

A System for Checking NP Agreement in German Texts

Gerhard Fliedner

Institut für Computerlinguistik

Universität Saarbrücken

fliedner@coli.uni-sb.de

Abstract

In this paper, we describe the design, implementation, and evaluation of a system for checking NP agreement in German texts. We use shallow parsing based on finite state automata in combination with constraint relaxation and a method for parse ranking based on optimality theory. The system's grammar was developed and tested over a reference corpus of about 800,000 words to ensure robustness and broad coverage. When evaluated against a corpus of approximately 200,000 words, the system reached recall and precision values around the 67 % mark.

1 Overview

Grammar checkers are computer systems that try to find grammatical errors (and sometimes also stylistic infelicities) in texts and present them to the user, generally also suggesting a repair. The first grammar checkers were implemented in the late 1970's (Jensen et al., 1993). Since then, more than two dozen systems have been developed for a fairly large number of different languages (Bredenkamp et al., 2000; Oliva, 1997; Povlsen et al., 1999).

So far, grammar checkers are still not being widely used in the 'real world'. Evaluations (Wedbjer Rambell, 1999) and user interviews (Bernth, 2000) suggest that among the most important reasons for this are the low rates for both recall (i. e. too few of the actual errors are discovered by the

system) and precision (i. e. too many spurious errors are flagged) that these systems exhibit. In addition, users have frequently complained that applications run too slowly to process serious amounts of text (Oliva, 1997).

We have implemented a prototype of a grammar checking system for German. The system was mainly designed in order to investigate a novel combination of existing techniques to address the issues mentioned above. The system design relies on the interaction of the following: fast, large coverage morphology, finite-state based parsing with added expressive power to allow self-embedding, constraint relaxation, and parse ranking based on optimality theory. For errors that are found by the system, one or more repairs are suggested.

Since the implementation of a full grammar checker for all phenomena of German was beyond the scope of our project, we decided to confine ourselves to checking NP-internal agreement and case assignment by the preposition in PPs. Both are local phenomena governed by a relatively small set of rules (in principle, at least). Moreover, agreement errors and case errors in PPs occur in texts relatively frequently, so we could hope to collect sufficient data for the project. Among the errors in NPs/PPs that we observed, agreement errors were the majority, the rest mostly made up by punctuation, capitalization and spelling errors.

One important goal of our project was to implement a system that is able to cope not only with artificial test sentences, but with 'dirty' real world texts, i. e. to ensure robustness and broad coverage.

We have chosen to hand-code grammar and parse

ranking constraints instead of using statistical approaches. The rather complex German inflection paradigms lead to a large number of features and thus to a sparse data problem when it comes to reliably identifying complex, nested NPs. We found a hand-coded grammar robust enough for the task in hand. Future work is planned, however, on using statistical part of speech disambiguation prior to parsing with the hand-coded grammar.

For developing and testing the system we have therefore used a boot-strapping approach: we collected a reference corpus of about 800,000 words of German texts, mostly not proof-read students' work (assignments, theses) and various texts from the Internet, both by native and non-native speakers. We have repeatedly tested the system and the core grammar on the corpus and incrementally improved them. The final system was then evaluated over another, similar corpus of another 200,000 words.

2 NP Agreement in German

In German, all words in a noun phrase (NP) that can be declined (namely nouns, adjectives, and determiners) must agree in case, number, and gender. Moreover, adjectives show a different inflection depending on whether they are preceded by a definite, indefinite, or zero determiner. These different inflection paradigms have traditionally been called strong, weak and mixed inflection. In addition, certain determiners rather arbitrarily require a following adjective to carry either strong or weak inflection. NP agreement has frequently been described as one of the most difficult issues in German grammar.

The correct automatic recognition of German NPs is made difficult by the fact that complements and adjuncts of adjectives must be positioned inside the NP right before the adjective, thus forming an adjective phrase (AP). In general, adjectives in German may be modified by PPs, a lot of them also subcategorize for NP or PP complements. Thus, in German NPs and PPs can be (theoretically) arbitrarily deeply nested. An example is shown in (1).

In the corpora we used for implementing and evaluating the system, we have found quite a large number of agreement errors (about 1,500). A few of them are shown in (1)–(3) as an impression of the kinds of mistakes that occur in texts. The examples show

the following types of mistakes: adjective declension (1), wrong noun inflection paradigm (2), and wrong PP case (3).

