# Abductive Explanation-based Learning Improves Parsing Accuracy and Efficiency

**Oliver Streiter**
Language and Law, European Academy, Bolzano, Italy
ostreiter@eurac.edu

## Abstract

Natural language parsing has to be accurate and quick. Explanation-based Learning (EBL) is a technique to speed-up parsing. The accuracy however often declines with EBL. The paper shows that this accuracy loss is not due to the EBL framework as such, but to deductive parsing. Abductive EBL allows extending the deductive closure of the parser. We present a Chinese parser based on abduction. Experiments show improvements in accuracy and efficiency.[1]

## 1 Introduction

The difficulties of natural language parsing, in general, and of parsing Chinese, in particular, are due to local ambiguities of words and phrases. Extensive linguistic and non-linguistic knowledge is required for their resolution (Chang, 1994; Chen, 1996). Different parsing approaches provide different types of knowledge. Example-based parsing approaches offer rich syntagmatic contexts for disambiguation, richer than rule-based approaches do (Yuang et al., 1992). Statistical approaches to parsing acquire mainly paradigmatic knowledge and require larger corpora, c.f. (Carl and Langlais, 2003). Statistical approaches handle unseen events via smoothing. Rule-based approaches use abstract category labels.

[1]This research has been carried out within *Logos Gaias* project, which integrates NLP technologies into a Internet-based natural language learning platform (Streiter et al., 2003).

Example-based parsing generalizes examples during compilation time, e.g. (Bod and Kaplan, 1998), or performs a similarity-based fuzzy match during runtime (Zavrel and Daelemans, 1997). Both techniques may be computationally demanding, their effect on parsing however is quite different, c.f. (Streiter, 2002a).

Explanation-based learning (EBL) is a method to speed-up rule-based parsing via the caching of examples. EBL however trades speed for accuracy. For many systems, a small loss in accuracy is acceptable if an order of magnitude less computing time is required. Apart from speed, one generally recognizes that EBL acquires some kind of knowledge from texts. However, what is this knowledge like if it does not help with parsing? Couldn't a system improve by learning its own output? Can a system learn to parse Chinese by parsing Chinese? The paper sets out to tackle these questions in theory and practice.

### 1.1 Explanation-based Learning (EBL)

Explanation-based learning techniques transform a general problem solver (PS) into a specific and operational PS (Mitchel et al., 1986). The caching of the general PS's output accounts for this transformation. The PS generates, besides the output, a documentation of the reasoning steps involved (the explanation). This determines which output the system will cache.

The *utility problem* questions the claim of speeding-up applications (Minton, 1990): Retrieving cached solutions in addition to regular processing requires extra time. If retrieval is slow and cached solutions are rarely re-used, the cost-benefit ratio is negative.

The accuracy of the derived PS is generally below that of the general PS. This may be due to the EBL framework as such or the deductive base of the PS. Research in abductive EBL (A-EBL) seems to suggest the latter: A-EBL has the potential to acquire new knowledge (Dimopoulos and Kakas, 1996). The relation between knowledge and accuracy however is not a direct and logical one. The *U-shaped* language learning curves in children exemplifies the indirect relation (Marcus et al., 1992). Wrong regular word forms supplant correct irregular forms when rules are learned. We therefore cannot simply equate automatic knowledge acquisition and accuracy improvement, in particular for complex language tasks.

### 1.2 EBL and Natural Language Parsing

Previous research has applied EBL for the speed-up of large and slow grammars. Sentences are parsed. Then the parse trees are filtered and cached. Subsequent parsing uses the cached trees. A complex HPSG-grammar transforms into tree-structures with instantiated values (Neumann, 1994). One hash table lookup of POS-sequences replaces typed-feature unification. Experiments conducted in EBL-augmented parsing consistently report a speed-up of the parser and a drop in accuracy (Rayner and Samuelsson, 1994; Srinivas and Joshi, 1995).

A loss of information may explain the drop of accuracy. Contextual information, taken into account by the original parser, may be unavailable in the new operational format (Sima'an, 1997), especially if partial, context-dependent solutions are retrieved. In addition, the set of cached parse trees, judged to be "sure to cache", is necessarily biased (Streiter, 2002b). Most cached tree structures are short noun phrases. Parsing from biased examples will bias the parsing.
A further reason for the loss in accuracy are incorrect parses which leak into the cache. A stricter filter does not solve the problem. It increases the bias in the cache, reduces the size of the cache, and evokes the utility problem.

EBL actually can improve parsing accuracy (Streiter, 2002b) if the grammar does not derive the parses to be cached via deduction but via abduction. The deductive closure[2] which cannot increase with EBL from deductive parsing may increase with abductive parsing.

