In *Proceedings of the Sixteenth International Conference on Machine Learning*, 87–96, 1999.

# Combining Error-Driven Pruning and Classification for Partial Parsing

**Claire Cardie, Scott Mardis, and David Pierce**
Department of Computer Science
Cornell University
Ithaca, NY 14853
cardie, mardis, pierce@cs.cornell.edu

## Abstract

We present a new approach to partial parsing of natural language texts that relies on machine learning methods. The approach combines corpus-based grammar induction with a very simple pattern-matching algorithm and an optional constituent verification step. The grammar induction algorithm acquires a set of rules for each level of linguistic analysis using a new technique for error-driven pruning of treebank grammars. The constituent verification step employs standard inductive learning techniques as an additional precision-enhancing device. We evaluate the approach on four partial parsing data sets and find that performance is very good (over 93% precision and recall) for applications that require or prefer fairly simple constituent bracketing. As the complexity of the partial parsing task increases, however, our approach lags the performance of competing approaches. We explain these differences in terms of the knowledge sources employed by each method and describe a number of features that make the approach attractive for large-scale, practical NLP applications.

## 1 Introduction

Partial natural language parsers provide a shallow analysis of the syntactic relationships that exist in a sentence. Figure 1, for example, compares one possible partial parse of a sentence with its full syntactic analysis as provided in the Penn Treebank Wall Street Journal (WSJ) corpus (Marcus *et al.*, 1993). Partial parsers produce a fairly flat parse tree in comparison to their full-parse counterparts, omitting some levels of linguistic analysis or leaving them for subsequent modules of the natural language processing (NLP) system. Nevertheless, the linguistic relationships that can be identified by partial parsers are rich enough to support a number of large-scale natural language processing applications including information extraction, phrase identification in information retrieval, named entity identification, and a variety of text-mining operations. In addition, partial parsers are typically very fast when compared to full parsers.

Many shallow parsers (e.g. Abney (1996), Hobbs *et al.* (1997)) rely on finite-state pattern-matching techniques to recognize all syntactic and semantic entities in a sentence (Roche and Schabes, 1997b). Parsing proceeds in stages: Each stage can be viewed as a finite-state transducer that regroups or relabels the tokens or constituents produced in preceding levels. The initial stages perform relatively simple tasks including tokenization, part-of-speech tagging, and simple noun phrase identification; all linguistic relationships that require higher-level attachment decisions are identified in subsequent stages and rely on output from earlier levels.

The patterns, or grammars, that drive each stage of processing traditionally have been designed manually (e.g. Voutilainen (1993), Bourigault (1992)). Also, it has been difficult to directly compare partial parsing techniques since most were evaluated by hand on a small test set. Recent work, however, has attempted the automatic acquisition of partial parsers and their evaluation on a large test corpus annotated with correct parses by an impartial third party. Ramshaw and Marcus (1998), for example, applied transformation-based learning (Brill, 1995) to the problem of noun phrase chunking, a basic step for most partial parsers.

Full parse:

```
(S   (NP-SBJ South Korea)
     (VP registered
          (NP (NP a trade deficit) (PP of (NP $101 billion)))
          (PP-TMP in (NP October)) ,
          (S-ADV (VP reflecting (NP (NP the country) 's economic sluggishness)))) .)
```

Partial parse:
[$_{\text{Subject}}$ [$_{\text{NP}}$ South Korea]] [$_{\text{V}}$ registered] [$_{\text{Object}}$ [$_{\text{NP}}$ a trade deficit]] of [$_{\text{NP}}$ \$101 billion]
in [$_{\text{NP}}$ October], [$_{\text{V}}$ reflecting] [$_{\text{NP}}$ the country]'s [$_{\text{Object}}$ [$_{\text{NP}}$ economic sluggishness]].

Figure 1: Comparison of a Full vs. a Partial Syntactic Parse for the Sentence: *South Korea registered a trade deficit of \$101 billion in October, reflecting the country's economic sluggishness.* The partial parse marks the subject, verb, direct object, and noun phrase information only.

Their noun phrase bracketer learns a set of transformation rules. Each transformation updates the noun phrase bracketings associated with a single word based on local features, such as neighboring words, part-of-speech tags, and bracket boundaries. To identify base noun phrases in a novel text, the learned transformations are applied, in order, across the entire document. Argamon *et al.* (1998) developed a variation of memory-based learning (Stanfill and Waltz, 1986) for two partial parsing tasks — base noun phrase identification and verb-object recognition. During training, their MBSL algorithm saves the entire raw training corpus, which has been annotated with parts of speech, and noun phrase and verb-object brackets. Generalization of the implicit partial parsing rules in the training corpus occurs at application time — MBSL searches the novel text for tag sequences or combinations of tag subsequences (tiles) that occurred during training in a similar context.

This paper presents a new approach to partial parsing that relies on machine learning methods. The approach is implemented in a shallow parser called Empire[1] that combines corpus-based grammar induction with a very simple pattern-matching algorithm and an optional constituent verification step. The grammar induction algorithm acquires a set of rules for each level of linguistic analysis using a new technique for error-driven pruning of treebank grammars (Cardie and Pierce, 1998). In addition, we add a constituent verification step that employs standard inductive learning techniques as a precision-enhancing device.