- (1) **die* [[*für diesen Prozess*]_{PP} *nötige*_{sg}]_{AP}
the for this process required
*Temperaturen*_{pl}
temperatures
‘the temperatures required for this process’,
correct: *nötigen*_{pl}
- (2) **durch den Autoren*
by the author(s)
Correct: *den Autor* or *die Autoren*
- (3) **mitsamt seiner Argumente*
together with his_{gen} arguments_{gen}
Correct: *seinen*_{dat} *Argumenten*_{dat}

3 Morphology

As a basis for the syntactic analysis in our system, we have used the commercially available German two-level morphology GERTWOL (also see <http://www.lingsoft.fi/doc/gertwol/intro/gertwol.txt>). This morphology tool has been developed by the Finnish company Lingsoft Oy, Helsinki.

This tool analyzes and stems German words using inflection, derivation, and composition rules. It comprises a lexicon of approximately 300,000 German word stems – sufficient for most general texts.

4 Parsing

As German NPs tend to have a ‘flat’ structure (apart from embedding), the obvious choice of a grammar type for describing them seems to be finite state automata (FSA, Hopcroft and Ullman (1979)). They have a number of interesting features (intuitive grammars, closed under many set operations, linear-time parsing).

However, as mentioned in section 2 above, NPs and PPs can be nested in the case of APs in German. There is no theoretical limit to the depth of embedding (though there are, of course, limits by performance). Therefore, to properly describe nested NPs, FSAs are insufficient, as they are not suited to represent such nesting structures (Hopcroft and Ullman, 1979).

To conserve as many of the advantages as possible, we still chose FSAs as a basis for the system’s grammar, slightly extending them to allow self-embedding. We have called this type of automaton finite state automata with recursion (FSAR for short). Using cascaded FSAs with several parse levels, where the results of one level feed the next level, would have been another possible approach. Cascaded automata, however, need several passes even for ‘flat’ structures, whereas our approach just parses once.

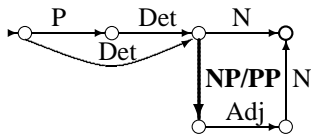


Figure 1: Toy FSAR for German NPs with self-embedding as in (1); the edge labelled *NP/PP* allows embedding an NP/PP

We will only describe the difference between FSA (Hopcroft and Ullman, 1979) and FSAR informally here: In addition to normal edges labelled with a single input symbol, an FSAR can have ‘self-embedding’ edges, labelled by a special symbol *S* not in the set of input symbols.

Whenever such an *S*-edge is encountered in parsing (i. e. when an embedded structure starts), its goal state is pushed onto a stack. Then the automaton is set into its initial state, and parsing continues. When a final state is reached (i. e. when the embedded structure ends), a state is popped from the stack and made the current state. This is resumed until a final state is reached with an empty stack. Figure 1 shows an example of a FSAR for a type of German NPs.

Though at first glance, FSARs resemble Recursive Transition Networks (RTN, Woods (1970)), there is an important difference: an RTN consists of a set of named automata that may call each other, so that their expressive power is exactly equal to push-down automata (Woods, 1970) and thus to context free grammars (CFG). With FSARs, an automaton may only call itself. We could show by an adapted pumping lemma (similar to the one for CFGs, Hopcroft and Ullman (1979)) that there are CFGs that cannot be represented by a FSAR (the CF language $a^n b^n c^m d^m$, e. g., cannot be described by FSARs). FSARs therefore provide an expressive

power that can be called ‘mildly context free’. The proof is beyond the scope of this paper.

FSARs can thus describe exactly the kind of self-embedding we need for parsing German NPs/PPs. Due to the constraint mentioned above, we expected them to parse more efficiently than CFGs. So far, a formal proof is still outstanding. However, the parser we have implemented allows efficient parsing (cf. section 7).

Using FSARs we developed two grammars, one of them a simple sentence structure grammar, the other the NP/PP grammar proper. The former is used to determine the sentence structure, i. e. to find relative and other subordinate clauses. This has proven necessary for disambiguation, as there is an ambiguity in German between relative pronouns and articles (for most inflection forms) and between subordinating conjunctions and prepositions.

The NP/PP grammar is used to describe most common types of German NPs and PPs, also including complex pre-nominal APs (chunker). The NP/PP grammar currently contains some 300 nodes and 7,500 edges, the sentence structure grammar contains some 60 nodes and 500 edges.