## 2 A Formal View on Parsing and Learning

We use the following notation throughout the paper: $\mu \rightarrow (y) = x$ (function $\mu$ applied to $y$ yields x), $\mu \mapsto (y) = x$ (relation $\mu$ applied to $y$ yields x). $< ... >$ and $\{...\}$ represent tuples and sets respectively. The $\#$ prefix denotes the cardinality of a collection, e.g. $\#\{o_1, o_2\} = 2$.

Uppercase variables stand for collections and lowercase variables for elements. Collections may contain the anonymous variable $\epsilon$ (the variable _ in PROLOG). Over-braces or under-braces should facilitate reading: $\underbrace{1+1}_{2} = \underbrace{2*1}_{2}$.

A theory $\mathcal{T}$ is $< \mathcal{A}, \mathcal{I}, \mathcal{R} >$ where $\mathcal{R}$ is a set of rules $r$. $\mathcal{A}$ and $\mathcal{I}$ are two disjoint sets of attributes $a$ and $i$ (e.g. $\mathcal{A} = \{"noun", "verb", ...\}; \mathcal{I} = \{"Bob", "hill", ...\}$). A rule is written as $r =< o, a >$ or $r \mapsto (o) = a$. A rule specifies the relation between an observable fact $o$ and an attribute $a$ assigned to it. $\mathcal{O}$ is the set of observable data with each $o \in \mathcal{O}$ being a tuple $o =< a, i >$.[3]

$\mathcal{C}$ is the set of data classified according to $\mathcal{T}$, with $c =< o, a >$. $o$, $i$ and $a$ may have an internal structure in the form of ordered or unordered collections of more elementary $o$, $i$ and $a$ respectively.

Transferring this notation to the description of parsing, $\mathcal{T}$ is a syntactic formalism and $\mathcal{R}$ a grammar. $\mathcal{A}$ is the union of syntax trees and morpho-syntactic tags. $\mathcal{O}$ is a corpus tagged with $\mathcal{A}$. $\mathcal{I}$ corresponds to a list of words, phrases or sentences (the surface strings). $\mathcal{C}$ is a treebank, a cache of parse trees, or a history of explanations.

$$c_{parse} =<< a_{pos}, i_{lexeme} >, a_{tree} > \quad (1)$$

### 2.1 Parsing: $\pi \mapsto (o) = \{c_{new}\}$

A **parser** defines a relation between $\mathcal{O}$ and $\mathcal{C}$ (c.f. 2). **Parsing** is a relation between $o$ and a subset of $\mathcal{C}$ (c.f. 3).

$$\pi \mapsto (\mathcal{O}) = \mathcal{C} \quad (2)$$

[2]The deductive closure of the set of axioms $\mathcal{X}$ is the set $\mathcal{S}$ which can be proved from it.

[3]The formalization follows (Dimopoulos and Kakas, 1996).

$$\pi \mapsto (o) = \{c_{new}\} \quad (3)$$

Simplifying, we can assume that $\pi$ is defined as the set of rules, i.e. $\pi =< \mathcal{O}, \mathcal{C} >= \mathcal{R}$. A specific parser $\pi$ is derived by the application of $\theta$ to the training material (e.g. $\mathcal{C}$): $\theta \mapsto (\mathcal{C}) = \pi$. The set of possible relations $\theta$ is $\Theta$. Elements of $\Theta$ are caching (no generalization), induction (hypothesis after data inspection) and abduction (hypothesis during classification). Equation (5) describes the cycle of grammar learning and grammar application.

$$\theta \mapsto (\mathcal{C}) = \pi \quad (4)$$

$$\underbrace{(\theta \mapsto (\mathcal{C}_{old}))}_{\pi} \mapsto (\mathcal{O}) = \mathcal{C}_{new} \quad (5)$$

#### 2.1.1 Memory-based Parsing

$\pi$ is based on *memory* if $(\theta \mapsto (c)) =< c, c >=< r, r >$. $\gamma$ in (6) is the trivial formalization of caching. Parsing proceeds via recalling $\rho$ defined in (7). The cycle of grammar learning and parsing $\rho \mapsto (\gamma)$ is defined in (8): The training material $c_w$ yields the parsing output $c_w$.[4]

$$\gamma \mapsto (< o_v, a_w >) = \underbrace{< o_v, a_w >}_{\rho} \quad (6)$$

$$\rho \mapsto (o_v) =< o_v, a_w > \quad (7)$$

$$\overbrace{\overbrace{(\gamma \mapsto (\overbrace{< o_v, a_w >}^{c_w})) \mapsto}^{\text{learning } \rho \text{ from } c} (o_v) =}^{\text{parsing } o_v} \underbrace{< o_v, a_w >}_{c_w} \quad (8)$$

