A Divide-and-Conquer Strategy for Shallow Parsing of German Free Texts

Günter Neumann* **Christian Braun†** **Jakub Piskorski‡**

Abstract

We present a divide-and-conquer strategy based on finite state technology for shallow parsing of real-world German texts. In a first phase only the topological structure of a sentence (i.e., verb groups, subclauses) are determined. In a second phase the phrasal grammars are applied to the contents of the different fields of the main and sub-clauses. Shallow parsing is supported by suitably configured preprocessing, including: morphological and on-line compound analysis, efficient POS-filtering, and named entity recognition. The whole approach proved to be very useful for processing of free word order languages like German. Especially for the divide-and-conquer parsing strategy we obtained an f-measure of 87.14% on unseen data.

1 Introduction

Current information extraction (IE) systems are quite successful in efficient processing of large free text collections due to the fact that they can provide a partial understanding of specific types of text with a certain degree of partial accuracy using fast and robust language processing strategies (basically finite state technology). They have been "made sensitive" to certain key pieces of information and thereby provide an easy means to skip text without deep analysis. The majority of existing IE systems are applied to English text, but there are now a number of systems which process other languages as well (e.g., German (Neumann et al., 1997), Italian (Ciravegna et al., 1999) or Japanese (Sekine and Nobata, 1998)). The majority of current systems perform a partial parsing approach using only very few general syntactic knowledge for the identification of nominal and prepositional phrases and verb groups. The combination of such units is then performed by means of domain-specific templates. Usually, these templates are triggered by domain-specific predicates attached only to a relevant subset of verbs which express domain-specific selectional restrictions for possible argument fillers.

In most of the well-known shallow text processing systems (cf. (Sundheim, 1995) and (SAIC, 1998)) cascaded chunk parsers are used which perform clause recognition after fragment recognition following a bottom-up style as described in (Abney, 1996). We have also developed a similar bottom-up strategy for the processing of German texts, cf. (Neumann et al., 1997). However, the main problem we experienced using the bottom-up strategy was insufficient robustness: because the parser depends on the lower phrasal recognizers, its performance is heavily influenced by their respective performance. As a consequence, the parser frequently wasn't able to process structurally simple sentences, because they contained, for example, highly complex nominal phrases, as in the following example:

> "[Die vom Bundesgerichtshof und den Wettbewerbshütern als Verstoß gegen das Kartellverbot gegeißelte zentrale TV-Vermarktung] ist gängige Praxis."
>
> *Central television marketing, censured by the German Federal High Court and the guards against unfair competition as an infringement of anti-cartel legislation, is common practice.*

During free text processing it might be not possible (or even desirable) to recognize such a phrase completely. However, if we assume that domain-specific templates are associated with certain verbs or verb groups which trigger template filling, then it will be very difficult to find the appropriate fillers without knowing the correct clause structure. Furthermore in a sole bottom-up approach some ambiguities – for example relative pronouns – can't be resolved without introducing much underspecification into the intermediate structures.

Therefore we propose the following *divide-and-conquer* parsing strategy: In a first phase only the verb groups and the topological structure of a sentence according to the linguistic *field the-*

* DFKI GmbH, Stuhlsatzenhausweg 3, 66123 Saarbrücken, Germany, `neumann@dfki.de`
† DFKI GmbH, Stuhlsatzenhausweg 3, 66123 Saarbrücken, Germany, `cbraun@dfki.de`
‡ DFKI GmbH, Stuhlsatzenhausweg 3, 66123 Saarbrücken, Germany, `piskorsk@dfki.de`

“[$_{CoordS}$ [$_{SSent}$ Diese Angaben konnte der Bundesgrenzschutz aber nicht bestätigen], [$_{SSent}$ Kinkel sprach von Horrorzahlen, [$_{relcl}$ denen er keinen Glauben schenke]]].”
This information couldn't be verified by the Border Police, Kinkel spoke of horrible figures that he didn't believe.

Figure 1: An example of a topological structure.