We evaluate our approach on four data sets that comprise two partial parsing tasks — base noun phrase identification and verb-object identification. We find that the performance of our treebank approach is very good (over 93% precision and recall) for applications that require or prefer fairly simple constituent bracketing. As the complexity of the partial parsing task increases, however, the treebank approach lags the performance of competing methods. We explain these differences in terms of the knowledge sources employed by each method. Nevertheless, our approach has a number of features that make it attractive for large-scale, practical NLP applications:

- both the grammar induction algorithm and the pattern-matching parser are exceedingly simple;
- the parser is very fast, operating in time linear in the length of the text when the constituent verification phase is not used;
- the learned grammars can be easily modified for use with corpora that differ from the training texts;
- the approach is very flexible in that it allows system developers to choose a performance metric (e.g. precision, recall, F-measure) with which to tune the grammar that best supports the goals of the larger NLP application.

The next section provides a definition for, and examples of, the two partial parsing tasks. Sections 3 and 4 describe and evaluate the basic grammar induction approach. Section 5 presents the optional constituent verification step. Section 6 evaluates the entire approach. We compare Empire with related work in Section 7.

[1]The name refers to our focus on empirical methods for development of the system.

## 2 Base Noun Phrase and Verb-Object Identification

We focus on two shallow parsing tasks: base noun phrase identification and verb-object identification. In this work we define a *base noun phrase* (base NP) to be a simple, nonrecursive noun phrase — a noun phrase that does not contain other noun phrase descendants. In the partial parse of Figure 1, for example, *a trade deficit of $101 billion* and *the country's economic sluggishness* are too complex to be base NPs; instead, they contain four simpler noun phrases, each of which is considered a base NP: *a trade deficit*, *$101 billion*, *the country*, and *economic sluggishness*. For base NP identification then, the goal of Empire is to take a sentence as input and produce a version of the sentence in which all base NPs are bracketed.

*Verb-object* (VO) recognition amounts to identifying the main verb and direct object head of each clause in a sentence. Verb-object identification is performed on a tag sequence in which the base noun phrases have already been identified. Here the goal for Empire is to bracket potential verb-object constituents starting with the main verb token and ending with the base noun phrase that contains the object token: *South Korea* [$_{\text{VO}}$ *registered a trade deficit*] *of $101 billion in October,* [$_{\text{VO}}$ *reflecting the country's economic sluggishness*]. Following the application of the parser, a small set of rules is applied to the verb-object "constituent" bracket to extract the verb and object. The above verb-object bracketing, for example, would produce the following output where *registered/deficit* and *reflecting/sluggishness* are the verb-object pairs: *South Korea* [$_{\text{VO}}$ [$_{\text{V}}$ *registered*] *a trade* [$_{\text{O}}$ *deficit*]] *of $101 billion in October,* [$_{\text{VO}}$ [$_{\text{V}}$ *reflecting*] *the country's economic* [$_{\text{O}}$ *sluggishness*]].

## 3 Grammar Induction

Figure 2 depicts Empire's approach to shallow parsing using the base noun phrase relationship as a running example. To learn to identify a particular linguistic relationship, REL, Empire requires a corpus that has been annotated with brackets indicating that relationship. More specifically, we assume that the training corpus is a sequence of elements (either words or constituents) $e_1, e_2, \ldots$, along with a set of annotations or brackets $b_{(i_1,j_1)}, b_{(i_2,j_2)}, \ldots$, where $b_{(i,j)}$ indicates that elements $i$ through $j$ represent an instance of REL: [$_{\text{REL}}$ $e_i, \ldots, e_j$].

The goal of the training phase is to create a grammar for REL based on the training corpus:

1. *Run all required lower levels of Empire to assign a tag, $t_i$, to each element $e_i$ in the training corpus.* For the base NP relationship, this amounts to assigning a part-of-speech tag to every token in the corpus. Verb-object recognition, on the other hand, requires both part-of-speech and base NP information: Each element in the training corpus is assigned either an NP label (for sentence elements recognized as base NPs) or a part-of-speech tag (for all tokens outside of base NPs).

2. *Extract from each* REL $b_{(i,j)}$ *in the training corpus its sequence of tags $t_i, \ldots, t_j$ to form a grammar.* In Figure 2, for example, the first base NP in the corpus, [$_{\text{NP}}$ *it*], produces the tag-sequence rule ⟨PRP⟩; verb-object rules contain both part-of-speech and NP tags.