5 Constraint Relaxation

In grammar checking, one is confronted with an important issue: Whereas in ‘normal’ applications, the aim of parsing is to find out whether or not a given input belongs to some language *L*, in grammar checking one expects to find ungrammatical inputs. Since the user wants to be told *what* is wrong with the input, a grammar checking system must, moreover, try to find out what the intended input was, guided only by an erroneous structure.

Two main techniques have been used to overcome this problem: error anticipation, i. e. explicitly coding all error patterns that the system is supposed to recognize into the grammar (Bredenkamp et al., 2000; Povlsen et al., 1999), and constraint relaxation (Oliva, 1997; Povlsen et al., 1999).

When parsing with constraint relaxation, grammatical constraints, such as agreement, are relaxed at some level of the process. This allows the system to recognize the input’s structure even if it is not fully grammatical. If an input can be parsed only when some constraint is relaxed, one can infer

that this constraint has been violated, indicating both type and position of the grammatical error.

We have decided to use constraint relaxation in our system, as anticipating and describing all possible error patterns for agreement in German NPs would have been tedious and error-prone due to their large number. There have been different approaches to constraint relaxation, among them repeated parses with different constraints relaxed (Jensen et al., 1993) and parsing without constraints. We have chosen first to parse without constraints, i. e. based solely on the words of the input and their parts of speech, and only then checking the constraints in the resultant analyses. The danger of this approach lies in its potentially leading to a huge number of possible parses, since the constraints that would normally rule out most of them are not immediately checked. It has still proven feasible in the case of agreement checking, since the structure of German NPs is rather straightforward with comparatively little ambiguity and since with agreement we have only relaxed one constraint. With more than one possible parse, parse ranking is needed to resolve ambiguities.

6 Parse Ranking

Parse ranking, i. e. selecting a best parse or a small number of best parses from a ‘parse forest’, is especially important when grammatical constraints are relaxed, as the number of possible parses may become quite large. Different techniques have been used in different systems, most of them assigning a number designating ‘badness’ to each possible parse (Jensen et al., 1993; Genthial et al., 1994).

We have based our technique for parse ranking on optimality theory (OT, Kager (1999)). OT is used to rank different analyses of a given input by their markedness relative to a hierarchy of constraints, assuming that one or more of the constraints have been violated. The hierarchy starts with the most important constraints (giving rise to very marked input when violated), followed by the less important ones.

To find the best analysis, the different analyses yielded by the parser, consisting of POS tags, agreement features, and information on which words should agree, are first compared by the number of violations of constraints of the highest level. For all

analyses that show the same, lowest number of violations, this process is continued on the next-lowest level in the constraint hierarchy until only one best analysis is left.

We have manually chosen a number of constraints to rank parses. Constraints and constraint weighting were chosen to fit the examples from the reference corpus. The most prominent among them is, of course, agreement. To exemplify the hierarchy of constraints we have used for ranking, we give some examples for the interaction of the main constraints.

Firstly, analyses where no constraints are violated are obviously ranked highest. This is useful in cases where one analysis containing an agreement error and an alternative, error-free analysis exist. Such an alternative analysis could be based on an ambiguity. An article could, e. g., be analyzed as a relative pronoun by the sentence structure grammar mentioned in section 4. In such a case the error-free analysis would be preferred.

This is shown in (4): in this phrase, *das Autos* would (erroneously) be identified as a possible NP (structure: det n) with an agreement error (*das* is singular, *Autos* is plural). Since an alternative analysis of *das* as relative pronoun exists, this is preferred, and no error is flagged.

- (4) *das Kind, das Autos sammelt, ...*
the child, who cars collects, ...
‘the child who collects cars’

Secondly, analyses spanning the same input with a lower number of NP/PP chunks are preferred. This is needed in cases of agreement errors like the following:

- (5) **der_{neut} Künstliche Intelligenz_{fem}*
the Artificial Intelligence

Here an alternative error-free analysis with two chunks is imaginable, but highly improbable: #[*der Künstliche*] [*Intelligenz*] (‘the artificial one’ ‘intelligence’). Since the constraint ‘prefer lower number of chunks’ ranks higher than agreement, the ‘one chunk’ analysis is preferred, and thus the error is, correctly, flagged.

