Combining Deep and Shallow Approaches in Parsing German

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

Abstract

The paper describes two parsing schemes: a shallow approach based on machine learning and a cascaded finite-state parser with a hand-crafted grammar. It discusses several ways to combine them and presents evaluation results for the two individual approaches and their combination. An underspecification scheme for the output of the finite-state parser is introduced and shown to improve performance.

1 Introduction

In several areas of Natural Language Processing, a *combination* of different approaches has been found to give the best results. It is especially rewarding to combine deep and shallow systems, where the former guarantees interpretability and high precision and the latter provides robustness and high recall. This paper investigates such a combination consisting of an n-gram based shallow parser and a cascaded finite-state parser[1] with hand-crafted grammar and morphological checking. The respective strengths and weaknesses of these approaches are brought to light in an in-depth evaluation on a treebank of German newspaper texts (Skut et al., 1997) containing ca. 340,000 tokens in 19,546 sentences. The evaluation format chosen (dependency tuples) is used as the common denominator of the systems in building a hybrid parser with improved performance. An underspecification scheme allows the finite-state parser partially ambiguous output. It is shown that the other parser can in most cases successfully disambiguate such information.

Section 2 discusses the evaluation format adopted (dependency structures), its advantages, but also some of its controversial points. Section 3 formulates a classification problem on the basis of the evaluation format and applies a machine learner to it. Section 4 describes the architecture of the cascaded finite-state parser and its output in a novel underspecification format. Section 5 explores several combination strategies and tests them on several variants of the two base components. Section 6 provides an in-depth evaluation of the component systems and the hybrid parser. Section 7 concludes.

2 Parser Evaluation

The simplest method to evaluate a parser is to count the parse trees it gets correct. This measure is, however, not very informative since most applications do not require one hundred percent correct parse trees. Thus, an important question in parser evaluation is how to break down parsing results.

In the PARSEVAL evaluation scheme (Black et al., 1991), partially correct parses are gauged by the number of nodes they produce and have in common with the gold standard (measured in precision and recall). Another figure (crossing brackets) only counts those incorrect nodes that change the partial order induced by the tree. A problematic aspect of the PARSEVAL approach is that the weight given to particular constructions is again grammar-specific,

[1]Although not everyone would agree that finite-state parsers constitute a 'deep' approach to parsing, they still are knowledge-based, require efforts of grammar-writing, a complex linguistic lexicon, manage without training data, etc.

since some grammars may need more nodes to describe them than others. Further, the approach does not pay sufficient heed to the fact that parsing decisions are often intricately twisted: One wrong decision may produce a whole series of other wrong decisions.

Both these problems are circumvented when parsing results are evaluated on a more abstract level, viz. *dependency structure* (Lin, 1995). Dependency structure generally follows predicate-argument structure, but departs from it in that the basic building blocks are words rather than predicates. In terms of parser evaluation, the first property guarantees independence of decisions (every link is relevant also for the interpretation level), while the second property makes for a better empirical justification. for evaluation units. Dependency structure can be modelled by a directed acylic graph, with word tokens at the nodes. In *labelled* dependency structure, the links are furthermore classified into a certain set of *grammatical roles*.

Dependency can be easily determined from constituent structure if in every phrase structure rule a constituent is singled out as the head (Gaifman, 1965). To derive a labelled dependency structure, all non-head constituents in a rule must be labelled with the grammatical role that links their head tokens to the head token of the head constituent.

There are two cases where the divergence between predicates and word tokens makes trouble: (1) predicates expressed by more than one token, and (2) predicates expressed by no token (as they occur in ellipsis). Case 1 frequently occurs within the verb complex (of both English and German). The solution proposed in the literature (Black et al., 1991; Lin, 1995; Carroll et al., 1998; Kübler and Telljohann, 2002) is to define a *normal form* for dependency structure, where every adjunct or argument attaches to some distinguished part of the verb complex. The underlying assumption is that those cases where scope decisions in the verb complex *are* semantically relevant (e.g. with modal verbs) are not resolvable in syntax anyway. There is no generally accepted solution for case 2 (ellipsis). Most authors in the evaluation literature neglect it, perhaps due to its infrequency (in the NEGRA corpus, ellipsis only occurs in 1.2% of all dependency relations). Robinson (1970, 280) proposes to promote one of the dependents (preferably an obligatory one) (1a) or even all dependents (1b) to head status.

