that our new EBL method refines these models.

We are concerned only with models that project the same grammatical description as that employed for annotation of the tree-bank. Among these models, the Data Oriented Parsing (DOP) model (Scha, 1990, Bod, 1995) takes the most radical point of view. DOP projects all partial-trees from a tree-bank and employs them as a stochastic grammar called a Stochastic Tree-Substitution Grammar (STSG). Other models in the same category are presented in (Charniak, 1996, Sekine and Grishman, 1995). Charniak (Charniak, 1996) employs the tree-bank for projecting Stochastic CFGs (SCFGs). And (Sekine and Grishman, 1995) presents a constrained DOP-like model which projects STSGs; cutting the tree-bank trees takes place only at nodes labeled either with S or with NP. In this section we concentrate on DOP since it constitutes a generalization of the other two efforts.

In (Bod, 1995), the specification of DOP is as follows. A DOP model has four parameters:

1. sentence-analyses, i.e. syntactically labeled phrase structure trees given in a tree-bank,
2. sub-analyses, i.e. partial-trees,
3. combination-operations, i.e. substitution, and
4. combination-probabilities.

The rest of the definition of the DOP model concerns how to infer probabilities of partial-trees from the tree-bank, and how to compute probabilities of combinations of partial-trees. The instantiation of DOP as realized in (Bod, 1995) is an STSG, which has the set of *all* partial-trees of the tree-bank trees as elementary-trees. We shall not give further details of DOP since this is out of the scope of this paper.

Let us rewrite Bod's specification using the terminology of EBL. Firstly, the so called domain-theory consists of the annotation convention as well as the annotation intuitions used for the annotation of the tree-bank. The tree-bank contains sentences and their tree structures: the trees constitute "explanations" (proofs) given by the domain-theory to the fact that the sequences of words on their frontiers are sentences. The target-concept of DOP is the concept of a constituent, represented by non-terminals of the tree-bank trees. The sub-analysis used by DOP are simply partial-trees, which form instances of the target-concept. These partial-trees are obtained by using a simple operationality criterion, which states that any partial-tree obtained from a tree-bank tree is acceptable (in the experiments mentioned in (Bod, 1995), Bod limits the depth of partial-trees, Charniak (Charniak, 1996) limits the partial-trees to CFG rules, and in (Sekine and Grishman, 1995) only a subset of the non-terminals are allowed to supply partial-trees). The combination-operation of DOP is inherent to the assumption that the theory (phrase structure grammar) employs that operation. The fourth parameter of DOP, i.e. the inference and the definitions of probabilities of combinations of partial-trees, extends the EBL scheme. This extension enables DOP, and the other models mentioned above, to apply statistical analysis over large sets of trees in order to facilitate disambiguation. The interesting part of viewing these models in EBL terminology is the fact that these models do not aim at speedup, but rather at the memory-based behavior of EBL.

The new EBL method can be used in order to define the operationality criterion for DOP as follows.

- Apply the algorithm in figure 2 to the given tree-bank. The result is the same tree-bank except that now there are marking on nodes which delimit the subtrees associated with the learned PA-SSFs.
- Mark also all nodes which are *not* internal to any subtree associated with a learned PA-SSF. And mark all PosTag nodes in all tree-bank trees.

If learning was successful, then only some of the nodes of the tree-bank trees are marked now. The operationality criterion for DOP is then:

A partial-tree is projected iff its root and *the nodes on its frontier are marked, i.e. cutting the trees for DOP is not allowed at unmarked nodes.* Crucially, this way of cutting allows the projection of partial-trees which are *combinations* of subtrees associated with learned PA-SSFs.

The main remaining question on this refinement of DOP concerns the probabilities of the partial-trees projected from the tree-bank. In DOP, the probability of a partial-tree with a root labeled N is defined as the ratio between its frequency and the total frequency of all partial-trees that have N as their root-label. Since the space of partial-trees is smaller in the refinement, the probabilities will be different than in the original DOP. We conjecture that due to reducing the number of parameters of the model, sparse-data effects should be reduced (future work shall address this issue).

