A Cascaded Finite-State Parser for German

Michael Schiehlen
Institute for Computational Linguistics, University of Stuttgart,
Azenbergstr. 12, D-70174 Stuttgart
mike@adler.ims.uni-stuttgart.de

Abstract

The paper presents two approaches to partial parsing of German: a tagger trained on dependency tuples, and a cascaded finite-state parser (Abney, 1997). For the tagging approach, the effects of choosing different representations of dependency tuples are investigated. Performance of the finite-state parser is boosted by delaying syntactically unsolvable disambiguation problems via underspecification. Both approaches are evaluated on a 340,000-token corpus.

1 Introduction

Traditional parsers are often quite brittle, and optimize precision over recall. It is therefore important to also look at shallow approaches that come at virtually no cost in manual labour but can potentially supplement more knowledge-prone approaches. The paper discusses one such approach which gets by with a tree bank and a tagger. Another issue in parsing is speed, which can only be gained by deterministic processing. Deterministic parsers return exactly one syntactic reading, which forces them to solve many locally unsolvable puzzles. Abney (1997) suggests a way out of this dilemma: The parser leaves ambiguities unresolved if they are contained in a local domain. So at least ambiguities of this kind can conceivably be handed over to some expert disambiguation module. The paper fleshes out this idea and shows its impact on overall performance.

2 Evaluation Method

Instead of using the prevalent PARSEVAL measures, we opted for a dependency-based evaluation (Lin, 1995), which is arguably (Srinivas et al., 1996) (Kübler and Telljohann, 2002) fairer to partial parsers. In a dependency structure, every word token (dependent) is related to another token (head) over a grammatical role, but for one word token, which is called the root. Thus, a parser constructing a dependency structure needs to associate every word token either with a head token plus grammatical role or mark it as the root or 'TOP' node. The task can be seen as a classification problem and measured in (*labelled*) precision and recall. To simplify the task, grammatical roles can be neglected (*unlabelled* precision and recall).

The details deserve some attention. With Kübler and Telljohann (2002) and in contrast to Lin (1995), we assume that PPs are headed by their internal NPs, and that conjoined phrases have multiple heads (the conjuncts), with the conjunction linked to the last conjunct. Carroll et al. (1998) introduce additional links for control phenomena, map several tokens to one node (e.g. linked preposition–noun and determiner–noun pairs), and allow nodes for elided words (e.g. in pro-/topic-drop and gapping). An important objection is that the weight of words is determined quite arbitrarily (Clark and Hockenmaier, 2002). Thus, we adopt Lin's scheme with the above provisos.

Training and test sets for the experiments described below were derived from a tokenized version of the Negra tree bank of German newspaper

texts (Skut et al., 1997), comprising ca. 340,000 tokens in 19,547 sentences. Different tagging qualities were taken into account by alternatively using Part-Of-Speech tags determined by the Tree Tagger (Schmid, 1994) (tagger tags), POS tags determined by the Tree Tagger trained on the tree bank (lexicon tags), or the POS tags of the tree bank (ideal tags). All experiments were run on a SUN Blade-1000.

3 Tagging Approach

The head tokens in dependency tuples can be coded in several ways. The **position** method represents a head token by its position in the sentence (pos_{head}). On the Negra tree bank, this method yields 121 unlabelled and 1810 labelled[1] classes. The **distance** method codes a head token by giving the distance to the dependent (pos_{head}-pos_{dep}), yielding 123 unlabelled but only 1139 labelled classes. Lin (1995) represents the head token by its word type and a position indicator which encodes the direction where the head can be found and the number of tokens of identical type between head and dependent (e.g. < first token with same word type on the left, >>> third token with same word type on the right, etc.). To get fewer classes, we use the category[2] of the head token instead of its word type. The resulting method (which we will call **nth-tag** method) yields 115 unlabelled and 639 labelled classes.