(1) a. the very <u>brave</u>

b. John likes tea and <u>Harry</u> <u>coffee</u>.

A more sweeping solution to these problems is to abandon dependency structure at all and directly go for predicate-argument structure (Carroll et al., 1998). But as we argued above, moving to a more theoretical level is detrimental to comparability across grammatical frameworks.

3 A Direct Approach: Learning Dependency Structure

According to the dependency structure approach to evaluation, the task of the parser is to find the correct dependency structure for a string, i.e. to associate every word token with pairs of head token and grammatical role or else to designate it as independent. To make the learning task easier, the number of classes should be reduced as much as possible. For one, the task could be simplified by focusing on *unlabelled* dependency structure (measured in "unlabelled" precision and recall (Eisner, 1996; Lin, 1995)), which is, however, in general not sufficient for further semantic processing.

3.1 Tree Property

Another possibility for reduction is to associate every word with at most one pair of head token and grammatical role, i.e. to only look at dependency *trees* rather than graphs. There is one case where the tree property cannot easily be maintained: coordination. Conceptually, all the conjuncts are head constituents in coordination, since the conjunction could be missing, and selectional restrictions work on the individual conjuncts (2).

(2) John ate (fish and chips|*wish and ships).

But if another word depends on the conjoined heads (see (4a)), the tree property is violated. A way out of the dilemma is to select a specific conjunct as modification site (Lin, 1995; Kübler and Telljohann, 2002). But unless care is taken, semantically vital information is lost in the process: Example (4) shows two readings which should be distinguished in dependency structure. A comparison of the two readings shows that if either the first conjunct or the last conjunct is unconditionally selected certain readings become undistinguishable. Rather, in order to distinguish a maximum number of readings, pre-modifiers must attach to the last conjunct and post-modifiers and coordinating conjunctions to the first conjunct[2]. The fact that the modifier refers to a conjunction rather than to the conjunct is recorded in the grammatical role (by adding `c` to it).

(4) a. the [fans and supporters] of Arsenal

b. [the fans] and [supporters of Arsenal]

Other constructions contradicting the tree property are arguably better treated in the lexicon anyway (e.g. control verbs (Carroll et al., 1998)) or could be solved by enriching the repertory of grammatical roles (e.g. relative clauses with null relative pronouns could be treated by adding the dependency relation between head verb and missing element to the one between head verb and modified noun).

In a number of linguistic phenomena, dependency theorists disagree on which constituent should be chosen as the head. A case in point are PPs. Few grammars distinguish between adjunct and subcategorized PPs at the level of prepositions. In predicate-argument structure, however, the embedded NP is in one case related to the preposition, in the other to the subcategorizing verb. Accordingly, some approaches take the preposition to be the head of a PP (Robinson, 1970; Lin, 1995), others the NP (Kübler and Telljohann, 2002). Still other approaches (Tesnière, 1959; Carroll et al., 1998) conflate verb, preposition and head noun into a triple, and thus only count content words in the evaluation. For learning, the matter can be resolved empirically:

[2]Even in this setting some readings cannot be distinguished (see e.g. (3) where a conjunction of three modifiers would be retrieved). Nevertheless, the proposed scheme fails in only 0.0017% of all dependency tuples.

(3) In New York, we never meet, but in Boston.

Note that by this move we favor interpretability over projectivity, but example (4a) is non-projective from the start.

Taking prepositions as the head somewhat improves performance, so we took PPs to be headed by prepositions.