3. *Remove duplicate rules from the grammar.*

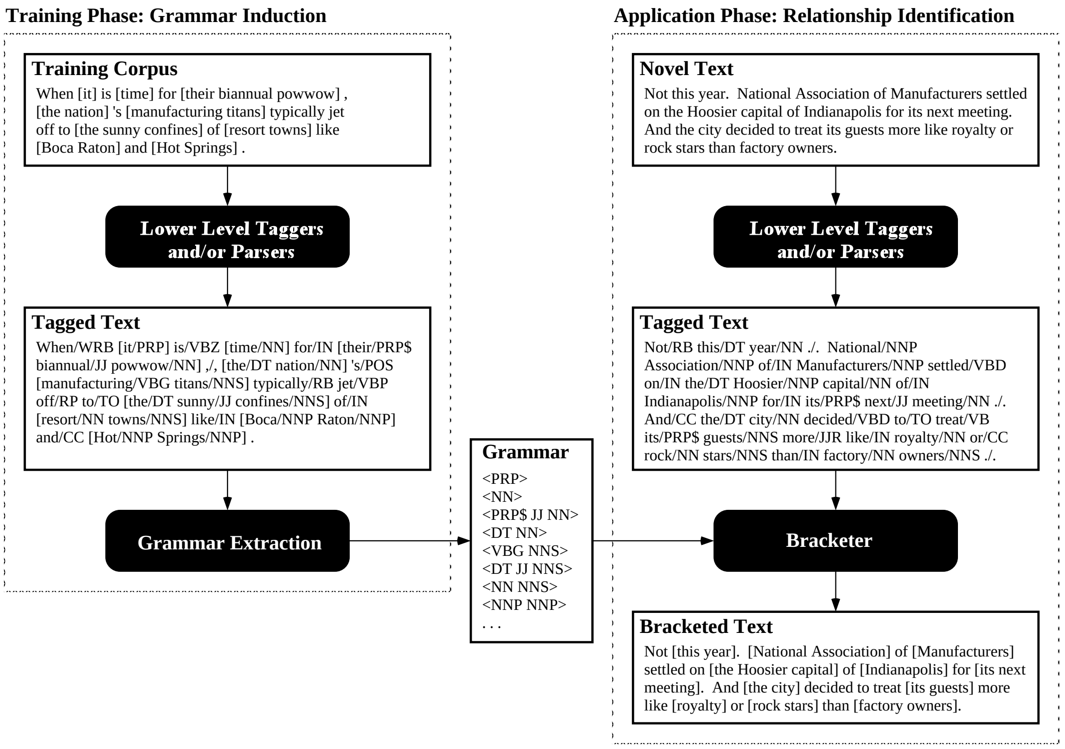
The resulting grammar can then be used to identify the linguistic relationship in a novel text in the application phase.

1. *Run all required lower levels of the parser* to assign tags $t_1, t_2, \ldots$ to the elements $e_1, e_2, \ldots$ in the input text.

2. *Proceed through the tagged text from left to right*, at each point matching the rules against the remaining tags $t_i, t_{i+1}, \ldots$ in the text.

3. If there are multiple rules that match beginning at $t_i$, *use the longest matching rule R.* Continue matching at $t_{i+|R|}$.

With the rules stored in an appropriate data structure, this greedy "parsing" of constituents is very fast. Unfortunately, grammars extracted in this fashion pick up many "bad" rules due to noise in the training data (including annotation errors, part-of-speech tagging errors, and genuine ambiguities). As a result, it has become common practice to assign probabilities to the rules based on their frequency of occurrence and then use a probabilistic parser to apply the grammar to unseen text (Charniak, 1996). Our grammar induction framework employs a much simpler solution: we include a pruning phase to eliminate bad rules from the grammar and retain the longest-match heuristic.



Training Phase: Grammar Induction
Training Corpus
When [it] is [time] for [their biannual powwow] , [the nation] 's [manufacturing titans] typically jet off to [the sunny confines] of [resort towns] like [Boca Raton] and [Hot Springs] .
Lower Level Taggers and/or Parsers
Tagged Text
When/WRB [it/PRP] is/VBZ [time/NN] for/IN [their/PRP$ biannual/JJ powwow/NN] ,/, [the/DT nation/NN] 's/POS [manufacturing/VBG titans/NNS] typically/RB jet/VBP off/RP to/TO [the/DT sunny/JJ confines/NNS] of/IN [resort/NN towns/NNS] like/IN [Boca/NNP Raton/NNP] and/CC [Hot/NNP Springs/NNP] .
Grammar Extraction
Grammar
<PRP>
<NN>
<PRP$ JJ NN>
<DT NN>
<VBG NNS>
<DT JJ NNS>
<NN NNS>
<NNP NNP>
. . .
Application Phase: Relationship Identification
Novel Text
Not this year. National Association of Manufacturers settled on the Hoosier capital of Indianapolis for its next meeting. And the city decided to treat its guests more like royalty or rock stars than factory owners.
Lower Level Taggers and/or Parsers
Tagged Text
Not/RB this/DT year/NN ./. National/NNP Association/NNP of/IN Manufacturers/NNP settled/VBD on/IN the/DT Hoosier/NNP capital/NN of/IN Indianapolis/NNP for/IN its/PRP$ next/JJ meeting/NN ./. And/CC the/DT city/NN decided/VBD to/TO treat/VB its/PRP$ guests/NNS more/JJR like/IN royalty/NN or/CC rock/NN stars/NNS than/IN factory/NN owners/NNS ./.
Bracketer
Bracketed Text
Not [this year]. [National Association] of [Manufacturers] settled on [the Hoosier capital] of [Indianapolis] for [its next meeting]. And [the city] decided to treat [its guests] more like [royalty] or [rock stars] than [factory owners].