For the experiment, the trigram-based Tree Tagger was used to map tokens directly to the dependency classes (see for a similar approach (Srinivas et al., 1996)). Performance was degraded when the tagger got information on both word type and POS tag of the tokens, so we only used POS tag. We didn't test the **position** method. Figure 1 shows results achieved via 10-fold cross-validation with Ideal and Tagger tags. The tagger always gives a unique answer, but head tokens not found in the string count as not assigned, hence the discrepancy between precision and recall. A figure is also given for the percentage of sentences getting a completely correct parse.

[1]NEGRA distinguishes 33 grammatical roles.

[2]Better performance is achieved when only the category information in the POS tag is used, but not Verb Form, or distinction between common and proper nouns.

	labelled			unlabelled			cor-
	prec	rec	speed	prec	rec	speed	rect
I dist	63.07	60.06	3.41	62.69	61.10	19.80	4.63
I nth	73.86	67.92	14.03	72.02	64.83	122.45	5.22
T dist	61.00	58.08	2.48	61.32	59.88	26.62	3.97
T nth	71.04	64.93	10.67	70.13	63.04	77.33	4.39

Figure 1: Evaluation Results for Tagger

Processing speed is measured in Words Per Second. We also combined the distance and nth-tag method by using a greedy method to choose between them on the basis of the POS tag of the token and the proposed result. This hybrid method achieved 80.99%/75.82% labelled precision recall on Ideal tags and 78.02%/72.83% on Tagger tags.

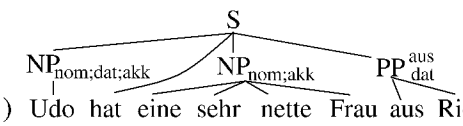
4 Cascaded Finite-State Parser

4.1 Description of the Parser

The parser described here essentially relies on techniques also used by Abney (1997). It basically consists of a noun chunker and a clause chunker.

The noun chunker is deterministic, but recognizes recursive noun chunks in several passes. Morphological information on case, number and gender coded is computed with bit vectors (Abney, 1997). A noun chunk is defined as an NP or PP with all adjuncts at the beginning (e.g. adverbs) and at the end (e.g. PPs and relative clauses) stripped off (Brants, 1999) (Schiehlen, 2002).

The clause chunker consists of three deterministic transducers recognizing verb-final, verb-first, and verb-second clauses. The parser aims to determine full clauses rather than the "simplex clauses" of Abney (1997) (i.e. non-recursive "core" parts of clauses). The verb-final clause transducer e.g. works from right to left so that subclauses are maximally embedded. Example (1) shows chunker output (a flat parse tree) after the recognition phase.

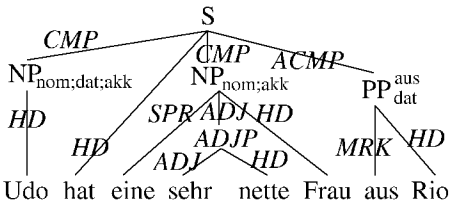


(1) Udo hat eine sehr nette Frau aus Rio .
Udo has a very nice wife from Rio.
or: A very nice woman has Udo from Rio.
or: A very nice woman from Rio has Udo.

An interpretation step follows, where non-deterministic transducers insert further syntactic structure (e.g. adjective phrases, phrases for coordinated VPs and prepositions) and grammatical roles[3]. The pertinent information is coded in the finite-state grammars although it is not seen by the recognition transducers. See below a rule in the grammar, semicolon symbols are only needed for interpretation.

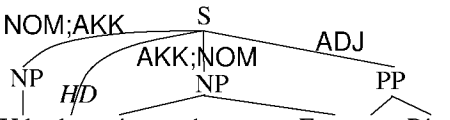
(2) det ;SPR (;[ADJP (adv ;ADJ)* adja ;HD ;]ADJP)* nn ;HD FINAL:NP

Verb government and verb complexes can only be computed after coordinated VPs have been inserted, since auxiliaries may distribute. Example (3) shows the parse tree after interpretation.