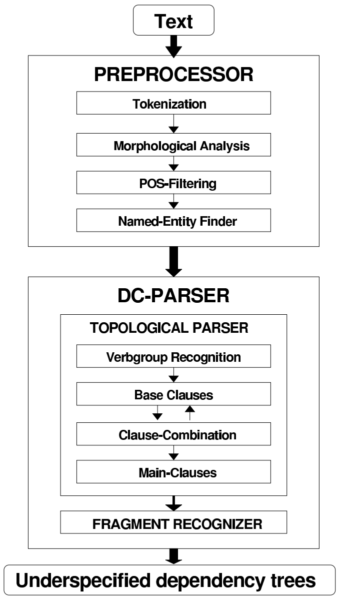

Figure 2: Overview of the system's architecture.

ory (cf. (Engel, 1988)) are determined domain-independently. In a second phase, general (as well as domain-specific) phrasal grammars (nominal and prepositional phrases) are applied to the contents of the different fields of the main and sub-clauses (see fig. 1)
This approach offers several advantages:

- improved robustness, because parsing of the sentence topology is based only on simple indicators like verbgroups and conjunctions and their interplay,
- the resolution of some ambiguities, including relative pronouns vs. determiner, subjunction vs. preposition and sentence coordination vs. NP coordination, and
- a high degree of modularity (easy integration of domain-dependent subcomponents).

The shallow divide-and-conquer parser (DC-PARSER) is supported by means of powerful morphological processing (including on-line compound analysis), efficient POS-filtering and named entity recognition. Thus the architecture of the complete shallow text processing approach consists basically of two main components: the preprocessor and the DC-PARSER itself (see fig. 2).

2 Preprocessor

The DC-PARSER relies on a suitably configured preprocessing strategy in order to achieve the desired simplicity and performance. It consists of the following main steps:

Tokenization The tokenizer maps sequences of consecutive characters into larger units called tokens and identifies their types. Currently we use more than 50 domain-independent token classes including generic classes for semantically ambiguous tokens (e.g., “10:15” could be a time expression or volleyball result, hence we classify this token as number-dot compound) and complex classes like abbreviations or complex compounds (e.g., “AT&T-Chief”). It proved that such variety of token classes simplifies the processing of subsequent submodules significantly.

Morphology Each token identified as a potential wordform is submitted to the morphological analysis including on-line recognition of compounds (which is crucial since compounding is a very productive process of the German language) and hyphen coordination (e.g., in “An- und Verkauf” (*purchase and sale*) “An-” is resolved to “Ankauf” (*purchase*)). Each token recognized as a valid word form is associated with the list of its possible readings, characterized by stem, inflection information and part of speech category.

POS-filtering Since a high amount of German word forms is ambiguous, especially word forms with a verb reading[1] and due to the fact that the quality of the results of the DC-PARSER relies essentially on the proper recognition of verb groups, efficient disambiguation strategies are needed. Using case-sensitive rules is straightforward since generally only nouns (and proper names) are written in standard German with a capitalized initial letter (e.g., “das Unternehmen” - *the enterprise* vs. “wir unternehmen” - *we undertake*). However for disambiguation of word forms appearing at the beginning of the sentence local contextual filtering rules are applied. For instance, the rule which forbids the verb written with a capitalized initial letter to be followed by a finite verb would filter out the verb reading of the word “unternehmen” in the sentence

[1] 30% of the wordforms in the test corpus “Wirtschaftswoche” (business news journal), which have a verb reading, turned to have at least one other non-verb reading.

"Unternehmen sind an Gewinnmaximierung interessiert." (*Enterprises are interested in maximizing their profits*). A major subclass of ambiguous wordforms are those which have an adjective or attributivly used participle reading beside the verb reading. For instance, in the sentence "Sie bekannten, die bekannten Bilder gestohlen zu haben." (*They confessed they have stolen the famous paintings.*) the wordform "bekannten" is firstly used as a verb (confessed) and secondly as an adjective (famous). Since adjectives and attributively used participles are in most cases part of a nominal phrase a convenient rule would reject the verb reading if the previous word form is a determiner or the next word form is a noun. It is important to notice that such rules are based on some regularities, but they may yield false results, like for instance the rule for filtering out the verb reading of some word forms extremely rarely used as verbs (e.g., "recht" - *right, to rake* (3rd person,sg)). All rules are compiled into a single finite-state transducer according to the approach described in (Roche and Schabes, 1995).[2]