Figure 2: Shallow Parsing in Empire. The base noun phrase relationship is used as an example.

## 3.1 Error-Driven Pruning

Empire's grammar pruning procedure is shown in Figure 3. First, we divide the training corpus into two parts: an extraction corpus and a pruning corpus. The initial grammar for the linguistic relationship is derived from the extraction corpus as described above. Next, the pruning corpus is used to evaluate the grammar and produce a ranking of the rules in terms of their utility in identifying the linguistic relationship. More specifically, we use the rule set and the longest match heuristic to find all instances of the relationship in the pruning corpus. Performance of the rule set is measured in terms of labeled precision ($P$):

$$P = \frac{\text{\# of correct proposed RELs}}{\text{\# of proposed RELs}}$$

We then assign to each rule a score that denotes the "net benefit" achieved by using the rule during parsing of the pruning corpus. The benefit of rule $r$ is given by $B_r = C_r - E_r$ where $C_r$ is the number of RELs correctly identified by $r$, and $E_r$ is the number of precision errors for which $r$ is responsible. Alternatively, a frequency-based benefit measure can be employed: We scale $B_r$ by the rule frequency to prune low-frequency low-precision rules before high-frequency rules with the same precision.

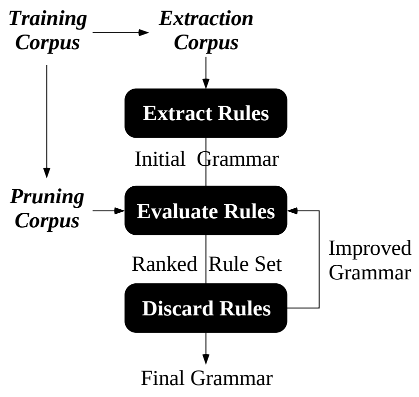


Figure 3: Error-Driven Pruning of Treebank Grammars

The benefit scores from evaluation on the pruning corpus are used to discard the worst rules from the grammar. In the experiments below, we use two types of pruning. Each iteration of *threshold pruning* discards rules whose score is less than a predefined threshold $T$; it halts when all rules have a score above $T$. *In-*

*cremental pruning* discards the $N$ worst rules in each iteration and then selects the grammar that yields either the maximum precision or the maximum recall on the pruning data, depending on the goals of the NLP application.

For a more detailed description of error-driven pruning, see Cardie and Pierce (1998). The notion of "pruning" inductive hypotheses has been explored previously in machine learning. Quinlan (1987), for example, presents several methods for simplifying decision trees using reduced-error pruning and a separate test set. If Empire's partial parsing approach is viewed as a very simple instantiation of case-based learning, then our pruning technique most resembles instance editing algorithms that discard training instances with poor classification accuracies (e.g. IB3 (Aha *et al.*, 1991)).

## 4 Evaluation of Error-Driven Pruning

**Data Sets.** In spite of the definitions in Section 2, there can be substantial differences in how base noun phrase and verb-object relationships can be encoded in any corpus. As a result, we created two data sets for each linguistic relationship, one data set that reflects a simple implementation of each relationship, and one that incorporates additional linguistic complexities. All are derived from the Penn Treebank WSJ corpus. In the Complex Base NP data set, each base NP corresponds to a non-recursive noun phrase in the Treebank parse. In particular, this data set is meant to duplicate the data used in Ramshaw and Marcus (1998) except for our handling of possessives. The Simple Base NP data set further simplifies Treebank base NPs that contain: (1) nominal conjunctions, (2) prepositions, (3) possessives, or (4) leading and trailing adverbs and verbs.

In the Complex Verb-Object data set, there is a verb-object pair for every verb and direct object component in the Treebank parse. In particular, if the object is a conjunctive phrase, the data set encodes separate verb-object pairs for each component of the conjunction. The Simple Verb-Object data set is meant to duplicate the Argamon *et al.* (1998) data and: (1) simplifies direct objects that are conjunctions or appositives, (2) simplifies conjunctive verbs, (3) simplifies ditransitives, and (4) omits objects marked by verb particles and copular verbs. By evaluating on both data sets for each linguistic task, we can explore the effect of linguistic relationship complexity on parser accuracy.

**Methodology.** All experiments are performed on the 25 sections of the WSJ portion of the Penn Treebank II corpus. Performance is measured in terms of precision (P) and recall (R):

$$P = \frac{\text{\# of correct proposed RELs}}{\text{\# of proposed RELs}}$$

$$R = \frac{\text{\# of correct proposed RELs}}{\text{\# of RELs in the annotated text}}$$

All results are averages using five-fold cross-validation at the document level; the same fold divisions are employed across all experiments. All experiments use the original (Treebank) segmentation for sentences and tokens.

