Trace Prediction and Recovery
With Unlexicalized PCFGs and Slash Features

Helmut Schmid
IMS, University of Stuttgart
schmid@ims.uni-stuttgart.de

Abstract
This paper describes a parser which generates parse trees with empty elements in which traces and fillers are co-indexed. The parser is an unlexicalized PCFG parser which is guaranteed to return the most probable parse. The grammar is extracted from a version of the PENN treebank which was automatically annotated with features in the style of Klein and Manning (2003). The annotation includes GPSG-style slash features which link traces and fillers, and other features which improve the general parsing accuracy. In an evaluation on the PENN treebank (Marcus et al., 1993), the parser outperformed other unlexicalized PCFG parsers in terms of labeled bracketing f-score. Its results for the empty category prediction task and the trace-filler co-indexation task exceed all previously reported results with 84.1% and 77.4% f-score, respectively.

1 Introduction
Empty categories (also called null elements) are used in the annotation of the PENN treebank (Marcus et al., 1993) in order to represent syntactic phenomena like constituent movement (e.g. wh-extraction), discontinuous constituents, and missing elements (PRO elements, empty complementizers and relative pronouns). Moved constituents are co-indexed with a trace which is located at the position where the moved constituent is to be interpreted. Figure 1 shows an example of constituent movement in a relative clause.

Figure 1: Co-indexation of traces and fillers

for determining the predicate-argument structure of a sentence. However, most broad-coverage statistical parsers (Collins, 1997; Charniak, 2000, and others) which are trained on the PENN treebank generate parse trees without empty categories. In order to augment such parsers with empty category prediction, three rather different strategies have been proposed: (i) pre-processing of the input sentence with a tagger which inserts empty elements into the input string of the parser (Dienes and Dubey, 2003b; Dienes and Dubey, 2003a). The parser treats the empty elements like normal input tokens. (ii) post-processing of the parse trees with a pattern matcher which adds empty categories after parsing (Johnson, 2001; Campbell, 2004; Levy and Manning, 2004) (iii) in-processing of the empty categories with a slash percolation mechanism (Dienes and Dubey, 2003b; Dienes and Dubey, 2003a). The empty elements are here generated by the grammar.

Good results have been obtained with all three approaches, but (Dienes and Dubey, 2003b) reported that in their experiments, the in-processing of the empty categories only worked with lexicalized parsing. They explain that their unlex-
icalized PCFG parser produced poor results because the beam search strategy applied there eliminated many correct constituents with empty elements. The scores of these constituents were too low compared with the scores of constituents without empty elements. They speculated that “doing an exhaustive search might help” here.

In this paper, we confirm this hypothesis and show that it is possible to accurately predict empty categories with unlexicalized PCFG parsing and slash features if the true Viterbi parse is computed. In our experiments, we used the BitPar parser (Schmid, 2004) and a PCFG which was extracted from a version of the PENN treebank that was automatically annotated with features in the style of (Klein and Manning, 2003).

2 Feature Annotation

A context-free grammar which generates empty categories has to make sure that a filler exists for each trace and vice versa. A well-known technique which enforces this constraint is the GPSG-style percolation of a slash feature: All constituents on the direct path from the trace to the filler are annotated with a special feature which represents the category of the filler as shown in figure 2. In order to restore the original treebank annotation with co-reference indices from the representation with slash features, the parse tree has to be traversed starting at a trace node and following the nodes annotated with the respective filler category until the filler node is encountered. Normally, the filler node is a sister node of an ancestor node of the trace, i.e. the filler c-commands the trace node, but in case of clausal fillers it is also possible that the filler dominates the trace. An example is the sentence “S-1 She had – he informed her *- I – kidney trouble” whose parse tree is shown in figure 3.

Besides the slash features, we used other features in order to improve the parsing accuracy of the PCFG, inspired by the work of Klein and Manning (2003). The most important ones of these features will now be described in detail. Section 4.3 shows the impact of these features on labeled bracketing accuracy and empty category prediction.

VP feature VPs were annotated with a feature that distinguishes between finite, infinitive, to-infinitive, gerund, past participle, and passive VPs.

S feature The S node feature distinguishes between imperatives, finite clauses, and several types of small clauses.