Named entity finder Named entities such as organizations, persons, locations and time expressions are identified using finite-state grammars. Since some named entities (e.g. company names) may appear in the text either with or without a designator, we use a dynamic lexicon to store recognized named entities without their designators (e.g., "Braun AG" vs. "Braun") in order to identify subsequent occurrences correctly. However a named entity, consisting solely of one word, may be also a valid word form (e.g., "Braun" – brown). Hence we classify such words as candidates for named entities since generally such ambiguities cannot be resolved at this level. Recognition of named entities could be postponed and integrated into the fragment recognizer, but performing this task at this stage of processing seems to be more appropriate. Firstly because the results of POS-filtering could be partially verified and improved and secondly the amount of the word forms to be processed by subsequent modules could be considerably reduced. For instance the verb reading of the word form "achten" (watch vs. eight) in the time expression "am achten Oktober 1995" (at the eight of the October 1995) could be filtered out if not done yet.

3 A Shallow Divide-and-Conquer Strategy

The DC-PARSER consists of two major domain-independent modules based on finite state technology: 1) construction of the topological sentence structure, and 2) application of phrasal grammars on each determined subclause (see also fig. 3). In this paper we will concentrate on the first step, because it is the more novel part of the DC-PARSER, and will only briefly describe the second step in section 3.2.

3.1 Topological structure

The DC-PARSER applies cascades of finite-state grammars to the stream of tokens and named entities delivered by the preprocessor in order to determine the topological structure of the sentence according to the linguistic field theory (Engel, 1988).[3]

Based on the fact that in German a verb group (like "hätte überredet werden müssen" — **have convinced been should* meaning *should have been convinced*) can be split into a left and a right verb part ("hätte" and "überredet werden müssen'") these parts (abbreviated as LVP and RVP) are used for the segmentation of a main sentence into several parts: the front field (VF), the left verb part, middle field (MF), right verb part, and rest field (RF). Subclauses can also be expressed in that way such that the left verb part is either empty or occupied by a relative pronoun or a subjunction element, and the complete verb group is placed in the right verb part, cf. figure 3. Note that each separated field can be arbitrarily complex with very few restrictions on the ordering of the phrases inside a field.

Recognition of the topological structure of a sentence can be described by the following four phases realized as cascade of finite state grammars (see also fig. 2; fig. 4 shows the different steps in action).[4] Initially, the stream of tokens and named entities is separated into a list of sentences based on punctuation signs.[5]

Verb groups A verb grammar recognizes all single occurrences of verbforms (in most cases corresponding to LVP) and all closed verbgroups (i.e., sequences of verbforms, corresponding to RVP). The parts of discontinuous verb groups (e.g., separated LVP and RVP or separated verbs and verb-prefixes) cannot be put together at that step of processing because one needs contextual information which will only be available in the next steps. The major problem at this phase is not a structural one but the

[2]The manually constructed rules proved to be a useful means for disambiguation, however not sufficient enough to filter out all unplausible readings. Hence supplementary rules determined by Brill's tagger were used in order to achieve broader coverage.

[3]Details concerning the implementation of the topological parsing strategy can be found in (Braun, 1999). Details concerning the representation and compilation of the used finite state machinery can be found in (Neumann et al., 1997)

[4]In this paper we can give only a brief overview of the current coverage of the individual steps. An exhaustive description of the covered phenomena can be found in (Braun, 1999).

[5] Performing this step after preprocessing has the advantage that the tokenizer and named entity finder already have determined abbreviation signs, so that this sort of disambiguation is resolved.