#### 2.1.2 Deduction-based Parsing

Let $delete$ be a function which replaces one or more elements of a collection by a named variable or $\epsilon$. $\pi$ is a deductive inference if $r$ is obtained from an *induction* (a reduction of $o$ with the help of $delete$). The following expressions define induction $\iota$ (9), deduction $\delta$ (10) and the inductive-deductive cycle $\delta \mapsto (\iota)$ (11):

[4]We use subscripts to indicate the identity of variables. The same subscript of two variables implies the identity of both variables. Different subscripts imply nothing. The variables may be identical or not identical. In memory-based parsing, learning material and parsing output are identical.

$$\iota \mapsto (\underbrace{<< a_w, i_w >, a_v >}_{c_v}) = \underbrace{< \underbrace{delete \to (< a_w, i_w >)}_{< a_w, \epsilon >}, a_v >}_{r_{v_\epsilon} = \delta} \quad (9)$$

$$\delta \mapsto (\underbrace{< a_w, i_x >}_{o_w}) =<< a_w, i_x >, a_v > \quad (10)$$

$$\overbrace{(\iota \mapsto (\overbrace{<< a_w, i_w >, a_v >)}^{c_v} \mapsto (< a_w, i_x >)) =}^{\text{parsing } o_z} << a_w, i_x >, a_v > \quad (11)$$

#### 2.1.3 Abduction-based Parsing

Abduction, defined as $\overbrace{(\psi \mapsto (c))}^{\alpha} \mapsto (o)$ is a runtime generalization which is triggered by a concrete $o$ to be classified. We separate $\psi$ and $\alpha$ for presentation purpose only.[5] The relation $\sigma$ may express a similarity, a temporal or causal relation. (12) and the cycle of $\alpha \mapsto (\psi)$ (13) define abduction.

$$\psi \mapsto (c) = \sigma \mapsto (c) = r \quad (12)$$

$$\overbrace{\overbrace{(\psi \mapsto (<< a_x, i_x >, a_c >))}^{\text{learning } \alpha \text{ from } c} \mapsto \overbrace{(< a_y, i_y >)}^{o_y} =}^{\text{parsing } o_y} < \underbrace{< a_y, i_y >}_{o_y}, a_r > \quad (13)$$

Abduction subsumes reasoning by analogy. Abduction is an analogy, if $\sigma$ describes a similarity. Reasoning from rain to snow is a typical analogy. Reasoning from wet street to rain is an abductive reasoning. For a parsing approach based on analogy c.f. (Lepage, 1999).

[5]Abduction is a process of hypothesis generation. Deduction and abduction may work conjointly whenever deductive inferences encounter gaps. A deductive inference stops in front of a gap between the premises and a possible conclusion. Abduction creates a new hypothesis, which allows to bridge the gap and to continue the inference.

### 2.2 Learning: $\mathcal{C} \cup ((\theta \mapsto (\mathcal{C})) \mapsto (o))$

In this section, we formalize EBL. We mechanically substitute $\theta$ in the definition of EBL by $\gamma, \delta, \alpha$ to show their learning potentials.

A learning system changes internal states, which influence the performance. The internal states of $\pi$ are determined by $\mathcal{C}$ and $\Theta$. We assume that, for a given $\pi$, $\Theta$ remains identical before and after learning. Therefore, the comparison of $\mathcal{C}$ (before learning) with $\mathcal{C} \cup c_{new}$ (after learning) reveals the acquired knowledge.

We define EBL in (14). $(\theta \mapsto (\mathcal{C}))$ is the parser before learning. This parser applies to $o$ and yields $c_{new}$, formalized as $(\theta \mapsto (\mathcal{C})) \mapsto (o)$. The new parser is the application of $\Theta$ to the union of $\mathcal{C}$ and $c_{new}$.

$$\pi_{new} = \theta \mapsto (\mathcal{C} \cup \underbrace{(\underbrace{(\theta \mapsto (\mathcal{C}))}_{\pi_{old}} \mapsto (o))}_{\{c_{new}\}}) \tag{14}$$

From two otherwise identical parsers, the parser with $c =<< a_o, \epsilon >, a_c >$ not present in the other has a greater deductive closure. The cardinality of $<< a_o, \epsilon >, a_c > \in \mathcal{C}$ reflects an empirical knowledge. The empirical knowledge does not allow to conclude something new, but to resolve ambiguities in accordance with observed data, e.g. for a sub-language as shown in (Rayner and Samuelsson, 1994). Both learning techniques have the potential of improving the accuracy.