Parent features Modifier categories like SBAR, PP, ADVP, RB and NP-ADV were annotated with a parent feature (cf. Johnson (1998)). The parent features distinguish between verbal (VP), adjectival (ADJP, WHADJP), adverbial (ADVP, WHADVP), nominal (NP, WHNP, QP), prepositional (PP) and other parents.

PENN tags The PENN treebank annotation uses semantic tags to refine syntactic categories. Most parsers ignore this information. We preserved the tags ADV, CLR, DIR, EXT, IMP, LGS, LOC, MNR, NOM, PRD, PRP, SBJ and TMP in combination with selected categories.

Auxiliary feature We added a feature to the part-of-speech tags of verbs in order to distinguish between be, do, have, and full verbs.

Agreement feature Finite VPs are marked with 3s (n3s) if they are headed by a verb with part-of-speech VBZ (VBP).

Genitive feature NP nodes which dominate a node of the category POS (possessive marker) are marked with a genitive flag.

Base NPs NPs dominating a node of category NN, NNS, NNP, NNPS, DT, CD, JJ, JJR, JJS, PRP, RB, or EX are marked as base NPs.

![Figure 2: Slash features: The filler node of category WHNP is linked to the trace node via percolation of a slash feature. The trace node is labeled with *T*.](http://www.ims.uni-stuttgart.de/schmid)
IN feature  The part-of-speech tags of the 45 most frequent prepositions were lexicalized by adding the preposition as a feature. The new part-of-speech tag of the preposition “by” is “IN/by”.

Irregular adverbs  The part-of-speech tags of the adverbs “as”, “so”, “about”, and “not” were also lexicalized.

Currency feature  NP and QP nodes are marked with a currency flag if they dominate a node of category $, #, or SYM.

Percent feature  Nodes of the category NP or QP are marked with a percent flag if they dominate the subtree (NN %). Any node which immediately dominates the token %, is marked, as well.

Punctuation feature  Nodes which dominate sentential punctuation (,?!) are marked.

DT feature  Nodes of category DT are split into indefinite articles (a, an), definite articles (the), and demonstratives (this, that, those, these).

WH feature  The wh-tags (WDT, WP, WRB, WDT) of the words which, what, who, how, and that are also lexicalized.

Colon feature  The part-of-speech tag ‘:’ was replaced with ‘;’, ‘–’ or ‘...’ if it dominated a corresponding token.

DomV feature  Nodes of a non-verbal syntactic category are marked with a feature if they dominate a node of category VP, SINV, S, SQ, SBAR, or SBARQ.

Gap feature  S nodes dominating an empty NP are marked with the feature gap.

Subcategorization feature  The part-of-speech tags of verbs are annotated with a feature which encodes the sequence of arguments. The encoding maps reflexive NPs to r, NP/NP-PRD/SBAR-NOM to n, ADJP-PRD to j, ADVP-PRD to a, PRT to t, PP/PP-DIR to p, SBAR/SBAR-CLR to b, S/fin to sf, S/ppres/gap to sg, S/to/gap to st, other S nodes to so, VP/ppres to vg, VP/ppast to vn, VP/pas to vp, VP/inf to vi, and other VPs to vo. A verb with an NP and a PP argument, for instance, is annotated with the feature np.

Adjectives, adverbs, and nouns may also get a subcat feature which encodes a single argument using a less fine-grained encoding which maps PP to p, NP to n, S to s, and SBAR to b. A node of category NN or NNS e.g. is marked with a subcat feature if it is followed by an argument category unless the argument is a PP which is headed by the preposition of.

RC feature  In relative clauses with an empty relative pronoun of category WHADVP, we mark the SBAR node of the relative clause, the NP node to which it is attached, and its head child of category NN or NNS, if the head word is either way, ways, reason, reasons, day, days, time, moment, place, or position. This feature helps the parser to correctly insert WHADVP rather than WHNP. Figure 4 shows a sample tree.

TMP features  Each node on the path between an NP-TMP or PP-TMP node and its nominal head is labeled with the feature tmp. This feature helps the parser to identify temporal NPs and PP.