For both base NP and verb-object recognition, part-of-speech tags were generated using Mitre's version of the Brill tagger (Brill, 1995). For verb-object identification, Empire itself identifies the base noun phrases. The threshold pruning experiments use $T = 1$; the incremental pruning experiments use $N = 3$ for verb-objects and $N = 10$ for base NPs. The verb-object experiments use recall-based pruning; incremental pruning for verb-objects uses the frequency-based benefit measure and threshold pruning employs the standard benefit measure.[2]

**Results.** Table 1 summarizes the performance of the Empire parser on both the Complex and Simple corpora for each linguistic relationship using incremental and threshold pruning. The first row of results for each data set shows the performance of the initial, unpruned grammars. The next two rows show the performance of the automatically pruned rule sets. As expected, the initial rule set performs quite poorly. For base NPs, both automated approaches provide significant increases in recall and precision. Note also the relatively small difference between the threshold and incremental pruning methods. For some applications, the minor drop in performance for threshold pruning may be worth the decrease in training time. For the remainder of the paper, we will use threshold pruning for base NP experiments.

For verb-object identification, the pruning results are less consistent. Both methods substantially increase precision; the finer-grained incremental pruning also increases recall. Threshold pruning, however, provides mixed results. For verb-object recognition, there are a few high-frequency, low-precision rules and the coarser

[2]The frequency-based benefit measure and threshold pruning with $T = 1$ are not compatible since all rule benefits would be below 1.

Table 1: Evaluation of Error-Driven Pruning for Base NP Identification and Verb-Object Recognition (P = precision; R = recall)

| Linguistic Relationship | Simple Corpus | Complex Corpus |
|---|---|---|
| Base NP | | |
| unpruned | .221P/.457R | .183P/.353R |
| threshold pruning | .920P/.932R | .870P/.896R |
| incremental pruning | .932P/.934R | .893P/.905R |
| Verb-Object | | |
| unpruned | .199P/.863R | .209P/.681R |
| threshold pruning | .661P/.612R | .650P/.719R |
| incremental pruning | .569P/.895R | .619P/.789R |

threshold pruning is not consistent in its handling of them: Results vary widely on a per-fold basis depending on whether these rules survive pruning. For the remainder of the paper, we will use incremental pruning for verb-object experiments.

Table 1 also clearly indicates the effects of linguistic relationship complexity. As expected, results for the Complex base NP data set are much lower than those for the Simple corpus (-5%P/-3.6%R). Since the two data sets share about 91% of their NPs, this performance drop is caused by problems introduced in the remaining 9%, of which possessives account for 3.6%; conjunctions, 2.3%; and NPs with leading and trailing adverbs and verbs, 2.4%. Analysis of the errors indicates that ambiguities introduced by allowing conjunctions and adverbs/verbs appear to be more difficult than those introduced by possessives.

Although the verb-object task is more difficult than the base NP task it subsumes, the results for the Simple and Complex verb-object data sets appear to contradict the expected trend — there is a large drop in recall (-10.6%) for the Complex verb-objects, but precision increases by 5%. One reason for this is that verb-object pairs with copular verbs are identified in both experiments but are considered errors for the simple data set. In addition, the complex data set yields many more rules (typically longer rules to handle conjunctions and appositives) that are low-frequency, but high-precision. Because of Empire's longest match heuristic, these long rules prevent Empire from identifying some erroneous verb-objects that would be selected by shorter, less accurate rules.

For two of the four data sets, our results can be compared to those obtained in evaluations of other approaches to partial parsing on the Penn Treebank. We ran an evaluation using as test data, the same sections of the Treebank as previous experimenters; the results are in Table 2.[3] Our results on Complex Base NP corpus lag the best published results (Ramshaw and Marcus, 1998) by 3-4%. Note, however, that we make no use of context or lexical (word-level) information; the best results on this data set were obtained using transformation-based learning, which employs both. By controlling for lexicalization in transformation-based learning, Empire performs more comparably (-1.2%P/-0.2%R). Empire also falls short of the best results to date on the Simple Verb-Object corpus. Argamon *et al.* (1998), however, make use of context to determine verb-object brackets. We address this issue for Empire in the next section.

Table 2: Direct Comparison to Previous Work (P = precision; R = recall). Base noun phrases were evaluated on section 20 of the WSJ corpus. Verb-object recognition was tested on section 00 of the WSJ.

| | |
|---|---|
| *Base NPs* | |
| Ramshaw and Marcus (R&M) | .931P/.935R |
| Argamon *et al.* | .916P/.916R |
| R&M (no lexical information) | .901P/.902R |
| Empire (incremental pruning) | .889P/.900R |
| *Verb-Objects* | |
| Argamon *et al.* | .771P/.898R |
| Empire (incremental pruning) | .610P/.935R |

[3]To match the conditions of the Argamon *et al.* evaluation, these tests were performed using the tags directly from the treebank rather than from the Mitre/Brill tagger.