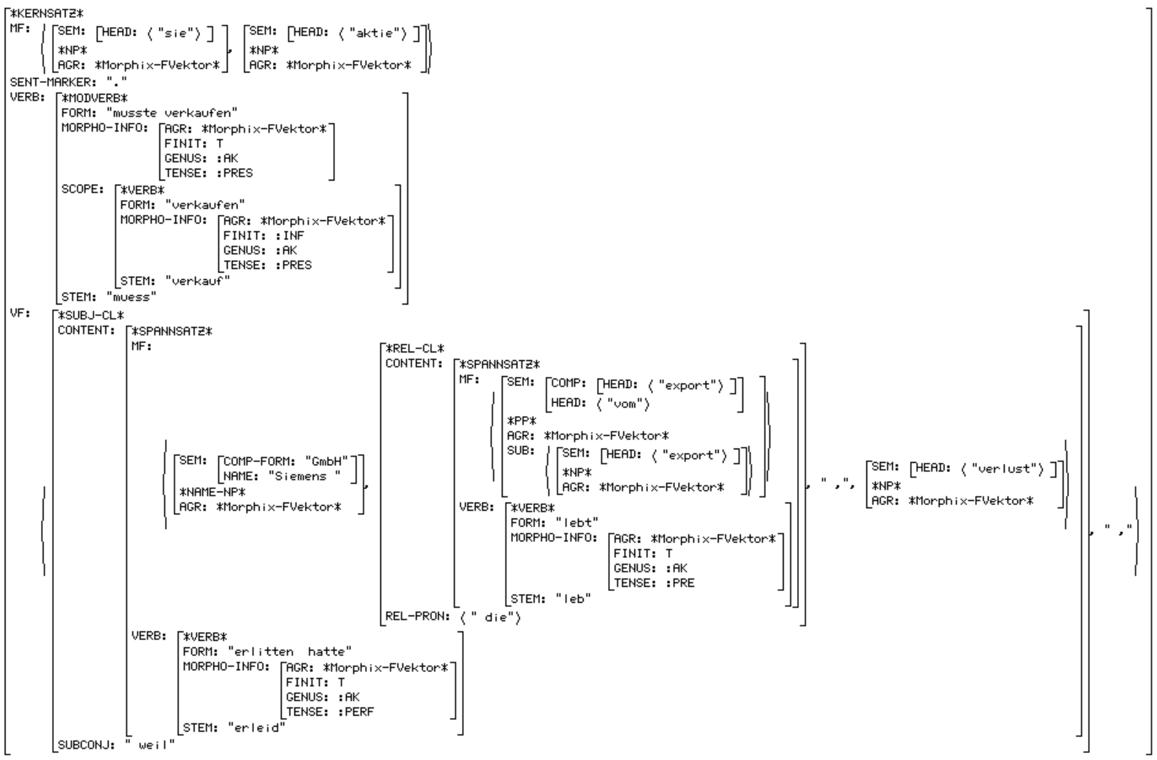

Figure 3: The result of the DC-PARSER for the sentence "Weil die Siemens GmbH, die vom Export lebt, Verluste erlitten hat, musste sie Aktien verkaufen." (*Because the Siemens GmbH which strongly depends on exports suffered from losses they had to sell some of the shares.*) abbreviated where convenient. It shows the separation of a sentence into the front field (VF), the verb group (VERB), and the middle field (MF). The elements of different fields have been computed by means of fragment recognition which takes place after the (possibly recursive) topological structure has been computed. Note that the front field consists only of one but complex subclause which itself has an internal field structure.

```
Weil die Siemens GmbH, die vom Export lebt, Verluste erlitt, musste sie Aktien verkaufen.
⇓
Weil die Siemens GmbH, die ...[Verb-Fin], Verl.  [Verb-Fin], [Modv-Fin] sie Akt.  [FV-Inf].
⇓
Weil die Siemens GmbH [Rel-Cl], Verluste [Verb-Fin], [Modv-Fin] sie Aktien [FV-Inf].
⇓
[Subconj-CL], [Modv-Fin] sie Aktien [FV-Inf].
⇓
[Subconj-CL], [Modv-Fin] sie Aktien [FV-Inf].
⇓
[clause]
```

Figure 4: The different steps of the DC-PARSER for the sentence of figure 3.

massive morphosyntactic ambiguity of verbs (for example, most plural verb forms can also be non-finite or imperative forms). This kind of ambiguity cannot be resolved without taking into account a wider context. Therefore these verb forms are assigned disjunctive types, similar to the underspecified chunk categories proposed by (Federici et al., 1996). These types, like for example `Fin-Inf-PP` or `Fin-PP`, reflect the different readings of the verbform and enable following modules to use these verb forms according to the wider context, thereby removing the ambiguity. In addition to a type each recognized verb form is assigned a set of features which represent various properties of the form like tense and mode information. (cf. figure 5).