MNR and EXT features  Similarly, each node on the path between an NP-EXT, NP-MNR or ADVP-TMP node and its head is labeled with the
Figure 4: Annotation of relative clauses with empty relative pronoun of category WHADVP

feature ext or mnr.

ADJP features Nodes of category ADJP which are dominated by an NP node are labeled with the feature “post” if they are in final position and the feature “attr” otherwise.

JJ feature Nodes of category JJ which are dominated by an ADJP-PRD node are labeled with the feature “prd”.

JJ-tmp feature JJ nodes which are dominated by an NP-TMP node and which themselves dominate one of the words “last”, “next”, “late”, “previous”, “early”, or “past” are labeled with tmp.

QP feature If some node dominates an NP node followed by an NP-ADV node as in (NP (NP one dollar) (NP-ADV a day)), the first child NP node is labeled with the feature “qp”. If the parent is an NP node, it is also labeled with “qp”.

NP-pp feature NP nodes which dominate a PP node are labeled with the feature pp. If this PP itself is headed by the preposition of, then it is annotated with the feature of.

MWL feature In adverbial phrases which neither dominate an adverb nor another adverbial phrase, we lexicalize the part-of-speech tags of a small set of words like “least” (at least), “kind”, or “sort” which appear frequently in such adverbial phrases.

Case feature Pronouns like he or him, but not ambiguous pronouns like it are marked with nom or acc, respectively.

Expletives If a subject NP dominates an NP which consists of the pronoun it, and an S-trace in sentences like It is important to..., the dominated NP is marked with the feature expl.

LST feature The parent nodes of LST nodes\(^2\) are marked with the feature lst.

Complex conjunctions In SBAR constituents starting with an IN and an NN child node (usually indicating one of the two complex conjunctions “in order to” or “in case of”), we mark the NN child with the feature sbar.

LGS feature The PENN treebank marks the logical subject of passive clauses which are realized by a by-PP with the semantic tag LGS. We move this tag to the dominating PP.

OC feature Verbs are marked with an object control feature if they have an NP argument which dominates an NP filler and an S argument which dominates an NP trace. An example is the sentence She asked him to come.

Corrections The part-of-speech tags of the PENN treebank are not always correct. Some of the errors (like the tag NNS in VP-initial position) can be identified and corrected automatically in the training data. Correcting tags did not always improve parsing accuracy, so it was done selectively.

The gap and domV features described above were also used by Klein and Manning (2003).

All features were automatically added to the PENN treebank by means of an annotation program. Figure 5 shows an example of an annotated parse tree.

3 Parameter Smoothing

We extracted the grammar from sections 2–21 of the annotated version of the PENN treebank. In order to increase the coverage of the grammar, we selectively applied markovization to the grammar (cf. Klein and Manning (2003)) by replacing long infrequent rules with a set of binary rules. Markovization was only applied if none of the non-terminals on the right hand side of the rule had a slash feature in order to avoid negative effects on the slash feature percolation mechanism.

The probabilities of the grammar rules were directly estimated with relative frequencies. No smoothing was applied, here. The lexical probabilities, on the other hand, were smoothed with...
the following technique which was adopted from Klein and Manning (2003). Each word is assigned to one of 216 word classes. The word classes are defined with regular expressions. Examples are the class \[A-Za-z0-9\-]+-old\] which contains the word 20-year-old, the class \[a-z\][a-z]+ifies\] which contains clarifies, and a class which contains a list of capitalized adjectives like Advanced. The word classes are ordered. If a string is matched by the regular expressions of more than one word class, then it is assigned to the first of these word classes. For each word class, we compute part-of-speech probabilities with relative frequencies. The part-of-speech frequencies of a word are smoothed by adding the part-of-speech probability of the word class according to equation 1 in order to obtain the smoothed frequency. The part-of-speech probability of the word class is weighted by a parameter whose value was set to 4 after testing on held-out data. The lexical probabilities are finally estimated from the smoothed frequencies according to equation 2.

\[
\hat{f}(w, t) = f(w, t) + \Theta p(t)[w] \tag{1}
\]

\[
p(w|t) = \frac{\hat{f}(w, t)}{\sum_{w'} \hat{f}(w', t)} \tag{2}
\]

4 Evaluation

In our experiments, we used the usual splitting of the PENN treebank into training data (sections 2–21), held-out data (section 22), and test data (section 23).