7 Empirical results

The present method was developed within a Dutch national project on a *dialogue system concerning public-transportation information* (called OVIS) (http://grid.let.rug.nl:4321/). Within the project, a vast amount of dialogues were collected, and the user's utterances were syntactically and semantically annotated (Scha et al., 1996). For experimentation we employ a tree-bank of the first 5000 syntactically annotated utterances. Here we only report experiments on parsing transcribed utterances[1].

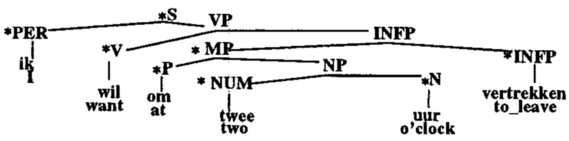

Figure 3: Another tree from OVIS

The annotation of the OVIS tree-bank is exemplified by the trees in figures 1 and 3. Due to the fact that OVIS contains answers to questions within a dialogue system, the sentences are often short but surprisingly variable in structure; many of these sentences contain repetitions, corrections and strange constructions (usually rendered ungrammatical by linguistic theories). Below we report on two sets of experiments. The first set observes the learning curves of the present EBL method by combining the learned partial-parsers with a T-parser (i.e. CFG). And the second set studies the refinement of DOP using the present EBL method. All timing experiments were conducted on SGI Indigo with 640 MB RAM.

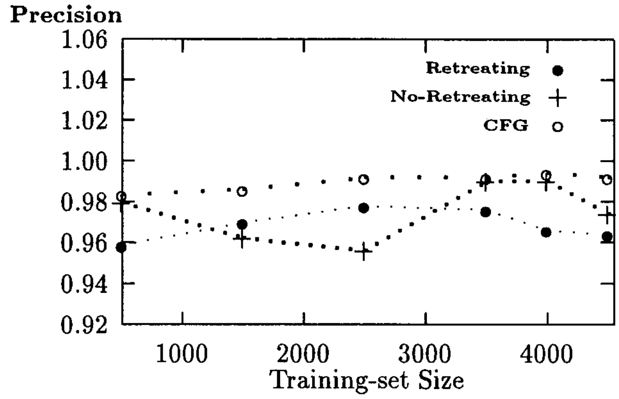

Figure 4: Learning curves for parser-precision

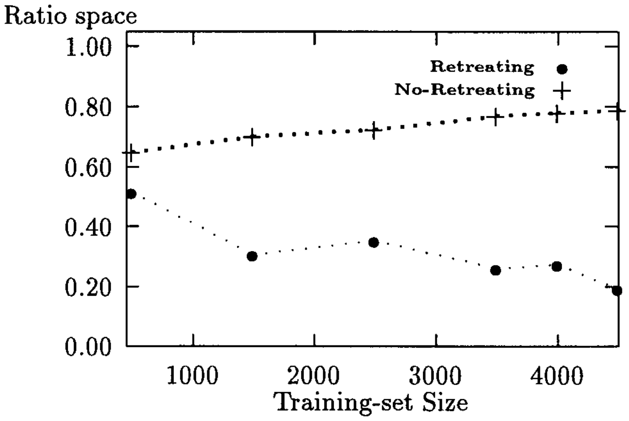

Figure 5: Learning curves: active-nodes $\frac{EBL+T\_Parser}{T\_Parser}$

OVIS experiments with T-parser

The experiments concern both coverage as well as size of parse-space. We employ the T-parser underlying the tree-bank (CFG) as a full-parser. In table 1 we list the results of ten independent experiments, each obtained by a random split of 4500 training-set and 500 test-set. Since the domain contains many (easy for parsing) one word utterances (e.g. "yes" or "no"), we exclude one word utterances from the results. On average, the ten test-sets contained 337.2 (of 500) utterances longer than one word. Table 1 shows the results on utterances longer than one word, with mean length of 5.57 words per utterance. For training the EBL learning algorithm we set a threshold on the frequency of SSFs: 0.3% of the size of the training-set (i.e. 14). To avoid problems of unknown words, we allowed the words of the test-set to be included with *all* postags with

[1] The present method was applied together with DOP for parsing word-graphs in a speech recognition-task and resulted in, compared to DOP, on average speedup of 10 times with virtually no loss of accuracy. Average speedup for word-graphs containing more than 40 states exceeds 20 times.