3.2 Encoding Head Tokens

Another question is how to encode the head token. The simplest method, encoding the word by its string **position**, generates a large space of classes. A more efficient approach uses the **distance** in string position between dependent and head token. Finally, Lin (1995) proposes a third type of representation: In his work, a head is described by its word type, an indication of the direction from the dependent (left or right) and the number of tokens of the same type that lie between head and dependent. An illustrative representation would be `»paper` which refers to the second nearest token *paper* to the right of the current token. Obviously there are far too many word tokens, but we can use Part-Of-Speech tags instead. Furthermore information on inflection and type of noun (proper versus common nouns) is irrelevant, which cuts down the size even more. We will call this approach **nth-tag**. A further refinement of the nth-tag approach makes use of the fact that dependency structures are acylic. Hence, only those words with the same POS tag as the head between dependent and head must be counted that do not depend directly or indirectly on the dependent. We will call this approach **covered-nth-tag**.

	pos	dist	nth-tag	cover
labelled	1,924	1,349	982	921
unlabelled	97	119	162	157

Figure 1: Number of Classes in NEGRA Treebank

Figure 1 shows the number of classes the individual approaches generate on the NEGRA Treebank. Note that the longest sentence has 115 tokens (with punctuation marks) but that punctuation marks do not enter dependency structure. The original treebank exhibits 31 non-head syntactic[3] grammatical roles. We added three roles for marker complements (CMP), specifiers (SPR), and floating quantifiers (NK+), and subtracted the roles for conjunction markers (CP) and coreference with expletive (RE).

[3]i.e. grammatical roles not merely used for tokenization

22 roles were copied to mark reference to conjunction. Thus, all in all there was a stock of 54 grammatical roles.

3.3 Experiments

We used n-grams (3-grams and 5-grams) of POS tags as context and C4.5 (Quinlan, 1993) for machine learning. All results were subjected to 10-fold cross validation.

The learning algorithm always returns a result. We counted a result as not assigned, however, if it referred to a head token outside the sentence. See Figure 2 for results[4] of the learner. The left column shows performance with POS tags from the treebank (ideal tags, I-tags), the right column values obtained with POS tags as generated automatically by a tagger with an accuracy of 95% (tagger tags, T-tags).

	I-tags			T-tags		
	F-val	prec	rec	F-val	prec	rec
dist, 3	.6071	.6222	.5928	.5902	.6045	.5765
dist, 5	.6798	.6973	.6632	.6587	.6758	.6426
nth-tag, 3	.7235	.7645	.6866	.6965	.7364	.6607
nth-tag, 5	.7716	.7961	.7486	.7440	.7682	.7213
cover, 3	.7271	.7679	.6905	.7009	.7406	.6652
cover, 5	.7753	.7992	.7528	.7487	.7724	.7264

Figure 2: Results for C4.5

The **nth-tag** head representation outperforms the **distance** representation by 10%. Considering acyclicity (**cover**) slightly improves performance, but the gain is not statistically significant (t-test with 99%). The results are quite impressive as they stand, in particular the nth-tag 5-gram version seems to achieve quite good results. It should, however, be stressed that most of the dependencies correctly determined by the n-gram methods extend over no more than 3 tokens. With the distance method, such 'short' dependencies make up 98.90% of all dependencies correctly found, with the nth-tag method still 82%, but only 79.63% with the finite-state parser (see section 4) and 78.91% in the treebank.

[4] If the learner was given a chance to correct its errors, i.e. if it could train on its training results in a second round, there was a statistically significant gain in F-value with recall rising and precision falling (e.g. F-value .7314, precision .7397, recall .7232 for nth-tag trigrams, and F-value .7763, precision .7826, recall .7700 for nth-tag 5-grams).

4 Cascaded Finite-State Parser

In addition to the learning approach, we used a cascaded finite-state parser (Schiehlen, 2003), to extract dependency structures from the text. The layout of this parser is similar to Abney's parser (Abney, 1991): First, a series of transducers extracts noun chunks on the basis of tokenized and POS-tagged text. Since center-embedding is frequent in German noun phrases, the same transducer is used several times over. It also has access to inflectional information which is vital for checking agreement and determining case for subsequent phases (see (Schiehlen, 2002) for a more thorough description). Second, a series of transducers extracts verb-final, verb-first, and verb-second clauses. In contrast to Abney, these are full clauses, not just simplex clause chunks, so that again recursion can occur. Third, the resulting parse tree is refined and decorated with grammatical roles, using non-deterministic 'interpretation' transducers (the same technique is used by Abney (1991)). Fourth, verb complexes are examined to find the head verb and auxiliary passive or raising verbs. Only then subcategorization frames can be checked on the clause elements via a non-deterministic transducer, giving them more specific grammatical roles if successful. Fifth, dependency tuples are extracted from the parse tree.