Table 3: Sample Verb-Object Constituent Verification Case for *Many traders predict the U.S. currency will remain stuck in the near term.* VB refers to a base form verb; NN is a singular noun; MD is a modal verb.

| Sentence Elements | Attribute | Value |
|---|---|---|
| | tag-prev2 | nil |
| [$_{\text{NP}}$ Many traders] | tag-prev1 | NP |
| [$_{\text{VO}}$ [$_{\text{V}}$ predict] [$_{\text{NP}}$ the U.S. [$_{\text{O}}$ currency]]] | verb-tag | VB |
| | object-tag | NN |
| will | tag-fol1 | MD |
| remain | tag-fol2 | VB |
| stuck in ... | | |
| | **Class:** | **bad** |

## 5 Constituent Verification

Although the role of context and lexicalization has not been fully explored for shallow parsing, both are known to improve the performance of a number of language learning tasks. The accuracy of Empire's shallow parsing algorithm is therefore likely to improve if context and lexical information can be suitably exploited. This section describes one method for extending Empire to incorporate context and lexicalization without sacrificing the overall simplicity of the shallow parsing approach. We use standard inductive learning techniques in a *constituent verification* step to evaluate the correctness of each proposed linguistic relationship: Those deemed correct will remain in the bracketed output of the shallow parser; those deemed incorrect are discarded. This post-processing stage is clearly a precision-enhancing device: For cascaded partial parsers, it is important that each level of analysis be as accurate at possible. In addition, some NLP applications will benefit more from output that has high precision and reasonable recall (e.g. identification of linguistic relationships for information retrieval) than the converse.

We have examined three machine learning algorithms for constituent verification: decision tree induction using C4.5 (Quinlan, 1992), an unweighted k-nearest neighbor (K-NN) classifier (e.g. IB1 (Aha *et al.*, 1991)), and a K-NN classifier that uses the value-difference metric as the similarity measure (Stanfill and Waltz, 1986). The goal of the training phase is to produce a set of training cases, each of which describes one proposed linguistic relationship and its context. A sample verb-object constituent verification case is shown in Table 3. Each case is a set of six features: the tags for the two elements that precede and that follow the bracket; and tags for the main verb and direct object head. A similar case representation is used for base NPs. Note that the current case representation contains no lexical information — the actual word tokens that comprise the bracket and its context are not included in the cases. It should be clear, however, that lexical information is easy to incorporate — in addition to the tags used in the current representation, cases would also include the token that represents the element. Finally, each training case has a class label ("good" or "bad") indicating the correctness of the proposed relationship.

Training instances are created automatically from the pruning corpus. For the K-NN variations, the value for $k$ is learned automatically during training; during testing, we use a simple majority vote to determine the class label.

Table 4 shows the performance of the three learning algorithms using 10-fold cross validation on data sets of 1000 cases derived from each of the linguistic relationship corpora. In terms of relative accuracy, the three learning algorithms perform comparably on each data set.[4] Since the goal for constituent verification is to increase precision, however, we hypothesize that the learning algorithm with best combination of a high accuracy and low false positive rate would provide the greatest improvement in precision for the partial parsing tasks: the K-NN+VDM alternative has this property for the data sets studied here. We also believe that classification for constituent verification can be improved by controlling the characteristics of the training set — the training data for both linguistic tasks is skewed w.r.t. class distribution; no attempt

[4] $\chi^2$ and two-tailed $t$ tests indicate that K-NN and K-NN+VDM outperform C4.5 ($p \leq .10$) for the Simple Base NP corpus; K-NN+VDM outperforms K-NN and C4.5 ($p \leq .10$) for the Simple Verb-Object corpus; and K-NN+VDM outperforms K-NN and C4.5 ($p \leq .10$) for the Complex Verb-Object corpus.

Table 4: Performance of Inductive Learning Algorithms for Constituent Verification. Results are shown in terms of accuracy (% correct) with standard deviations. False positive rates are provided in *italics*.

| Data Set | C4.5 | K-NN | K-NN + VDM |
|---|---|---|---|
| Simple Base NP | $90.5 \pm 2.2$ | $91.9 \pm 2.6$ | $91.6 \pm 1.8$ |
| *false positive rate* | *7.4* | *8.1* | *6.8* |
| Complex Base NP | $86.7 \pm 3.3$ | $87.4 \pm 2.9$ | $87.0 \pm 2.1$ |
| *false positive rate* | *10.3* | *12.3* | *9.0* |
| Simple Verb-Object | $68.6 \pm 5.2$ | $68.1 \pm 4.7$ | $70.6 \pm 4.9$ |
| *false positive rate* | *19.0* | *24.4* | *19.4* |
| Complex Verb-Object | $74.1 \pm 2.5$ | $73.7 \pm 4.0$ | $75.9 \pm 3.1$ |
| *false positive rate* | *15.9* | *22.0* | *15.5* |

was made to balance the data sets for the experiments reported here.