#### 2.2.1 Learning through Parsing

A substitution of $\Theta$ with $\gamma, \delta, \alpha$ reveals the transformation of $c_{old}$ to $c_{new}$. We start with caching and recalling (Equation 15).

$$\pi_{new} = \gamma \mapsto (\{c_i\} \cup \underbrace{\underbrace{(\gamma \mapsto (c_i))}_{\rho} \mapsto (o_i)}_{c_i}) \tag{15}$$

Parsing $o_i$ with the cache of $c_i$ yields $c_i$. The deductive closure is not enlarged. Quantitative relations with respect to $o$ change in $\mathcal{C}$. If $c_i$ is not cached twice, memory-based EBL is idempotent.[6]

[6] Idempotence is the property of an operation that results in the same state no matter how many times it is executed.

EBL with induction and deduction is shown in (16). Here the subscripts merit special attention: $o =< a_x, i_y >$ is parsed from $c =<< a_x, i_x >, a_v >$. This yields $c_{new} =<< a_x, i_y >, a_v >$. Integrating $c_{new}$ into C changes the empirical knowledge with respect to $a$ and $i$. If the empirical knowledge does not influence $\iota$, D-EBL is idempotent. The deductive closure does not increase as $<< a_x, \epsilon >, a_v > \in \mathcal{C}$.

$$\begin{aligned}\pi_{new} = \iota \mapsto (&\{< a_x, i_x >, a_v >\} \cup \\ &\underbrace{((\iota \mapsto (\underbrace{<< a_x, i_x >, a_v >}_{c})) \mapsto (< a_x, i_y >))}_{<< a_x, i_y >, a_v >})\end{aligned} \tag{16}$$

Abductive EBL (A-EBL) is shown in (17). A-EBL acquires empirical knowledge similarly to D-EBL. In addition, a new $<< a_y, \epsilon >, a_r >$ is acquired. This $c_{new}$ may differ from $c_{old}$ with respect to $a_y$ and/or $a_r$. In the experiments in A-EBL we reported below, $a_y \neq a_x$ and $a_r = a_c$ holds.

$$\begin{aligned}\pi_{new} = \psi \mapsto (&\{< a_x, i_x >, a_c >\} \cup \\ &\underbrace{(\underbrace{(\psi \mapsto (\underbrace{<< a_x, i_x >, a_c >}_{c}))}_{\text{learning } \alpha} \mapsto (\underbrace{< a_y, i_y >}_{o_y}))}_{<< a_y, i_y >, a_r >})\end{aligned} \tag{17}$$

#### 2.2.2 Recursive Rule Application

Parsing is a classification task in which $a \in \mathcal{A}$ is assigned to $o \in \mathcal{O}$. Differently from typical classification tasks in machine learning, natural language parsing requires an open set $\mathcal{A}$. This is obtained via the recursive application of $\mathcal{R}$, which unlike non-recursive styles of analysis (Srinivas and Joshi, 1999) yields $\mathcal{A}$ (syntax trees) of any complexity. Then $delete$ is applied to $\mathcal{A}$ so that $delete \mapsto (A)$ can be matched by further rules (c.f. 18). Without this reduction, recursive parsing could not go beyond memory-based parsing.

$$\begin{aligned} r_m =<< &< a_m, i_m >, \\ &< delete \mapsto (r_p \mapsto (o_p)), i_p >>, \\ &< a_n, < a_m, r_p \mapsto (o_p) >>> \end{aligned} \tag{18}$$

Figure 1: An explanation produced by OCTOPUS. At the top, the final parse obtained via deductive substitutions. Abductive term identification bridges gaps in the deduction (X $\sim$ Y). The marker '?' is a graphical shortcut for the set of lexemes $\{i\}$ in $c$.

```
              VP(uncond:Cbaa:不管|epist:Dbaa:是|Head:VA4:求學|compl:Tb:也好)
                                              |
                          ------------------------------------
                          |                                  |
              VP(epist:Dbaa:是|Head:VA4:求學)  VP(uncond:Cbaa:不管|Head:VA4:求學|compl:Tb:也好)
                          |                                  |
    -----------------VC2 ~ VA4--                  ------------------------------------
    |      |                   |                  |                                  |
 Dbaa:是 VA4:求學 VP(epist:Dbaa:?|Head:VC2:?)  VP(uncond:Cbaa:不管|Head:VA4:求學)  VP(Head:VA4:求學|compl:Tb:也好)
                                                  |                                  |
                                            --------------------------     ----- VA4 ~ Tb ---------------------
                                            |                |       |     |                       |      |
                              VP(uncond:Cbaa:?|Head:VA4:?) Cbaa:不管 VA4:求學 VP(Head:VA4:?|compl:VA4:?) VA4:求學 Tb:也好
```

The function $delete$ defines an induction and recursive parsing is thus a deduction. Combinations of memory-based and deduction-based parsing are deductions, combinations of abduction-based parsing with any another parsing are abductions.