Base clauses (BC) are subclauses of type subjunctive and subordinate. Although they are embedded into a larger structure they can independently

$$\begin{bmatrix} \text{Type} & \texttt{VG-final} \\ \text{Subtype} & \texttt{Mod-Perf-Ak} \\ \text{Modal-stem} & \text{könn} \\ \text{Stem} & \text{lob} \\ \text{Form} & \text{nicht gelobt haben kann} \\ \text{Neg} & \text{T} \\ \text{Agr} & \ldots \end{bmatrix}$$

Figure 5: The structure of the verb fragment "nicht gelobt haben kann" – **not praised have could-been* meaning *could not have been praised*

and simply be recognized on the basis of commas, initial elements (like complementizer, interrogative or relative item – see also fig. 4, where SUBCONJ-CL and REL-CL are tags for subclauses) and verb fragments. The different types of subclauses are described very compactly as finite state expressions. Figure 6 shows a (simplified) BC-structure in feature matrix notation.

$$\begin{bmatrix} \text{Type} & \textit{Subj-Cl} \\ \text{Subj} & \text{wenn} \\ \text{Cont} & \begin{bmatrix} \text{Type} & \textit{Spannsatz} \\ \text{Verb} & \begin{bmatrix} \text{Type} & \textit{Verb} \\ \text{Form} & \text{stellten} \\ \ldots & \end{bmatrix} \\ \text{MF} & \big(\text{die Arbeitgeber Forderungen}\big) \\ \text{NF} & \left(\begin{bmatrix} \text{Type} & \textit{Inf-Cl} \\ \text{Subj} & \text{ohne} \\ \text{Cont} & \begin{bmatrix} \text{Type} & \textit{Simple-Inf} \\ \text{Verb} & \begin{bmatrix} \text{Type} & \textit{Verb} \\ \text{Form} & \text{zu schaffen} \\ \ldots & \end{bmatrix} \\ \text{MF} & \text{(als Gegenleistung neue Stellen)} \end{bmatrix} \end{bmatrix}\right) \end{bmatrix} \end{bmatrix}$$

Figure 6: Simplified feature matrix of the base clause "…, wenn die Arbeitgeber Forderungen stellten, ohne als Gegenleistung neue Stellen zu schaffen." … *if the employers make new demands, without compensating by creating new jobs.*

Clause combination It is very often the case that base clauses are recursively embedded as in the following example:

> … weil der Hund den Braten gefressen hatte, den die Frau, nachdem sie ihn zubereitet hatte, auf die Fensterbank gestellt hatte.
> *Because the dog ate the beef which was put on the window sill after it had been prepared by the woman.*

Two sorts of recursion can be distinguished: 1) *middle field* (MF) recursion, where the embedded base clause is framed by the left and right verb parts of the embedding sentence, and 2) the *rest field* (RF) recursion, where the embedded clause follows the right verb part of the embedding sentence. In order to express and handle this sort of recursion using a finite state approach, both recursions are treated as iterations such that they destructively substitute recognized embedded base clauses with their type. Hence, the complexity of the recognized structure of the sentence is reduced successively. However, because subclauses of MF-recursion may have their own embedded RF-recursion the CLAUSE COMBINATION (CC) is used for bundling subsequent base clauses before they would be combined with subclauses identified by the outer MF-recursion. The BC and CC module are called until no more base clauses can be reduced. If the CC module would not be used, then the following incorrect segmentation could not be avoided:

> … *[daß das Glück [, das Jochen Kroehne empfunden haben sollte `Rel-Cl`] [, als ihm jüngst sein Großaktionär die Übertragungsrechte bescherte `Subj-Cl`], nicht mehr so recht erwärmt `Subj-Cl`]

In the correct reading the second subclause "… als ihm jüngst sein …" is embedded into the first one "… das Jochen Kroehne …".