Parser	Right parse in chart	Any parse in chart	Precision	Active nodes
T_parser	97.78% (1.1%)	99.62% (0.3%)	98.15% (1.2%)	135.16 (248.93)
Par+T_parser	93.23% (1.1%)	99.11% (0.5%)	94.06% (1.5%)	31.17 (81.45)

Table 1: Means and STDs of ten experiments (OVIS): Par denotes Partial-Parser

which they appear in the whole tree-bank (for both parsers).

Table 1 shows the statistical means and (in brackets) the standard deviations of the ten experiments (always for sentences longer than 1 word). **Right parse** (also structural consistency) denotes the percentage of test sentences for which the parser's chart contains the right parse (i.e. test-set parse). **Any parse** (also coverage) denotes the percentage of test sentences for which the parser's chart contained a parse. **Precision** denotes the ratio (Right parse/Any parse), which expresses the precision of the parser as a parse-space generator. And **active nodes** denotes the mean number of active items in a CYK parser implementation; active items are those items that participate in a full parse of the sentence.

On average the partial-parser reduces the space by 4.33 times on all sentence lengths. The reduction of space reaches a mean of 7 times on sentences longer than 6. The degradation in precision (4%) is due to several reasons. Firstly, the fact that the partial-parser is currently implemented as a context-free recognizer clearly contributes to this degradation. Secondly, after analyzing the test-results of one experiment, we found out that about half of the errors are due to deeper structures assigned by the Partial-Parser rather than really wrong structures; typically those were compound NPs which received shallow annotations in the tree-bank. Thirdly, part of the errors is due to tree-bank annotation mistakes. And finally, there is a remaining part of errors which is due to the assumptions of the EBL method; these are harder to solve than the previous three.

In figure 4 and 5 we show the learning curves of the present method for six sizes of training-sets; five of the six training-sets were obtained randomly from a set of 4500 trees, and the sixth consisted of the whole set. For these experiments we employed the same set of 500 test-trees randomly chosen (all length sentences). The experiments were repeated twice: once allowing "retreating" on local-context (as explained earlier), and once not allowing that, during the learning phase (the two versions are denoted "Retreating" and "No-Retreating" respectively). The learning curves of the Retreating partial-parser, show that from a certain point on there is some deterioration of precision but further gain of space-reduction. The situation is different with the No-Retreating version. The explanation for the loss of precision is that when the training-set is smaller, less PA-SSFs are learned, which implies a larger role for the T-Parser. This situation is magnified by the fact that the coverage of the T-Parser is lower on smaller training-sets. The deterioration of precision of the Retreating version compared to the No-Retreating version is due to the fact that the number of learned local-context PA-SSFs becomes much larger; this implies reduction of parse-space but also some loss of precision (since the partial-parser does not employ the local-context).

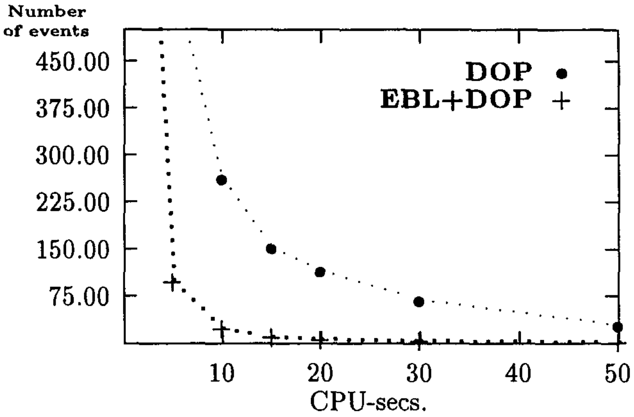

Figure 6: Number of sentences to CPU-time

OVIS experiments with DOP

To test the present method together with DOP we employed the same 10 random splits which we employed in the previous experiments. This time we did *not* include anything about unknown words in the test-sets (i.e. a sentence that includes an unknown word is not parsable). DOP and EBL+DOP were trained employing the following parameter setting for partial-trees (cf. (Sima'an, 1996a)): for each projected partial-tree, a maximum was set on its depth (D), number of substitution-sites (N) on its frontier, number of words (W) and number of consecutive words (C) on its frontier. The setting was D=4, N=2, W=7 and C=2. This reduces the number of elementary-trees which DOP projects drastically without loss of accuracy. Furthermore, the EBL algorithm was trained with a threshold on the frequency of SSFs equal to 14. The EBL method is used for both specializing the T-parser, which DOP