*Macro Learning* is the common term for the combination of EBL with recursive deduction (Tadepalli, 1991). A macro $r_{macro}$ is a rule which yields the same result as a set of rules $\mathcal{R}'$ with $\#\mathcal{R}' \geq 2$ and $r_{macro} \notin \mathcal{R}'$ does. In terms of a grammar, such macros correspond to redundant phrases, i.e. phrases that are obtained by composing smaller phrases of $\mathcal{R}$. Macros represent shortcuts for the parser and, possibly, improved likelihood estimate of the composed structure compared to the estimates under independency assumption (Abney, 1996). When the usage of macros excludes certain types of analysis, e.g. by trying to find longest/best matches we can speak of pruning. This is the contribution of D-EBL for parsing.

## 3 Experiments in EBL

### 3.1 Experimental purpose and setup

The aim of the experiments is to verify whether new knowledge is acquired in A-EBL and D-EBL. Secondly, we want to test the influence of new knowledge on parsing accuracy and speed.

The general setup of the experiment is the following. We use a section of a treebank as seed-corpus ($\mathcal{C}_{seed}$). We train the seed-corpus to a corpus-based parser. Using a test-corpus we establish the parsing

Figure 2: The main parsing algorithm of OCTOPUS. The parser interleaves memory-based, deductive, and abductive parsing strategies in five steps: Recalling, non-recursive deduction, deduction via chunk substitution, first with lexemes, then without lexemes and finally abduction.

$\pi \rightarrow (< a, i >)$ {

**# 1 recalling from POS (a) and lexeme (i)**
RETURN $c$ IF ($c = \rho \rightarrow (< a, i >)$)

**# 2 deduction on the basis of POS (a)**
RETURN $c$ IF ($c = \delta(< a, \epsilon >)$)

**# 3 deductive, recursive parsing with POS and lexeme**
**# Substitutions are defined as in TAGs (Joshi, 2003)** IF
$((< a_{chunk}, i_{chunk} >, < a_{redu}, i_{redu} >) = best\_chunk \rightarrow (< a, i >))$ {

RETURN $substitution \rightarrow$ ( **# deduction**
$\pi \rightarrow (< a_{chunk}, i_{chunk}, >)$,
$\pi \rightarrow (< a_{redu}, i_{redu}) >)$}

**# 4a deductive recursive parsing with lexeme,**
**# 4b compared to abductive parsing**
IF $((< a_{chunk}, i_{chunk} >, < a_{redu}, i_{redu} >) = best\_chunk \rightarrow (< a, \epsilon >))$ {

RETURN $arg_{max} \rightarrow$(
$\alpha \rightarrow (< a, i >)$, **#abduction**
$substitution \rightarrow$ ( **#deduction**
$\pi \rightarrow (a_{chunk}, i_{chunk})$,
$\pi \rightarrow (a_{redu}, i_{redu})$))}

**# 5 abduction as robust parsing solution**
RETURN $\alpha \rightarrow (< a, i >)$ }

Figure 3: Abductive parsing with k-nn retrieval and adaptation of retrieved examples.

$\alpha \rightarrow (< a, i >) \{$

RETURN $adaptation \rightarrow (viterbi.align \rightarrow (k.nn.retrieval \rightarrow (< a, i >)))\}$

---

accuracy and speed of the parser ($evaluate(\pi \mapsto (O_{test}))$ =(recall,precision,f-score,time)). Then, we parse a large corpus ($\pi \mapsto (O) = \{c_{new}\}$). A filter criterion that works on the explanation applies. We train those trees which pass the filter to the parser ($\theta \mapsto (C_{seed} \cup \{c_{new}\}) = \pi_{new}$)). Then the parsing accuracy and speed is tested against the same training corpus ($evaluate(\pi_{new} \mapsto (O_{test}))$ =(recall,precision,f-score,time)).

Sections of the Chinese Sinica Treebank (Huang et al., 2000) are used as seed-treebank and gold standard for parsing evaluation. Seed-corpora range between 1.000 and 20.000 trees. We train them to the parser OCTOPUS (Streiter, 2002a). This parser integrates memory- deduction- and abduction-based parsing in a hierarchy of preferences, starting from **1** memory-based parsing, **2** non-recursive deductive parsing, **3** recursive deductive parsing and **5** finally abductive parsing (Fig. 2).