(3) Udo hat eine sehr nette Frau aus Rio .

Finally, a deterministic transducer recognizes subcategorization frames using a grammar automatically constructed from lexically specified frames and introduces a fine-grained differentiation of the complement relation (61 additional grammatical roles). See example (4) for output.



(4) Udo hat eine sehr nette Frau aus Rio .

If the frame transducer fails, an unspecified grammatical role is left (Carroll et al., 1998). Such roles are counted as correct only in a set of figures that we shall call *half-labelled* precision and recall.

4.2 Explicit Underspecification

An apparent drawback of deterministic parsers is the need for forced guessing, i.e. the need to make decisions without access to the requisite disambiguating knowledge. Cases in point are PP attachment and (sometimes) determination of case in German (cf. example (4)). In context-free parsing, the solution to this problem is conservation of ambiguities in the output: Difficult decisions are delayed to a later stage. Similar techniques can be used in finite-state parsing (Elworthy et al., 2001).

Underspecification can be elegantly implemented with context variables (Maxwell III and Kaplan, 1989) (Dörre, 1997). Since subcategorization ambiguities are specific to main verbs in clauses and never interact across clause boundaries, the clause nodes themselves can be interpreted as context variables. The different options are implicitly encoded by bringing the varying grammatical roles of a constituent node in a clause-wide uniform order (e.g. in example (4) position 1: first NP nominative, second NP accusative; position 2: first NP accusative, second NP nominative). VP coordination sometimes gives rise to structures with constituents figuring in several subcategorization frames at once. In this case several lists of grammatical roles are associated with the constituent, one for each conjunct in left-to-right order (cf. example (5)).

(5) Hans(⟨N;A⟩⟨N;D⟩) [[kennt Maria(⟨A;N⟩)] und [hilft Karla(⟨D;N⟩)]].
Hans knows Maria and helps Karla.
or: Maria knows and Karla helps Hans.
or: Maria knows Hans and he helps Karla.
or: Hans knows Maria and Karla helps him.

In a final processing step, the constituent trees are converted into dependency tuples. In this step, attachment and subcategorization ambiguities are overtly represented with context variables, cf. (6):

(6)	Udo/0	hat/1	[1a]:NPnom,[1b]:NPakk
	hat/1	TOP	
	eine/2	Frau/5	SPR
	sehr/3	nette/4	ADJ
	nette/4	Frau/5	ADJ
	Frau/5	hat/1	[1a]:NPakk,[1b]:NPnom
	aus/6	Rio/7	MRK
	Rio/7	hat/1	ADJ [1A0]
		Frau/5	ADJ [1A1]
	./8	TOP	

Riezler et al. (2002) evaluate underspecified syntactic representations by distinguishing lower bound performance (random choice of a parse)

[3]There are 13 grammatical roles: head, adjunct, apposition, complement, adjunct or complement, conjunction, first part of conjunction, measure phrase, marker, specifier, subject, governed verb, unconnected.

and upper bound performance (selection of the best parse according to the test set).

4.3 Evaluation Results

Currently, the speed of the finite-state parser is at 2430 Words Per Second, but this figure can still be improved by compiling the backtracking necessitated in Abney's (1997) approach into the transition tables. See Figure 2 for results of parsing

	labelled		half-lab.		unlab.		cor-
	prec	rec	prec	rec	prec	rec	rect
I lower	82.9	76.1	85.0	77.9	89.6	82.2	12.5
I upper	90.9	83.4	92.9	85.2	95.0	87.2	39.0
L lower	82.3	74.7	84.2	76.5	89.3	81.1	11.7
L upper	90.3	82.0	92.2	83.7	94.7	85.9	36.5
T lower	81.5	71.3	83.5	73.0	88.6	77.5	10.6
T upper	88.9	77.7	90.9	79.5	93.6	81.8	31.0

Figure 2: Results for Finite-State Parser

on Ideal, Lexicon and Tagger tags. The last column of the table shows the percentage of completely correct analyses of sentences. For the lower bound, only unambiguous sentence analyses count as correct. When we combined chunker and tagger results using the greedy method, performance was boosted to 94.48%/87.28% labelled precision/recall on ideal tags (upper-bound) and 94.36%/86.35% (lower-bound). These figures can be compared with the values reported by Neumann et al. (2000) (precision 89.68%, recall 84.75%) although they used a much smaller corpus for evaluation (10,400 tokens) which was not annotated independently.