System	Coverage	Accuracy	CPU-secs. for sentence length		
			≥ 2	≥ 7	≥ 10
DOP	95.00% (1.4%)	93.50% (0.1%)	3.98 (11.29)	13.55 (22.84)	37.46 (41.35)
EBL+DOP	94.61% (1.4%)	91.72% (0.1%)	1.28 (2.31)	2.98 (4.46)	6.21 (8.67)
EBL0.75+DOP	94.96% (1.3%)	91.90% (1.4%)	1.33 (2.43)	3.18 (4.69)	6.85 (8.97)

System	number of trees	number of nodes in trees
DOP	27907 (1634)	141960 (938)
EBL+DOP	23660 (302)	117134 (1627)
ParPar	84.4 (4.8)	818.5 (10.26)
EBL0.75+DOP	23728 (138)	117750 (1551)
ParPar0.75	80.6 (3.0)	812 (10.40)

Table 2: Means and STDs of ten experiments (OVIS), ParPar denotes Partial-Parser

employs prior to disambiguation (Sima'an, 1996a), and for specifying the cut-nodes for DOP.

Another set of experiments on the same 10 random splits (denoted EBL0.75 in table 2) was conducted where the threshold on θ was set at 0.75, i.e. a sequence of grammar symbols was allowed to be learned if it was for at least 75% of the time an SSF. This was achieved by allowing the learning algorithm to change the threshold (θ) on the definition of PA-SSF; each time there are no more PA-SSFs to learn, θ was reduced by 0.03 and learning went on.

Table 2 lists the means and standard deviation for the 10 experiments for all sentences of length larger or equal to 2 words. The average (std of) percentage of the sentences that included an unknown word is 2.56% (0.93%). The measures which the table lists are coverage and accuracy, where **coverage** *is the percentage of sentences that received a parse*, and **accuracy** *is the percentage of parsable sentences that received exactly the same parse as the test-set counterpart. The* **precision** *of a method is equal to the multiplication of the two previous measures.*

On average, DOP "guesses" in 88.82% (i.e. precision) of the cases exactly the same test-set parse; with EBL this becomes 86.77%, i.e. a loss of 2.05%. The speedup is on average 3.1 times but, more importantly, the standard-deviation in processing time is less than a fifth. On longer sentences, the speedup exceeds 6 times. Figure 6 shows the accumulative frequency of sentences to CPU-time: for x secs., the figure shows the number of sentences that take at least x secs. in parsing. If a deadline of 5 secs. is set beforehand, DOP misses around the 600 cases (of 3372) while the EBL misses less than 100 cases. At 10 secs. the figures are 263 to 23, and at 20 secs. it's 116 to 6 cases respectively.

The version EBL0.75 shows similar learning capabilities to the EBL (i.e. EBL1.0) version. Its precision is slightly better with 87.26% and its coverage is virtually the same as DOP's. The EBL0.75 does not improve speedup though (actually it's slightly slower). The explanation to this behavior is simple: EBL0.75 does not seem to learn significantly many more rules than EBL1.0 and, during parsing, it gives up the assumption that PA-SSF borders are trustworthy. This way it takes less risk but then it slightly loses speed. Again we conjecture that EBL0.75 would provide more speedup if local-context would be used during partial-parsing. Table 2 shows also the sizes of grammars which DOP projects with and without EBL. The number of elementary-trees in the table for the Partial-Parser does not include the lexicon. The sizes of the statistical grammars of DOP with EBL is about 1.2 times smaller than DOP's. This is not the reduction which we hoped for, but it is quite evident that this is due to constraining the EBL mechanism; currently learning takes place only where local-context can be assumed of minor importance.

8 Conclusions and future work

We described a new view of EBL methods for parsing aiming directly at partial-disambiguation. Speedup is due to fast parsing that minimizes the parse-space prior to the, often, expensive probabilistic disambiguation. This view is exemplified by an EBL method, which 1) specializes parsers by inferring partial-parsers, and 2) refines existing stochastic models of disambiguation. From preliminary experiments with a pilot implementation we observe that the method has the potential of speeding-up parsing, especially for Speech Understanding where the input is a word-graph. Also we see that it is possible to minimize coverage loss when using EBL and still gain space-reduction and speed. However, these experiments have shown that it is hard to gain speed and space-reduction without employing local-context and without extensive training-sets.