Learning the seed corpora ($\theta \mapsto (c_{1000} \cdots {}_{20.000})$) results in $\pi_{1000} \ldots \pi_{20.000}$. For each $\pi \in \{\pi_{1000} \ldots \pi_{20.000}\}$, a POS tagged corpus $\mathcal{O}$ with $\#\mathcal{O} = 200.000$ is parsed, producing the corpora $\mathcal{C}_{o_{1000}} \ldots \mathcal{C}_{o_{20.000}}$. The corpus used is a subset of the 5 Million word Sinica Corpus (Huang and Chen, 1992).

For every $o \in \mathcal{O}$ the parser produces one parse-tree $c =< o, a >$ and an *explanation*. The explanation has the form of a derivation tree in TAGs, c.f (Joshi, 2003). The deduction and abduction steps are visible in the explanation. Filters apply on the explanation and create sub-corpora that belong to one inference type.

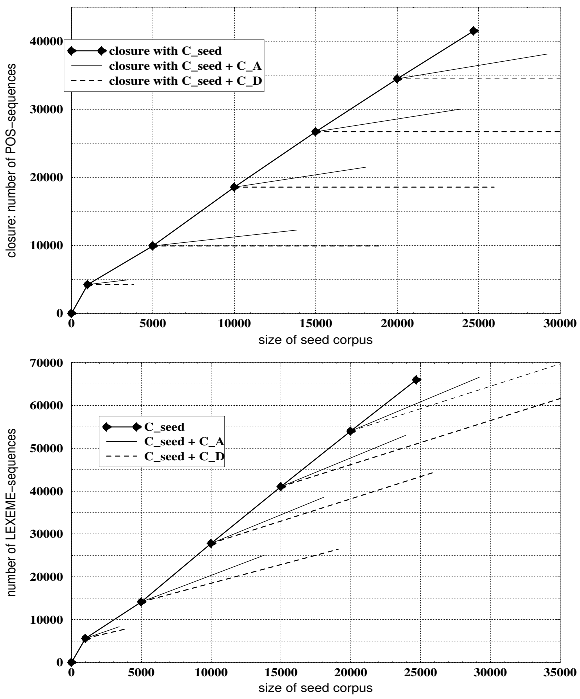
The first filter requires the explanation to contain only one non-recursive deduction, i.e. only parsing step **2**. As deductive parsing is attempted after memory-based parsing (**1**), $i_x \neq i_v$ holds.

A second filter extracts those structures, which are obtained by parsing step **4a** or **5** where only one POS-labels may be different in the last characters (e.g. $\sigma \mapsto ("Nab") = "Nac"$). The resulting corpora are $\mathcal{C}_{o_{1000_{deduct}}} \ldots \mathcal{C}_{o_{20.000_{deduct}}}$ and $\mathcal{C}_{o_{1000_{abduct}}} \ldots \mathcal{C}_{o_{20.000_{abduct}}}$.

### 3.2 The Acquired Knowledge

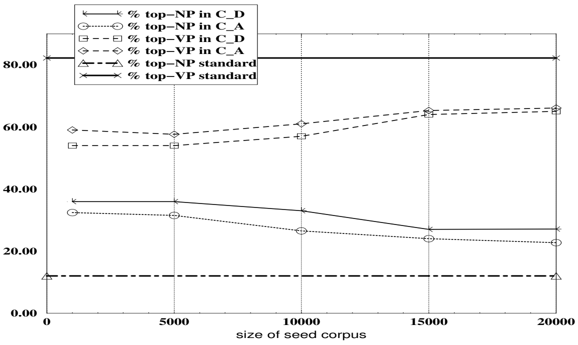
We want to know whether or not new knowledge has been acquired and what the nature of this acquired knowledge is. As parsing was not recursive, we can approach the closure by the types of POS-sequences from all trees and their subtrees in a corpus. We contrast this with to the types of lexeme-sequences. The data show that only A-EBL increases the closure. But even when looking at lexemes, i.e. empirical knowledge, the A-EBL acquires richer information than D-EBL does.

Figure 4: The number of types of POS-sequences as approximation of the closure with $C_{seed}$, A-EBL and D-EBL. Below the number of type of LEXEME-sequences.




The representatives of the cached parses is gauged by the percentage of top NPs and VPs (including Ss) as top-nodes. Fig 5 shows the bias of cached parses which is more pronounced with D-EBL than with A-EBL.