4.1 Underspecification

Some parsing decisions are known to be not resolvable by grammar. Such decisions are best handed over to subsequent modules equipped with the relevant knowledge. Thus, in chart parsing, an underspecified representation is constructed, from which all possible analyses can be easily and efficiently read off. Elworthy et al. (2001) describe a cascaded parser which underspecifies PP attachment by allowing modifiers to be linked to several heads in a dependency tree. Example (5) illustrates this scheme.

(5) I saw a man in a car on the hill.

The main drawback of this scheme is its overgeneration. In fact, it allows six readings for example (5), which only has five readings (the speaker could not have been in the car, if the man was asserted to be on the hill). A similar clause with 10 PPs at the end would receive 39,916,800 readings rather than 58,786. So a more elaborate scheme is called for, but one that is just as easy to generate.

A device that often comes in handy for underspecification are context variables (Maxwell III and Kaplan, 1989; Dörre, 1997). First let us give every sequence of prepositional phrases in every clause a specific name (e.g. 1B for the second sequence in the first clause). Now we generate the ambiguous dependency relations (like (Elworthy et al., 2001)) but label them with *context variables*. Such context variables consist of the sequence name N, a number d designating the dependent in left-to-right order (e.g. 0 for *in*, 1 for *on* in example (5)), and a number h designating the head in left-to-right (e.g. 0 for *saw*, 1 for *man*, 2 for *hill* in (5)). If the links are stored with the dependents, the number d can be left implicit. Generation of such a representation is straightforward and, in particular, does not lead to a higher class of complexity of the full system. Example (6) shows a tuple representation for the two prepositions of sentence (5).

(6) in [1A00] saw ADJ, [1A01] man ADJ
on [1A10] saw ADJ, [1A11] man ADJ, [1A12] car ADJ

In general, a dependent d can modify $d+1$ heads, viz. the heads numbered $0,\ldots,d+1$. Now we put the following constraint on resolution: A dependent d_2 can only modify a head h_2 if no previous dependent d_1 which could have attached to h_2 (i.e. $h_2 \leq d_1 + 1$) chose some head h_1 to the left of h_2 rather than h_2. The condition is formally expressed in (7). In example (6) there are only two dependents (d_1 =*in*, d_2 =*on*). If *in* attaches to *saw*, *on* cannot attach to a head between *saw* and *in*; conversely if *on* attaches to *man*, *in* cannot attach to a head before *man*. Nothing follows if *on* attaches to *car*.

(7) Constraint: $\neg(Nd_1h_1 \wedge Nd_2h_2 \wedge d_1 < d_2 \wedge h_1 < h_2 \leq d_1 + 1)$ for all PP sequences N

The cascaded parser described adopts this underspecification scheme for right modification. Left modification (see (8)) is usually not stacked so the simpler scheme of Elworthy et al. (2001) suffices.

(8) They are usually competent people.

German is a free word order language, so that subcategorization can be ambiguous. Such ambiguities should also be underspecified. Again we introduce a context variable N for every ambiguous subcategorization frame (e.g. 1 in (9)) and count the individual readings r (with letters a,b in (9)).

(9) Peter kennt Karl. (Peter knows Karl / Karl knows Peter.)

Peter	kennt	[1a] SBJ/[1b] OA
kennt	TOP	
Karl	kennt	[1a] OA/[1b] SBJ

Since subcategorization ambiguity interacts with attachment ambiguity, context variables sometimes need to be coupled: In example (10) the attachment ambiguity only occurs if the PP is read as adjunct.

(10) Karl fügte einige Gedanken zu dem Werk hinzu. (Karl added some thoughts on/to the work.)