The grammar extracted from the automatically annotated version of the training corpus contained 52,297 rules with 3,453 different non-terminals. Subtrees which dominated only empty categories were collapsed into a single empty element symbol. The parser skips over these symbols during parsing, but adds them to the output parse. Overall, there were 308 different empty element symbols in the grammar.

Parsing section 23 took 169 minutes on a Dual-Opteron system with 2.2 GHz CPUs, which is about 4.2 seconds per sentence.

Table 1 shows the labeled bracketing accuracy on section 23

|            | precision | recall | f-score |
|------------|-----------|--------|---------|
| this paper | 86.9      | 86.3   | 86.6    |
| Klein/Manning | 86.3   | 85.1   | 85.7    |

Table 1: Labeled bracketing accuracy on section 23

4.1 Empty Category Prediction

Table 2 reports the accuracy of the parser in the empty category (EC) prediction task for ECs occurring more than 6 times. Following Johnson (2001), an empty category was considered correct if the treebank parse contained an empty node of the same category at the same string position. Empty SBAR nodes which dominate an empty S node are treated as a single empty element and listed as SBAR-S in table 2.

Frequent types of empty elements are recognized quite reliably. Exceptions are the traces of adverbial and prepositional phrases where the recall was only 65% and 48%, respectively, and empty relative pronouns of type WHNP and WHADVP with f-scores around 60%. A couple of empty relative pronouns of type WHADVP were mis-analyzed as WHNP which explains why the precision is higher than the recall for WHADVP, but vice versa for WHNP.
The accuracy of the pseudo attachment labels *RNR*, *ICH*, *EXP*, and *PPA* was generally low with a precision of 41%, recall of 21%, and f-score of 28%. Empty elements with a test corpus frequency below 8 were almost never generated by the parser.

### 4.2 Co-Indexation

Table 3 shows the accuracy of the parser on the co-indexation task. A co-indexation of a trace and a filler is represented by a 5-tuple consisting of the category and the string position of the trace, as well as the category, start and end position of the filler. A co-indexation is judged correct if the treebank parse contains the same 5-tuple.

For NP\(^3\) and S\(^4\) traces of type ‘*T*’, the co-indexation results are quite good with 85% and 92% f-score, respectively. For ‘*T*’-traces of other categories and for NP traces of type ‘*’,\(^5\) the parser shows high precision, but moderate recall. The recall of infrequent types of empty elements is again low, as in the recognition task.

### 4.3 Feature Evaluation

We ran a series of evaluations on held-out data in order to determine the impact of the different features which we described in section 2 on the parsing accuracy. In each run, we deleted one of the features and measured how the accuracy changed compared to the baseline system with all features. The results are shown in table 4.

---

\(^3\)NP traces of type ‘*T*’ result from wh-extraction in questions and relative clauses and from fronting.

\(^4\)S traces of type ‘*T*’ occur in sentences with quoted speech like the sentence “That’s true!”, he said ‘*T*.”

\(^5\)The trace type ‘*’ combines two types of traces with different linguistic properties, namely empty objects of passive constructions which are co-indexed with the subject, and empty subjects of participial and infinitive clauses which are co-indexed with an NP of the matrix clause.
Table 4: Differences between the baseline f-scores for labeled bracketing, EC prediction, and co-indexation (CI) and the f-scores without the specified feature.