Main clauses (MC) Finally the MC module builds the complete topological structure of the input sentence on the basis of the recognized (remaining) verb groups and base clauses, as well as on the word form information not yet consumed. The latter includes basically punctuations and coordinations. The following figure schematically describes the current coverage of the implemented MC-module (see figure 1 for an example structure):

CSent	::=	… LVP … [RVP] …
SSent	::=	LVP … [RVP] …
CoordS	::=	CSent (**,** CSent)* Coord CSent |
	::=	CSent (**,** SSent)* Coord SSent
AsyndSent	::=	CSent **,** CSent
CmpCSent	::=	CSent **,** SSent | CSent , CSent
AsyndCond	::=	SSent **,** SSent

3.2 Phrase recognition

After the topological structure of a sentence has been identified, each substring is passed to the FRAGMENT RECOGNIZER in order to determine the internal phrasal structure. Note that processing of a substring might still be partial in the sense that no complete structure need be found (e.g., if we cannot combine sequences of phrases to one larger unit). The FRAGMENT RECOGNIZER uses finite state grammars in order to extract nominal and prepositional phrases, where the named entities recognized by the preprocessor are integrated into appropriate places (unplausible phrases are rejected by agreement checking; see (Neumann et al., 1997) for more details)). The phrasal recognizer currently only considers processing of simple, non-recursive structures (see fig. 3; here, *NP* and *PP* are used for denoting phrasal types). Note that because of the high degree of modularity of our shallow parsing architecture, it is very easy to exchange the currently domain-independent fragment recognizer with a domain-specific one, without effecting the domain-independent DC-PARSER.

The final output of the parser for a sentence is an underspecified dependence structure UDS. An UDS is a flat dependency-based structure of a sentence, where only upper bounds for attachment and scoping of modifiers are expressed. This is achieved by collecting all NPs and PPs of a clause into separate sets as long as they are not part of some subclauses. This means that although the exact attachment point of each individual PP is not known it is guaranteed that a PP can only be attached to phrases which are dominated by the main verb of the sentence (which is the root node of the clause's tree). However, the exact point of attachment is a matter of domain-specific knowledge and hence should be defined as part of the domain knowledge of an application.

4 Evaluation

Due to the limited space, we concentrate on the evaluation of the topological structure. An evaluation of the other components (based on a subset of 20.000 tokens of the mentioned corpus from the "Wirtschaftswoche", see below) yields: From the 93,89% of the tokens which were identified by the morphological component as valid word forms, 95,23% got a unique POS-assignment with an accuracy of 97,9%. An initial evaluation on the same subset yielded a precision of 95.77% and a recall of 85% (90.1% F-measure) for our current named entity finder. Evaluation of the compound analysis of nouns, i.e. how often a morphosyntactical correct segmentation was found yield: Based on the 20.000 tokens, 1427 compounds are found, where 1417 have the correct segmentation (0.9929% precision). On a smaller subset of 1000 tokens containing 102 compounds, 101 correct segmentations where found (0.9901% recall), which is a quite promising result. An evaluation of simple NPs yielded a recall of 0.7611% and precision of 0.9194%. The low recall was mainly because of unknown words.

During the 2nd and 5th of July 1999 a test corpus of 43 messages from different press releases (viz. DEUTSCHE PREESSEAGENTUR (dpa), ASSOCIATED PRESS (ap) and REUTERS) and different domains (equal distribution of politics, business, sensations) was collected.[6] The corpus contains 400 sentences with a total of 6306 words. Note that it also was created after the DC-PARSER and all grammars were finally implemented. Table 1 shows the result of the evaluations (the F-measure was computed with β=1). We used the correctness criteria as defined in figure 7.

[6]This data collection and evaluation was carried out by (Braun, 1999).

The evaluation of each component was measured on the basis of the result of all previous components. For the BC and MC module we also measured the performance by manually correcting the errors of the previous components (denoted as "isolated evaluation"). In most cases the difference between the precision and recall values is quite small, meaning that the modules keep a good balance between coverage and correctness. Only in the case of the MC-module the difference is about 5%. However, the result for the isolated evaluation of the MC-module suggests that this is mainly due to errors caused by previous components.

A more detailed analysis showed that the majority of errors were caused by mistakes in the preprocessing phase. For example ten errors were caused by an ambiguity between different verb stems (only the first reading is chosen) and ten errors because of wrong POS-filtering. Seven errors were caused by unknown verb forms, and in eight cases the parser failed because it could not properly handle the ambiguities of some word forms being either a separated verb prefix or adverb.