Gedanken	fügte	[1a] OA/[1b] OA
zu	[1A0] fügte	[1a] PP:zu/[1b] ADJ
	[1A1] Gedanken	PP:zu

1A1 < 1b

4.2 Evaluation of the Underspecified Representation

In evaluating underspecified representations, Riezler et al. (2002) distinguish upper and lower bound, standing for optimal performance in disambiguation and average performance, respectively. In

	I-tags			T-tags		
	F-val	prec	rec	F-val	prec	rec
upper	.8816	.9137	.8517	.8377	.8910	.7903
direct	.8471	.8779	.8183	.8073	.8588	.7617
lower	.8266	.8567	.7986	.7895	.8398	.7449

Figure 3: Results for Cascaded Parser

Figure 3, values are also given for the performance of the parser without underspecification, i.e. always favoring maximal attachment and word order without scrambling (direct). Interestingly this method performs significantly better than average, an effect mainly due to the preference for high attachment.

5 Combining the Parsers

We considered several strategies to combine the results of the diverse parsing approaches: simple voting, weighted voting, Bayesian learning, Maximum Entropy, and greedy optimization of F-value.

Simple Voting. The result predicted by the majority of base classifiers is chosen. The finite-state parser, which may give more than one result, distributes its vote evenly on the possible readings.

Weighted Voting. In weighted voting, the result which gets the most votes is chosen, where the number of votes given to a base classifier is correlated with its performance on a training set.

Bayesian Learning. The Bayesian approach of Xu et al. (1992) chooses the most probable prediction. The probability of a prediction π is computed by the product $\Pi_c P(\pi|\pi_c)$ of the probability of π given the predictions π_c made by the individual base classifiers c. The probability $P(\pi_1|\pi_2)$ of a correct prediction π_1 given a learned prediction π_2 is approximated by relative frequency in a training set.

Maximum Entropy. Combining the results can also be seen as a classification task, with base predictions added to the original set of features. We used the Maximum Entropy approach[5] (Berger et al., 1996) as a machine learner for this task. Underspecified features were assigned multiple values.

Greedy Optimization of F-value. Another method uses a decision list of prediction–classifier pairs to choose a prediction by a classifier. The list is obtained by greedy optimization: In each step, the prediction–classifier pair whose addition results in the highest gain in F-value for the combined model on the training set is appended to the list. The algorithm terminates when F-value cannot be improved by any of the remaining candidates. A finer distinction is possible if the decision is made dependent on the POS tag as well. For greedy optimization, the predictions of the finite-state parser were classified only in grammatical roles, not head positions. We used 10-fold cross validation to determine the decision lists.

[5] More specifically, the OpenNLP implementation (http://maxent.sourceforge.net/) was used with 10 iterations and a cut-off frequency for features of 10.

	F-val	prec	rec
simple voting	.7927	.8570	.7373
weighted voting	.8113	.8177	.8050
Bayesian learning	.8463	.8509	.8417
Maximum entropy	.8594	.8653	.8537
greedy optim	.8795	.8878	.8715
greedy optim+tag	.8849	.8957	.8743

Figure 4: Combination Strategies

We tested the various combination strategies for the combination Finite-State parser (lower bound) and C4.5 5-gram nth-tag on ideal tags (results in Figure 4). Both simple and weighted voting degrade the results of the base classifiers. Greedy optimization outperforms all other strategies. Indeed it comes near the best possible choice which would give an F-score of .9089 for 5-gram nth-tag and finite-state parser (upper bound) (cf. Figure 5).

	without POS tag			with POS tag		
I-tags	F-val	prec	rec	F-val	prec	rec
upp, nth 5	.9008	.9060	.8956	.9068	.9157	.8980
low, nth 5	.8795	.8878	.8715	.8849	.8957	.8743
low, dist 5	.8730	.8973	.8499	.8841	.9083	.8612
low, nth 3	.8722	.8833	.8613	.8773	.8906	.8644
low, dist 3	.8640	.9034	.8279	.8738	.9094	.8410
dir, nth 5	.8554	.8626	.8483	.8745	.8839	.8653