| Feature                | LB       | EC       | CI       |
|------------------------|----------|----------|----------|
| slash feature          | 0.43     | –        | –        |
| VP features            | 2.93     | 6.38     | 5.46     |
| PENN tags              | 2.34     | 4.54     | 6.75     |
| IN feature             | 2.02     | 2.57     | 5.63     |
| S features             | 0.49     | 3.08     | 4.13     |
| V subcat feature       | 0.68     | 3.17     | 2.94     |
| punctuation feat.      | 0.82     | 1.11     | 1.86     |
| all PENN tags          | 0.84     | 0.69     | 2.03     |
| domV feature           | 1.76     | 0.15     | 0.00     |
| gap feature            | 0.04     | 1.20     | 1.32     |
| DT feature             | 0.57     | 0.44     | 0.99     |
| RC feature             | 0.00     | 1.11     | 1.10     |
| colon feature          | 0.41     | 0.84     | 0.44     |
| ADV parent             | 0.50     | 0.04     | 0.93     |
| auxiliary feat.        | 0.40     | 0.29     | 0.77     |
| SBAR parent            | 0.45     | 0.24     | 0.71     |
| agreement feat.        | 0.05     | 0.52     | 1.15     |
| ADVP subcat feat.      | 0.33     | 0.32     | 0.55     |
| genitive feat.         | 0.39     | 0.29     | 0.44     |
| NP subcat feat.        | 0.33     | 0.08     | 0.76     |
| no-temp                | 0.14     | 0.90     | 0.16     |
| base NP feat.          | 0.47     | -0.24    | 0.55     |
| tag correction         | 0.13     | 0.37     | 0.44     |
| irr. adverb feat.      | 0.04     | 0.56     | 0.39     |
| PP parent              | 0.08     | 0.04     | 0.82     |
| ADJP features          | 0.14     | 0.41     | 0.33     |
| currency feat.         | 0.06     | 0.82     | 0.00     |
| qp feature             | 0.13     | 0.14     | 0.50     |
| PP tmp feature         | -0.24    | 0.65     | 0.60     |
| WH feature             | 0.11     | 0.25     | 0.27     |
| percent feat.          | 0.34     | -0.10    | 0.10     |
| NP-ADV parent f.       | 0.07     | 0.14     | 0.39     |
| MNR feature            | 0.08     | 0.35     | 0.11     |
| JJ feature             | 0.08     | 0.18     | 0.27     |
| case feature           | 0.05     | 0.14     | 0.27     |
| Expletive feat.        | -0.01    | 0.16     | 0.27     |
| LGS feature            | 0.17     | 0.07     | 0.00     |
| ADJ subcat             | 0.00     | 0.00     | 0.33     |
| OC feature             | 0.00     | 0.00     | 0.22     |
| JJ-temp feat.          | 0.09     | 0.00     | 0.00     |
| refl. pronoun          | 0.02     | -0.03    | 0.16     |
| EXT feature            | -0.04    | 0.09     | 0.16     |
| MWL feature            | 0.05     | 0.00     | 0.00     |
| complex conj. f.       | 0.07     | -0.07    | 0.00     |
| LST feature            | 0.12     | -0.12    | -0.11    |
| NP-pp feature          | 0.13     | -0.57    | -0.39    |

5 Comparison

Table 7 compares the empty category prediction results of our parser with those reported in Johnson (2001), Dienes and Dubey (2003b) and Campbell (2004). In terms of recall and f-score, our parser outperforms the other parsers. In terms of precision, the tagger of Dienes and Dubey is the best, but its recall is the lowest of all systems.

|          | prec. | recall | f-score |
|----------|-------|--------|---------|
| this paper | 86.0  | 82.3   | 84.1    |
| Campbell  | 85.2  | 81.7   | 83.4    |
| Dienes & Dubey | 86.5 | 72.9   | 79.1    |
| Johnson   | 85    | 74     | 79      |

Table 5: Accuracy of empty category prediction on section 23

The good performance of our parser on the empty element recognition task is remarkable considering the fact that its performance on the labeled bracketing task is 3% lower than that of the Charniak (2000) parser used by Campbell (2004).

|          | prec. | recall | f-score |
|----------|-------|--------|---------|
| this paper | 81.7  | 73.5   | 77.4    |
| Campbell  | 78.3  | 75.1   | 76.7    |
| Dienes & Dubey (b) | 81.5 | 68.7   | 74.6    |
| Dienes & Dubey (a) | 80.5 | 66.0   | 72.6    |
| Johnson   | 73    | 63     | 68      |

Table 6: Co-indexation accuracy on section 23

Table 6 compares our co-indexation results with those reported in Johnson (2001), Dienes and Dubey (2003b), Dienes and Dubey (2003a), and Campbell (2004). Our parser achieves the highest precision and f-score. Campbell (2004) reports a higher recall, but lower precision.