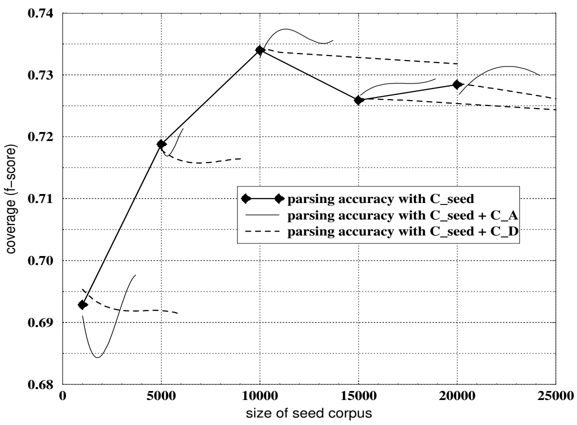
Figure 5: The proportion of top-NPs and top-VP(S) in abduced and deduced corpora.




### 3.3 Evaluating Parsing

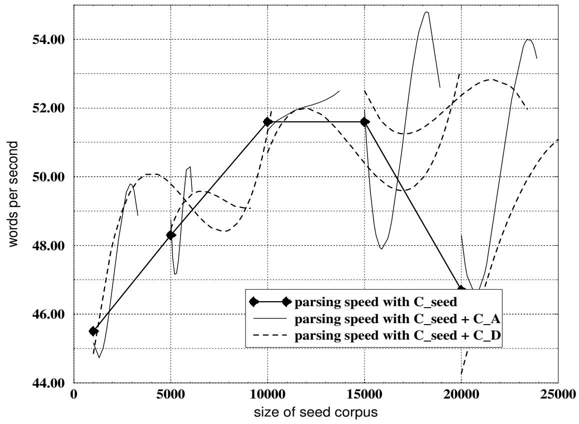
The experiments consist in evaluating the parsing accuracy and speed for each $\mathcal{C}_{seed} \cup \mathcal{C}_{o1000_{deduct}} \ldots \mathcal{C}_{seed} \cup \mathcal{C}_{o20.000_{abduct}}$.

Figure 6: The parsing accuracy with abductive EBL ($C_{seed} + C_A$) and deductive EBL ($C_{seed} + C_D$).




We test the parsing accuracy on 300 untrained and randomly selected sentences using the f-score on unlabeled dependency relations. Fig. 6 shows parsing accuracy depending on the size of the seed-corpus. The graphs show side branches where we introduce the EBL-derived training material. This allows comparing the effect of A-EBL, D-EBL and hand-coded trees (the baseline). Fig. 7 shows the parsing speed in words per second (Processor:1000 MHz, Memory:128 MB) for the same experiments. Rising lines indicate a speed-up in parsing. We have interpolated and smoothed the curves.

Figure 7: The parsing time with A-EBL ($C_{seed} + C_A$) and D-EBL ($C_{seed} + C_D$).




The experimental results confirm the drop in parsing accuracy with D-EBL. This fact is consistent across all experiments. With A-EBL, the parsing accuracy increases beyond the level of departure.

The data also show a speed-up in parsing. This speed-up is more pronounced and less data-hungry with A-EBL. Improving accuracy and efficiency are thus not mutually exclusive, at least for A-EBL.

## 4 Conclusions

Explanation-based Learning has been used to speed-up natural language parsing. We show that the loss in accuracy results from the deductive basis of parsers, not the EBL framework. D-EBL does not extend the deductive closure and acquires only empirical (disambiguation) knowledge. The accuracy declines due to cached errors, the statistical bias the filters introduce and the usage of shortcuts with limited contextual information.

Alternatively, if the parser uses abduction, the deductive closure of the parser enlarges. This makes accuracy improvements possible - not a logical consequence. In practice, the extended deductive closure compensates for negative factors such as wrong parses or unbalanced distributions in the cache.

On a more abstract level, the paper treats the problem of automatic knowledge acquisition for Chinese NLP. Theory and practice show that abduction-based NLP applications acquire new knowledge and increase accuracy and speed. Future research will maximize the gains.

## References


Steven Abney. 1996. Partial Parsing via Finite-State Cascades. In *Proceedings of the ESSLLI '96 Robust Parsing Workshop*.

Rens Bod and Ronald M. Kaplan. 1998. A probabilistic corpus-driven model for lexical-functional analysis. In *COLING-ACL'98*.

Michael Carl and Philippe Langlais. 2003. Tuning general translation knowledge to a sublanguage. In *Proceedings of CLAW 2003*, Dublin, Ireland, May, 15-17.