The evaluation has been performed with the Lisp-based version of SMES (cf. (Neumann et al., 1997)) by replacing the original bidirectional shallow buttom-up parsing module with the DC-PARSER. The average run-time per sentence (average length 26 words) is 0.57 sec. A C++-version is nearly finished missing only the re-implementation of the base and main clause recognition phases, cf. (Piskorski and Neumann, 2000). The run-time behavior is already encouraging: processing of a German text document (a collection of business news articles from the "Wirtschaftswoche") of 197118 tokens (1.26 MB) needs 45 seconds on a PentiumII, 266 MHz, 128 RAM, which corresponds to 4380 tokens per second. Since this is an increase in speed-up by a factor > 20 compared to the Lisp-version, we expect to be able to process 75-100 sentences per second.

5 Related Work

To our knowledge, there are only very few other systems described which process free German texts. The new shallow text processor is a direct successor of the one used in the SMES–system, an IE-core system for real world German text processing (Neumann et al., 1997). Here, a bidirectional verb-driven bottom-up parser was used, where the problems described in this paper concerning parsing of longer sentences were encountered. Another similar divide-

Criterium	Matching of annotated data and results	Used by module
`Borders`	start and end points	verbforms, BC
`Type`	start and end points, type	verbforms, BC, MC
`Partial`	start or end point, type	BC
`Top`	start and end points, type for the largest tag	MC
`Struct1`	see `Top`, plus test of substructures using `Partial`	MC
`Struct2`	see `Top`, plus test of substructures using `Type`	MC

Figure 7: Correctness criteria used during evaluation.

Verb-Module						
correctness	*Verbfragments*			*Recall*	*Precision*	*F-measure*
criterium	total	found	correct	in %	in %	in %
`Borders`	897	894	883	98.43	98.77	98.59
`Type`	897	894	880	98.10	98.43	98.26
Base-Clause-Module						
correctness	*BC-Fragments*			*Recall*	*Precision*	*F-measure*
criterium	total	found	correct	in %	in %	in %
`Type`	130	129	121	93.08	93.80	93.43
`Partial`	130	129	125	96.15	96.89	96.51
Base-Clause-Module (isolated evaluation)						
correctness	*Base-Clauses*			*Recall*	*Precision*	*F-measure*
criterium	total	found	correct	in %	in %	in %
`Type`	130	131	123	94.61	93.89	94.24
`Partial`	130	131	127	97.69	96.94	97.31
Main-Clause-Module						
correctness	*Main-Clauses*			*Recall*	*Precision*	*F-measure*
criterium	total	found	correct	in %	in %	in %
`Top`	400	377	361	90.25	95.75	92.91
`Struct1`	400	377	361	90.25	95.75	92.91
`Struct2`	400	377	356	89.00	94.42	91.62
Main-Clause-Module (isolated evaluation)						
correctness	*Main-Clauses*			*Recall*	*Precision*	*F-measure*
criterium	total	found	correct	in %	in %	in %
`Top`	400	389	376	94.00	96.65	95.30
`Struct1`	400	389	376	94.00	96.65	95.30
`Struct2`	400	389	372	93.00	95.62	94.29
complete analysis						
correctness	*all components*			*Recall*	*Precision*	*F-measure*
criterium	total	found	correct	in %	in %	in %
`Struct2`	400	377	339	84.75	89.68	87.14

Table 1: Results of the evaluation of the topological structure

and-conquer approach using a chart-based parser for analysis of German text documents was presented by (Wauschkuhn, 1996). Nevertheless, comparing its performance with our approach seems to be rather difficult since he only measures for an unannotated test corpus how often his parser finds at least one result (where he reports 85.7% "coverage" of a test corpus of 72.000 sentences) disregarding to measure the accuracy of the parser. In this sense, our parser achieved a "coverage" of 94.25% (computing $found/total$), almost certainly because we use more advanced lexical and phrasal components, e.g., pos-filter, compound and named entity processing. (Peh and Ting, 1996) also describe a divide-and-conquer approach based on statistical methods, where the segmentation of the sentence is done by identifying so called link words (solely punctuations, conjunctions and prepositions) and disambiguating their specific role in the sentence. On an annotated test corpus of 600 English sentences they report an accuracy of 85.1% based on the correct recognition of part-of-speech, comma and conjunction disambiguation, and exact noun phrase recognition.