## 6 Combining Pruning and Constituent Verification

Table 5 shows the effect of constituent verification on base NP and verb-object identification. For these experiments, the learning algorithms for constituent verification were trained on 1000 cases derived from the associated pruning corpus. As hypothesized, the increase in precision is greatest for the K-NN+VDM variation. The results also show that constituent verification is a more effective way to integrate context and improve precision for verb-object identification than for base NP identification: For both the Simple and Complex verb-object data sets, more precision was gained than recall lost (+12.0% precision vs. -9% and -6% recall, respectively). Constituent verification provides much smaller precision gains for base NP recognition. One explanation is that the skewed training data for constituent verification limited performance. It is also likely that accurate verb-object identification simply requires more context, e.g. to distinguish direct objects from subjects of subordinate clauses as in the examples below. In preliminary experiments, however, we find for all of the partial parsing data sets that the precision-recall tradeoff can be further manipulated by controlling the balance of "good" and "bad" cases in the training set: As the percentage of "good" cases is decreased, precision rises and recall falls.

In spite of the addition of context, Empire still falls below the best reported results on the verb-object task: Even after re-training Empire using perfect tags (to match the Argamon *et al.* (1998) training procedure), precision remains about 6 precision points lower at the same level of recall (72.5P/87.5R). Only by assuming both perfect tags and perfect NPs can Empire surpass their results. We would expect to recover some of this discrepancy by replacing the longest-match heuristic with one that chooses the best combination of possible bracketings. For base NP identification, precision gains are modest, leaving Empire still shy of Argamon *et al.*'s results.

We conclude this section with an example of the constituent verification step in action: Table 6 shows a proposed verb-object pair that is correctly discarded (on the left) and the top-ranked case retrieved in response to the test case (on the right).

## 7 Related Work and Conclusions

In the evaluation sections above, we compared Empire to two other machine learning methods that have been applied to partial parsing — Ramshaw and Marcus's transformation-based bracketer and Argamon *et al.*'s MBSL. In addition to differences in recall and precision, the methods also vary in other ways. For example, the transformation-based bracketer requires the most training time, with one pass through the training data for each candidate template for each new transformation. Empire needs only a handful to a few hundred passes for pruning (for threshold and incremental pruning, respectively), while MBSL trains in a single pass over the training data. In terms of runtime speed, the Empire longest-match bracketer is the fastest, using a quick linear time algorithm. The transformation-based bracketer can be equally fast, provided that the transformations are precompiled into a single finite-state transducer (Roche and Schabes, 1997a). Empire's increase in running time for constituent verification depends on the learning algorithm used. We

Table 5: Effect of Constituent Verification (P = precision; R = recall)

| Data Set | No Constituent Verification | Constituent Verification Using C4.5 | K-NN | K-NN+VDM |
|---|---|---|---|---|
| Simple Base NP | .920P/.932R | .922P/.925R | .920P/.931R | .928P/.919R |
| Complex Base NP | .870P/.896R | .879P/.877R | .872P/.893R | .896P/.830R |
| Simple Verb-Object | .569P/.895R | .678P/.750R | .655P/.796R | .690P/.806R |
| Complex Verb-Object | .619P/.789R | .727P/.701R | .692P/.758R | .741P/.728R |

Table 6: Sample Verb-Object Constituent Verification Case for *. . . said they . . .* in *Two Japanese scientists said they discovered an antibody that. . . .* In the case representation, VBD is a past tense verb; PRP is a personal pronoun; VO is a verb-object pair; IN is a preposition or subordinating conjunction. The right-hand column shows the top-ranked retrieved case.

| Test Case Sentence Elements | Case Representation Attribute | Value | Retrieved Case Sentence Elements |
|---|---|---|---|
| | tag-prev2 | nil | |
| [$_{NP}$ Two Japanese scientists] | tag-prev1 | NP | [$_{NP}$ Clinton Gas Systems Inc.] |
| [$_{VO}$ [$_{V}$ **said**] [$_{NP}$ [$_{O}$ **they**]]] | verb-tag | VBD | [$_{VO}$ [$_{V}$ **said**] [$_{NP}$ [$_{O}$ **it**]]] |
| | object-tag | PRP | |
| [$_{VO}$ [$_{V}$ discovered] [$_{NP}$ an antibody]] | tag-fol1 | VO | [$_{VO}$ [$_{V}$ received] [$_{NP}$ a contract]] |
| that . . . | tag-fol2 | IN | from . . . |
| | | | **Class: bad** |

believe MBSL to be the slowest bracketer, comparing multiple tiling covers to score each potential NP. In addition, the runtime space burden for MBSL is very large because the memory-base stores the entire training corpus. In contrast, the transformation-based and Empire bracketers store only a few hundred to a few thousand transformation or grammar rules at runtime (for verb-object and base NP recognition, respectively).[5]

Finally, Empire's main training product — a tag-sequence grammar — can be modified for new genres of text without retraining. This level of portability is impossible with MBSL, which creates no explicit constituent grammar; and adding or deleting transformations to handle new text genres would likely yield unpredictable results with the transformation-based bracketer. In Empire, individual rules can easily be removed from or added to the grammar.

[5]Daelemans *et al.* (1999) present a learning algorithm for base NP chunking that is similar to MBSL in its memory-based approach. Their results, however, are not comparable to those reported here: They measure performance in terms of the number of words correctly bracketed rather than measuring the number of completely correct noun phrase brackets.