Table 7 shows the trace prediction accuracies of our parser, Johnson’s (2001) parser with parser input and perfect input, and Campbell’s (2004) parser with perfect input. The accuracy of Johnson’s parser is consistently lower than that of the other parsers and it has particular difficulties with ADVP traces, SBAR traces, and empty relative pronouns (WHNP 0). Campbell’s parser and our parser cannot be directly compared, but when we take the respective performance difference to Johnson’s parser as evidence, we might conclude that Campbell’s parser works particularly well on NP *, *U*, and WHNP 0, whereas our system
|       | paper | J1 | J2 | C   |
|-------|-------|----|----|-----|
| NP *  | 83.2  | 82 | 91 | 97.5|
| NP *T*| 86.2  | 81 | 91 | 96.2|
| 0     | 92.3  | 88 | 96 | 98.5|
| *U*   | 94.5  | 92 | 95 | 98.6|
| ADVP *T* | 71.7 | 56 | 66 | 79.9|
| S *T* | 90.1  | 88 | 90 | 92.7|
| SBAR-S *T* | 82.1 | 70 | 74 | 84.4|
| WHNP 0 | 60.4  | 47 | 77 | 92.4|
| WHADVP 0 | 60.0 | –  | –  | 73.3|

Table 7: Comparison of the empty category prediction accuracies for different categories in this paper (paper), in (Johnson, 2001) with parser input (J1), in (Johnson, 2001) with perfect input (J2), and in (Campbell, 2004) with perfect input.

is slightly better on empty complementizers (0), ADVP traces, and SBAR traces.

6 Summary

We presented an unlexicalized PCFG parser which applies a slash feature percolation mechanism to generate parse trees with empty elements and co-indexation of traces and fillers. The grammar was extracted from a version of the PENN treebank which was annotated with slash features and a set of other features that were added in order to improve the general parsing accuracy. The parser computes true Viterbi parses unlike most other parsers for treebank grammars which are not guaranteed to produce the most likely parse tree because they apply pruning strategies like beam search.

We evaluated the parser using the standard PENN treebank training and test data. The labeled bracketing f-score of 86.6% is – to our knowledge – the best f-score reported for unlexicalized PCFGs, exceeding that of Klein and Manning (2003) by almost 1%. On the empty category prediction task, our parser outperforms the best previously reported system (Campbell, 2004) by 0.7% reaching an f-score of 84.1%, although the general parsing accuracy of our unlexicalized parser is 3% lower than that of the parser used by Campbell (2004). Our parser also ranks highest in terms of the co-indexation accuracy with 77.4% f-score, again outperforming the system of Campbell (2004) by 0.7%.

References

Richard Campbell. 2004. Using linguistic principles to recover empty categories. In Proceedings of the 42nd Annual Meeting of the ACL, pages 645–652, Barcelona, Spain.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In Proceedings of the 1st Meeting of the North American Chapter of the Association for Computational Linguistics (ANLP-NAACL 2000), pages 132–139, Seattle, Washington.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In Proceedings of the 35th Annual Meeting of the ACL, Madrid, Spain.

Péter Dienes and Amit Dubey. 2003a. Antecedent recovery: Experiments with a trace tagger. In Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing, Sapporo, Japan.

Péter Dienes and Amit Dubey. 2003b. Deep syntactic processing by combining shallow methods. In Proceedings of the 41st Annual Meeting of the ACL, pages 431–438, Sapporo, Japan.

Mark Johnson. 1998. PCFG models of linguistic tree representations. Computational Linguistics, 24(4):613–632.

Mark Johnson. 2001. A simple pattern-matching algorithm for recovering empty nodes and their antecedents. In Proceedings of the 39th Annual Meeting of the ACL, pages 136–143, Toulouse, France.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In Proceedings of the 41st Annual Meeting of the ACL, pages 423–430, Sapporo, Japan.

Roger Levy and Christopher D. Manning. 2004. Deep dependencies from context-free statistical parsers: Correcting the surface dependency approximation. In Proceedings of the 42nd Annual Meeting of the ACL, pages 327–334, Barcelona, Spain.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. Computational Linguistics, 19(2):313–330, June.

Helmut Schmid. 2004. Efficient parsing of highly ambiguous context-free grammars with bit vectors. In Proceedings of the 20th International Conference on Computational Linguistics (COLING 2004), volume 1, pages 162–168, Geneva, Switzerland.