6 Conclusion and future work

We have presented a divide-and-conquer strategy for shallow analysis of German texts which is supported by means of powerful morphological processing, efficient POS-filtering and named entity recognition. Especially for the divide-and-conquer parsing strategy we obtained an F-measure of 87.14% on unseen data. Our shallow parsing strategy has a high degree of modularity which allows the integration of the domain-independent sentence recognition part with arbitrary domain-dependent subcomponents (e.g., specific named entity finders and fragment recognizers).

Considered from an application-oriented point of view, our main experience is that even if we are only interested in some parts of a text (e.g., only in those linguistic entities which verbalize certain aspects of a domain-concept) we have to unfold the structural relationship between all elements of a large enough area (a paragraph or more) up to a certain level of depth in which the relevant information is embedded. Beside continuing the improvement of the whole approach we also started investigations towards the integration of deep processing into the DC-PARSER. The core idea is to call a deep parser only to the separated field elements which contain sequences of simple NPs and PPs (already determined by the shallow parser). Thus seen the shallow parser is used as an efficient preprocessor for dividing a sentence into syntactically valid smaller units, where the deep parser's task would be to identify the exact constituent structure only on demand.

Acknowledgments

The research underlying this paper was supported by a research grant from the German Bundesministerium für Bildung, Wissenschaft, Forschung und Technologie (BMBF) to the DFKI project PARADIME, FKZ ITW 9704. Many thanks to Thierry Declerck and Milena Valkova for their support during the evaluation of the system.

References

S. Abney. 1996. Partial parsing via finite-state cascades. Proceedings of the ESSLLI 96 Robust Parsing Workshop.

C. Braun. 1999. Flaches und robustes Parsen Deutscher Satzgefüge. Master's thesis, University of the Saarland.

F. Ciravegna, A. Lavelli, N. Mana, L. Gilardoni, S. Mazza, M. Ferraro, J. Matiasek, W. Black, F. Rinaldi, and D. Mowatt. 1999. Facile: Classifying texts integrating pattern matching and information extraction. In *Proceedings of IJCAI-99*, Stockholm.

Ulrich Engel. 1988. *Deutsche Grammatik*. Julius Groos Verlag, Heidelberg, 2., improved edition.

S. Federici, S. Monyemagni, and V. Pirrelli. 1996. Shallow parsing and text chunking: A view on underspecification in syntax. In *Workshop on Robust Parsing, 8th ESSLLI*, pages 35–44.

G. Neumann, R. Backofen, J. Baur, M. Becker, and C. Braun. 1997. An information extraction core system for real world german text processing. In *5th International Conference of Applied Natural Language*, pages 208–215, Washington, USA, March.

L. Peh and Christopher H. Ting. 1996. A divide-and-conquer strategy for parsing. In *Proceedings of the ACL/SIGPARSE 5th International Workshop on Parsing Technologies*, pages 57–66.

J. Piskorski and G. Neumann. 2000. An intelligent text extraction and navigation system. In *6th International Conference on Computer-Assisted Information Retrieval (RIAO-2000)*. Paris, April. 18 pages.

E. Roche and Y. Schabes. 1995. Deterministic part-of-speech tagging with finite state transducers. *Computational Linguistics*, 21(2):227–253.

SAIC, editor. 1998. *Seventh Message Understanding Conference (MUC-7)*, http://www.muc.saic.com/. SAIC Information Extraction.

S. Sekine and C. Nobata. 1998. An information extraction system and a customization tool. In *Proceedings of Hitachi workshop-98*, http://cs.nyu.edu/cs/projects/proteus/sekine/.

B. Sundheim, editor. 1995. *Sixth Message Understanding Conference (MUC-6)*, Washington. Distributed by Morgan Kaufmann Publishers, Inc.,San Mateo, California.

O. Wauschkuhn. 1996. Ein Werkzeug zur partiellen syntaktischen Analyse deutscher Textkorpora. In Dafydd Gibbon, editor, *Natural Language Processing and Speech Technology. Results of the Third KONVENS Conference*, pages 356–368. Mouton de Gruyter, Berlin.