Figure 5: Combinations via Optimization

Figure 5 shows results for some combinations with the greedy optimization strategy on ideal tags. All combinations listed yield an improvement of more than 1% in F-value over the base classifiers. It is striking that combination with a shallow parser does not help the Finite-State parser much in coverage (upper bound), but that it helps both in disambiguation (pushing up the lower bound to almost the level of upper bound) and robustness (remedying at least some of the errors). The benefit of underspecification is visible when lower bound and direct are compared. The nth-tag 5-gram method was the best method to combine the finite-state parser with. Even on T-tags, this combination achieved an F-score of .8520 (lower, upper: .8579, direct: .8329) without POS tag and an F-score of .8563 (lower, upper: .8642, direct: .8535) with POS tags.

6 In-Depth Evaluation

Figure 6 gives a survey of the performance of the parsing approaches relative to grammatical role. These figures are more informative than overall F-score (Preiss, 2003). The first column gives the name of the grammatical role, as explained below. The second column shows corpus frequency in percent. The third column gives the standard deviation of distance between dependent and head. The three last columns give the performance (recall) of C4.5 with distance representation and 5-grams, C4.5 with nth-tag representation and 5-grams, and the cascaded finite-state parser, respectively. For the finite-state parser, the number shows performance with optimal disambiguation (upper bound) and, if the grammatical role allows underspecification, the number for average disambiguation (lower bound) in parentheses.

Relations between function words and content words (e.g. specifier (SPR), marker complement (CMP), infinitival *zu* marker (PM)) are frequent and easy for all approaches. The cascaded parser has an edge over the learners with arguments (subject (SB), clausal (OC), accusative (OA), second accusative (OA2), genitive (OG), dative object (DA)). For all these argument roles a slight amount of ambiguity persists (as can be seen from the divergence between upper and lower bound), which is due to free word order. No ambiguity is found with reported speech (RS). The cascaded parser also performs quite well where verb complexes are concerned (separable verb prefix (SVP), governed verbs (OC), and predicative complements (PD, SP)). Another clearly discernible complex are adjuncts (modifier (MO), negation (NG), passive subject (SBP); one-place coordination (JUnctor) and discourse markers (DM); finally postnominal modifier (MNR), genitive (GR), or *von*-phrase (PG)), which all exhibit attachment ambiguities. No attachment ambiguities are attested for prenominal genitives (GL). Some types of adjunction have not yet been implemented in the cascaded parser, so that it performs badly on them (e.g. relative clauses (RC), which are usually extraposed to the right (average distance is -11.6) and thus quite difficult also for the learners; comparative constructions (CC, CM), measure phrases (AMS), floating quantifiers (NK+)). Attachment ambiguities also occur with appositions (APP, NK[6]). Notoriously difficult is coordination (attachment of conjunction to conjuncts (CD), and dependency on multiple heads (. . .c)). Vocatives (VO) are not treated in the cascaded parser. AC is the relation between parts of a circumposition.

role	freq	dev	dist	nth-t	FS-parser
MO	24.922	4.5	65.4	75.2	86.9(75.7)
SPR	14.740	1.0	97.4	98.5	99.4
CMP	13.689	2.7	83.4	93.4	98.7
SB	9.682	5.7	48.4	64.7	84.5(82.6)
TOP	7.781	0.0	47.6	46.7	49.8
OC	4.859	7.4	43.9	85.1	91.9(91.2)
OA	4.594	5.8	24.1	37.7	83.5(70.6)
MNR	3.765	2.8	76.2	73.9	89.0(48.1)
CD	2.860	4.6	67.7	74.8	77.4
GR	2.660	1.3	66.9	65.6	95.0(92.8)
APP	2.480	3.4	72.6	72.5	81.6(77.4)
PD	1.657	4.6	31.3	39.7	55.1
RC	0.899	5.8	5.5	1.6	19.1
. . .c	0.868	7.8	13.1	13.3	34.4(26.1)
SVP	0.700	5.8	29.2	96.0	97.4
DA	0.693	5.4	1.9	1.8	77.1(71.9)
NG	0.672	3.1	63.1	73.8	81.7(70.7)
PM	0.572	0.0	99.7	99.9	99.2
PG	0.381	1.5	1.9	1.4	94.9(53.2)
JU	0.304	4.6	35.8	47.3	62.1(45.5)
CC	0.285	4.4	22.3	20.9	4.0(3.1)
CM	0.227	1.4	85.8	85.8	0
GL	0.182	1.1	70.3	67.2	87.5
SBP	0.177	4.1	4.7	3.6	93.7(77.0)
AC	0.110	2.5	63.9	60.6	91.9
AMS	0.078	0.7	63.6	60.5	1.5(0.9)
RS	0.076	8.9	0	0	25.0
NK	0.020	3.4	0	0	46.2(40.4)
OG	0.019	4.5	0	0	57.4(54.3)
DM	0.017	3.1	9.1	18.2	63.6(59.1)
NK+	0.013	3.3	16.1	16.1	0
VO	0.010	3.2	50.0	25.0	0
OA2	0.005	5.7	0	0	33.3(29.2)
SP	0.004	7.0	0	0	55.6(33.3)