Work on extending the parsing algorithm to accommodate local-context is being carried out and shall be ready very soon. Further exploration will proceed on several fronts. We intend to test this method on larger and harder tree-banks. An implementation as CSFTs will be studied and implemented. We shall also study other measures of utility, as mentioned earlier in this paper. And finally, we might extend this method as in (Srinivas and Joshi, 1995) or employ existing similarity-based measures for matching PA-SSFs, instead.

Acknowledgements: This work was supported by project 305 00 903, Priority Programme for Language and Speech Technology, Dutch Organization for Scientific Research (NWO). I thank Christer Samuelsson, Remko Scha and Remko Bonnema for comments on earlier versions.

References

Bod, R. (1995). *Enriching Linguistics with Statistics: Performance models of Natural Language.* Phd-thesis, ILLC-dissertation series 1995-14, University of Amsterdam.

Charniak, E. (1996). Tree-bank Grammars. In *Proceedings AAAI'96*, Portland, Oregon.

DeJong, G. and Mooney, R. (1986). Explanation-Based Generalization: A Alternative View. *Machine Learning 1:2*, pages 145–176.

Minton, S. (1990). Quantitative Results Concerning the Utility Problem of Explanation-Based Learning. *Artificial Intelligence*, 42:363–392.

Mitchel, T., Keller, R., and Kedar-Cabelli, S. (1986). Explanation-Based Generalization: A Unifying View. *Machine Learning 1:1.*

Rayner, M. (1988). Applying Explanation-Based Generalization to Natural Language Processing. In *Proceedings International Conference on Fifth Generation Computer Systems*, pages 1267–1274, Kyoto, Japan.

Rayner, M. and Carter, D. (1996). Fast Parsing using Pruning and Grammar Specilization. In *Proceedings ACL-96, Santa Cruz, CA.*

Rayner, M. and Samuelsson, C. (1994). Corpus-Based Grammar Specilization for Fast Analysis. In *Spoken Language Translator: First Year Report, Agnas et. al.* SRI technical report CRC-043, *http : //www.cam.sri.com.*

Samuelsson, C. (1994). Grammar Specialization Through Entropy Thresholds. In *Proceedings ACL'94*, Las Cruces, New Mexico.

Scha, R. (1990). Language Theory and Language Technology; Competence and Performance (in Dutch). In de Kort, Q. and Leerdam, G., editors, *Computertoepassingen in de Neerlandistiek*, Almere: LVVN-jaarboek.

Scha, R., Bonnema, R., Bod, R., and Sima'an, K. (1996). *Disambiguation and Interpretation of Wordgraphs using Data Oriented Parsing.* Probabilistic Natural Language Processing in the NWO priority Programme on Language and Speech Technology, Amsterdam.

Sekine, S. and Grishman, R. (1995). A Corpus-based Probabilistic Grammar with Only Two Non-terminals. In *Proceedings Fourth International Workshop on Parsing Technologies*, Prague, Czech Republic.

Sima'an, K. (1996a). An optimized algorithm for Data Oriented Parsing. In Mitkov, R. and Nicolov, N., editors, *Recent Advances in Natural Language Processing 1995*, volume 136 of *Current Issues in Linguistic Theory.* John Benjamins, Amsterdam.

Sima'an, K. (1996b). Computational Complexity of Probabilistic Disambiuation by means of Tree Grammars. In *Proceedings of COLING'96*, volume 2, pages 1175–1180, Copenhagen, Denmark.

Srinivas, B. and Joshi, A. (1995). Some Novel Applications of Explanation-Based Learning to Parsing Lexicalized Tree-Adjoining Grammars. In *Proceedings ACL-95.*

van Harmelen, F. and Bundy, A. (1988). Explanation-Based Generalization = Partial Evaluation (research note). *Artificial Intelligence 36*, pages 401–412.

Younger, D. (1967). Recognition and parsing of context-free languages in time n^3. *Inf. Control*, 10(2):189–208.