Hsing-Wu Chang. 1994. Word segmentation and sentence parsing in reading Chinese. In *Advances in the Study of Chinese Language Processing*, National Taiwan University, Taipei.

Keh-Jiann Chen. 1996. A model for robust Chinese parser. *Computational Linguistics and Chinese Language*, 1(1):183–204.

Yanis Dimopoulos and Antonis Kakas. 1996. Abduction and inductive learning. In L. De Taedt, editor, *Advances in Inductive Logic Programming*, pages 144–171. IOS Press.

Chu-Ren Huang and Keh-Jiann Chen. 1992. A Chinese corpus for linguistics research. In *COLING'92*, Nantes, France.

Chu-Ren Huang, Feng-Yi Chen, Keh-Jiann Chen, Zhaoming Gao, and Kuang-Yu Chen. 2000. Sinica treebank: Design criteria, annotation guidelines and online interface. In M. Palmer, M. Marcus, A. K. Joshi, and F. Xia, editors, *Proceedings of the Second Chinese Language Processing Workshop*, Hong Kong, October. ACL.

Aravind K. Joshi. 2003. Tree-adjoining grammars. In R. Mitkov, editor, *The Oxford Handbook of Computational Linguistics*. Oxford University Press, Oxford.

Yves Lepage. 1999. Open set experiments with direct analysis by analogy. In *Proceedings NLPRS'99 (Natural Language Processing Pacific Rim Symposium)*, pages 363–368, Beijing.

Gary F. Marcus, Steven Pinker, Michael Ullman, Michelle Hollander, John T. Rosen, and Fei Xu. 1992. *Overregularization in Language Learning*. Monographs of the Society for Research in Child Development, 57 (No. 4, Serial No. 228).

Steven Minton. 1990. Quantitative results concerning the utility problem of explanation-based learning. *Artificial Intelligence*, 42:363–393.

Tom S. Mitchel, R. Keller, and S. Kedar-Cabelli. 1986. Explanation-based generalization: A unifying view. *Machine Learning*, 1(1).

Günter Neumann. 1994. Application of explanation-based learning for efficient processing of constraint-based grammars. In *The 10th Conference on Artificial Intelligence for Applications*, San Antonio, Texas.

Manny Rayner and Christer Samuelsson. 1994. Corpus-based grammar specification for fast analysis. In *Spoken Language Translator: First Year Report*, SRI Technical Report CRC-043, pg. 41-54.

Khalil Sima'an. 1997. Explanation-based leaning of partial-parsers. In W. Daelemans, A. van den Bosch, and A. Weijters, editors, *Workshop Notes of the ECML/ML Workshop on Empirical Learning of Natural Language Processing Tasks*, pages 137–146, Prague, Czech Republic, April.

Bangalore Srinivas and Aravind K. Joshi. 1995. Some novel applications of explanation-based learning to parsing lexicalized tree-adjoining grammars. In *33th Annual Meeting of the ACL*, Cambridge, MA.

Bangalore Srinivas and Aravind K. Joshi. 1999. Supertagging: An approach to almost parsing. *Computational Linguistics*, 25(2):237–265.

Oliver Streiter, Judith Knapp, and Leonhard Voltmer. 2003. Gymn@zilla: A browser-like repository for open learning resources. In *ED-Media, World Conference on Educational Multimedia, Hypermedia & Telecommunications*, Honolulu, Hawaii, June, 23-28.

Oliver Streiter. 2002a. Abduction, induction and memorizing in corpus-based parsing. In *ESSLLI-2002 Workshop on "Machine Learning Approaches in Computational Linguistics"*, pages 73–90, Trento, Italy, August 5-9.

Oliver Streiter. 2002b. Treebank development with deductive and abductive explanation-based learning: Exploratory experiments. In *Workshop on Treebanks and Linguistic Theories 2002*, Sozopol, Bulgaria, September 20-21.

Prasad Tadepalli. 1991. A formalization of explanation-based macro-operator learning. In *IJCAI, Proceedings of the International Joint Conference of Artificial Intelligence*, pages 616–622, Sydney, Australia. Morgan Kaufmann.

Chunfa Yuang, Changming Huang, and Shimei Pan. 1992. Knowledge acquisition and Chinese parsing based on corpus. In *COLING'92*.

Jakub Zavrel and Walter Daelemans. 1997. Memory-based learning: Using similarity for smoothing. In W. Daelemans, A. van den Bosch, and A. Weijters, editors, *Workshop Notes of the ECML/ML Workshop on Empirical Learning of Natural Language Processing Tasks*, pages 71–84, Prague, Czech Republic, April.