5 Conclusion

The paper presents a cascaded finite-state parser incorporating some degree of underspecification. The idea is that such syntactically unresolvable ambiguities are later resolved by expert disambiguation modules. The performance of the finite-state parser has been compared with a very simple tagging approach which nevertheless gets more than 50% of the dependency structure correct. I am grateful to Helmut Schmid for discussion and to the reviewers for hints on literature.

References

Steven Abney. 1997. Partial Parsing via Finite-State Cascades. *Journal of Natural Language Engineering*, 2(4):337–344.

Thorsten Brants. 1999. Cascaded Markov Models. In *Proceedings of EACL'99*, Bergen, Norway.

John Carroll, Ted Briscoe, and Antonio Sanfilippo. 1998. Parser Evaluation: a Survey and a New Proposal. In *Proceedings of LREC*, pages 447–454, Granada.

Stephen Clark and Julia Hockenmaier. 2002. Evaluating a Wide-Coverage CCG Parser. In *Beyond PARSEVAL – Twoards Improved Evaluation Measures for Parsign Systems (LREC Workshop)*.

Jochen Dörre. 1997. Efficient Construction of Underspecified Semantics under Massive Ambiguity. In *Proceedings of ACL'97*, pages 386–393, Madrid, Spain.

David Elworthy, Tony Rose, Amanda Clare, and Aaron Kotcheff. 2001. A natural language system for retrieval of captioned images. *Journal of Natural Language Engineering*, 7(2):117–142.

Sandra Kübler and Heike Telljohann. 2002. Towards a Dependency-Oriented Evaluation for Partial Parsing. In *Beyond PARSEVAL – Towards Improved Evaluation Measures for Parsing Systems (LREC Workshop)*.

Dekang Lin. 1995. A Dependency-based Method for Evaluating Broad-Coverage Parsers. In *Proceedings of the IJCAI-95*, pages 1420–1425, Montreal.

John T. Maxwell III and Ronald M. Kaplan. 1989. An overview of disjunctive constraint satisfaction. In *Proceedings of the International Workshop on Parsing Technologies*, Pittsburgh, PA.

Günter Neumann, Christian Braun, and Jakub Piskorski. 2000. A Divide-and-Conquer Strategy for Shallow Parsing of German Free Text. In *Proceedings of ANLP'00*, pages 239–246, Seattle, WA.

Stefan Riezler, Tracy H. King, Ronald M. Kaplan, Richard Crouch, John T. Maxwell III, and Mark Johnson. 2002. Parsing the Wall Street Journal using a Lexical-Functional Grammar and Discriminative Estimation Techniques. In *Proceedings of ACL'02*.

Michael Schiehlen. 2002. Experiments in German Noun Chunking. In *Proceedings of COLING '02*, Taipei.

Helmut Schmid. 1994. Probabilistic Part-Of-Speech Tagging Using Decision Trees. Technical report, Institut für maschinelle Sprachverarbeitung, Universität Stuttgart.

Wojciech Skut, Brigitte Krenn, Thorsten Brants, and Hans Uszkoreit. 1997. An Annotation Scheme for Free Word Order Languages. In *Proceedings of the ANLP-97*, Washington, DC.

B. Srinivas, Christine Doran, Beth Ann Hockey, and Aravind Joshi. 1996. An approach to Robust Partial Parsing and Evaluation Metrics. In *Proceedings of the ESSLLI'96 Workshop on Robust Parsing*, pages 70–82, Prague.