Figure 6: Grammatical Roles

[6]Other relations classified as NK in the original treebank have been reclassified: prenominal determiners to SPR, prenominal adjective phrases to MO.

7 Conclusion

The paper has presented two approaches to German parsing (n-gram based machine learning and cascaded finite-state parsing), and evaluated them on the basis of a large amount of data. A new representation format has been introduced that allows underspecification of select types of syntactic ambiguity (attachment and subcategorization) even in the absence of a full-fledged chart. Several methods have been discussed for combining the two approaches. It has been shown that while combination with the shallow approach can only marginally improve performance of the cascaded parser if ideal disambiguation is assumed, a quite substantial rise is registered in situations closer to the real world where POS tagging is deficient and resolution of attachment and subcategorization ambiguities less than perfect.

In ongoing work, we look at integrating a statistic context-free parser called BitPar, which was written by Helmut Schmid and achieves .816 F-score on NEGRA. Interestingly, the performance goes up to .9474 F-score when BitPar is combined with the FS parser (upper bound) and .9443 for the lower bound. So at least for German, combining parsers seems to be a pretty good idea. Thanks are due to Helmut Schmid and Prof. C. Rohrer for discussions, and to the reviewers for their detailed comments.

References

Steven Abney. 1991. Parsing by Chunks. In Robert C. Berwick, Steven P. Abney, and Carol Tenny, editors, *Principle-based Parsing: computation and psycholinguistics*, pages 257–278. Kluwer, Dordrecht.

Adam Berger, Stephen Della Pietra, and Vincent Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71, March.

E. Black, S. Abney, D. Flickinger, C. Gdaniec, R. Grishman, P. Harrison, D. Hindle, R. Ingria, F. Jelinek, J. Klavans, M. Liberman, M. Marcus, S. Roukos, B. Santorini, and T. Strzalkowski. 1991. A procedure for quantitatively comparing the syntactic coverage of English grammars. In *Proceedings of the DARPA Speech and Natural Language Workshop 1991*, Pacific Grove, CA.

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

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

Jason M. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. COLING '96, Copenhagen.

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.

Haim Gaifman. 1965. Dependency Systems and Phrase-Structure Systems. *Information and Control*, 8(3):304–337.

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.

Judita Preiss. 2003. Using Grammatical Relations to Compare Parsers. EACL'03, Budapest.

J. Ross Quinlan. 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA.

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. ACL'02, Philadelphia.

Jane J. Robinson. 1970. Dependency Structures and Transformational Rules. *Language*, 46:259–285.

Michael Schiehlen. 2002. Experiments in German Noun Chunking. COLING'02, Taipei.

Michael Schiehlen. 2003. A Cascaded Finite-State Parser for German. Research Note in EACL'03.

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

Lucien Tesnière. 1959. *Elements de syntaxe structurale*. Librairie Klincksieck, Paris.

Lei Xu, Adam Krzyzak, and Ching Y. Suen. 1992. Several Methods for Combining Multiple Classifiers and Their Applications in Handwritten Character Recognition. *IEEE Trans. on System, Man and Cybernetics*, SMC-22(3):418–435.