In conclusion, we presented a new approach to partial parsing of natural language texts that combines error-driven pruning of treebank grammars and an optional classsification-based constituent verification step. In an evaluation on four data sets, we find that error-driven pruning provides substantial increases in recall and precision over the unpruned rule sets; constituent verification provides additional increases in precision, but works especially well for the more complex partial parsing tasks. The approach also has a number of attractive features: the grammar induction algorithm and the pattern-matching parser are exceedingly simple; the parser is very fast; the learned grammars can be easily modified for use with corpora that differ from the training texts. In addition, the approach is very flexible in that it allows system developers to choose a performance metric with which to tune the grammar that best supports the goals of the larger NLP application.

Furthermore, we believe that the treebank approach to grammar acquisition may be general enough to tackle other natural language learning tasks including the acquisition of patterns for information extraction and for higher-level text-mining operations. It might also be applied to tasks outside of natural language processing applications including learning simple languages for agent communication or in grammar-based biology problems.

Finally, we believe that there are a number of extensions and improvements that could be made to obtain additional performance gains at the possible expense of some added complexity: The naive longest-match heuristic can be replaced by a dynamic programming algorithm that considers all combinations of constituent brackets; machine learning algorithms could be used to correct, rather than discard, erroneous bracketings; lexicalized features can be added to the case representation; the constituent verification phase could be incorporated directly into the pruning algorithm; training data for constituent verification can be better balanced with respect to positive and negative instances.

**Acknowledgments.** This work was supported in part by NSF Grants IRI–9624639 and GER–9454149. We thank Mitre for providing their part-of-speech tagger.

## References

(Abney, 1996) Abney, Steven. 1996. Partial Parsing via Finite-State Cascades. In *Workshop on Robust Parsing*. 8–15.

(Aha *et al.*, 1991) Aha, D.; Kibler, D.; and Albert, M. 1991. Instance-Based Learning Algorithms. *Machine Learning* 6(1):37–66.

(Argamon *et al.*, 1998) Argamon, S.; Dagan, I.; and Krymolowski, Y. 1998. A Memory-Based Approach to Learning Shallow Natural Language Patterns. In *Proceedings of the 36th Annual Meeting of the ACL and COLING-98*. Association for Computational Linguistics. 67–73.

(Bourigault, 1992) Bourigault, D. 1992. Surface Grammatical Analysis for the Extraction of Terminological Noun Phrases. In *Proceedings, COLING-92*. 977–981.

(Brill, 1995) Brill, Eric 1995. Transformation-Based Error-Driven Learning and Natural Language Processing: A Case Study in Part-of-Speech Tagging. *Computational Linguistics* 21(4):543–565.

(Cardie and Pierce, 1998) Cardie, C. and Pierce, D. 1998. Error-Driven Pruning of Treebank Grammars for Base Noun Phrase Identification. In *Proceedings of the 36th Annual Meeting of the ACL and COLING-98*. Association for Computational Linguistics. 218–224.

(Charniak, 1996) Charniak, E. 1996. Treebank Grammars. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, Portland, OR. AAAI Press / MIT Press. 1031–1036.

(Daelemans *et al.*, 1999) Daelemans, W.; Bosch, A.van den; and Zavrel, J. 1999. Forgetting Exceptions is Harmful in Language Learning. *Machine Learning* 11(1–3):11–43.

(Hobbs *et al.*, 1997) Hobbs, J.; Appelt, D.; Bear, J.; Israel, D.; Kameyama, M.; Stickel, M.; and Tyson, M. 1997. FASTUS: A Cascaded Finite-State Transducer for Extracting Information from Natural-Language Text. In E. Roche and Y. Schabes, editors, *Finite-State Language Processing*. MIT Press, Cambridge, MA. 383–406.

(Marcus *et al.*, 1993) Marcus, M.; Marcinkiewicz, M.; and Santorini, B. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics* 19(2):313–330.

(Quinlan, 1987) Quinlan, J. R. 1987. Simplifying Decision Trees. *International Journal of Man-Machine Studies* 27:221–234.

(Quinlan, 1992) Quinlan, J. R. 1992. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA.

(Ramshaw and Marcus, 1998) Ramshaw, Lance A. and Marcus, Mitchell P. 1998. Text chunking using transformation-based learning. In *Natural Language Processing Using Very Large Corpora*. Kluwer. Originally appeared in WVLC95, 82–94.

(Roche and Schabes, 1997a) Roche, E. and Schabes, Y. 1997. Deterministic part-of-speech tagging with finite-state transducers. In Roche, E. and Schabes, Y., editors, *Finite-State Language Processing*. MIT Press. chapter 7, 205–239.

(Roche and Schabes, 1997b) Roche, E. and Schabes, Y., editors, 1997. *Finite-State Language Processing*. MIT Press, Cambridge, MA.

(Stanfill and Waltz, 1986) Stanfill, C. and Waltz, D. 1986. Toward Memory-based Reasoning. *Communications of the ACM* 29:1213–1228.

(Voutilainen, 1993) Voutilainen, A. 1993. NPTool, A Detector of English Noun Phrases. In *Proceedings of the Workshop on Very Large Corpora*. Association for Computational Linguistics. 48–57.