Thirdly, in the case of NPs containing an agreement error, parse ranking is used to find the best repair suggestion:

- (6) **die ähnlichen Text*
 the_{pl} $similar_{pl}$ $text_{sg}$

Here, we could assume that number agreement is violated and either plural or singular is intended (actually, among several other possibilities involving case agreement violation). In the former case agreement is violated once (and only one word must be changed, namely *Text* to *Texte*), in the latter case twice (and two words must be changed, namely *die ähnlichen* to *der ähnliche*). Therefore, only the first repair is suggested to the user, as fewer constraint violations on the same level have been counted.

7 Evaluation

After testing and improving our system over the 800,000 word reference corpus, we have evaluated it using another corpus of similar texts comprising about 200,000 words.

It is notoriously difficult to evaluate natural language processing systems and, even more so, to compare the results (EAGLES, 1995). Only very few of the groups who have implemented grammar checkers have actually reported their results (e.g. Wedbjer Rambell (1999)). For all systems evaluated in these reports, either precision or recall value were below the 50 %-mark. Quite often, both values were below 50 %.

Even given the figures, comparison between systems remains difficult: first, there is the obvious question, whether grammar checkers for different languages are comparable at all. Then, evaluation corpora may not be comparable: real world corpora, e. g. produce quite different results from ‘artificial’ test sentences (as in Wedbjer Rambell (1999)).

For want of a standardized testbed and/or evaluation corpus, we have therefore tried to evaluate the system as fairly as possible using a previously unseen corpus of ‘real world’ texts.

For this evaluation, recall and precision were determined. To determine precision, all errors that were flagged by the system were extracted and then annotated by two raters independently of each other (both of them students of computational linguistics). The raters were asked to classify flagged errors as correct or spurious and then to classify the error by its assumed source. In cases where the two raters disagreed, a third was used as an ‘expert’ to

arbitrate. The results are shown in Table 1. Precision varied according to text type between 34 % (news-papers) and 91 % (NNS), with professionally proof-read texts showing lower precision rates.

	real errors	spurious errors	errors flagged	precision
NS	252	154	406	62 %
NNS	239	25	264	91 %
Σ	491	179	670	73 %

Table 1: Precision values for the whole evaluation corpus (NS: texts by native speakers, NNS: text by non-natives)

Of the precision errors, about 40 % were due to the wrong assignment of NP boundaries (i. e. two NPs were erroneously treated as one). Here adding subcategorization information to the system could help (cf. section 8). Another 20 % are accounted for by unknown words, mostly foreign words, especially in the middle of complex NPs.

	Recall		Precision		<i>F</i> -Value	
	NS	NNS	NS	NNS	NS	NNS
S	67 %	83 %	78 %	73 %	72 %	77 %
G	42 %	49 %	42 %	26 %	42 %	34 %

Table 2: Recall, precision and *F*-values for two selected texts; comparison with Grammatik (S: our System, G: Grammatik)

To determine recall, we first had to establish a baseline, i. e. try to find all errors in the text. As this has to be done manually, it is a very time-consuming task. We therefore chose to annotate only two texts, one by a native speaker of German, one by a non-native speaker. Both texts were about equal in length (i. e. some 20,000 words). For determining the recall, we then added up all ‘real’ errors that were found in the text by either of the raters or the system to form the baseline for the recall figures. Table 2 shows the results.

About half of the recall errors of the system were due to missing coverage of the morphology, e. g. technical terms. Most cases of the other half have to do with the coverage of the grammar: here, work on the grammar should be continued.

We have compared these figures with those achieved by a commercial spelling checker for German. We chose ‘Grammatik’, last developed and distributed by Lernhout&Hauspie (Belgium). The system has reached a wide distribution, as it is part of the Microsoft Word 2000 distribution. Grammatik is a general purpose grammar checker, so it should be able to handle the sort of texts in the corpus. We only evaluated errors flagged by Grammatik that were due to NP/PP agreement errors. As mentioned in section 2, these errors tend to be (comparatively) local phenomena and well distinguishable from other grammatical errors. The results are shown in table 2.

The system speed has been evaluated over both reference and evaluation corpus on different machine configurations. The average processing speed is approximately 225 words/second on an industry standard PC with a Pentium III processor at 400 MHz (128 MByte RAM, Microsoft Windows NT), including morphological analysis and parsing. So, an average page can be processed in about 1.2 sec.

8 Conclusions and Future Work

We have presented in this article a system for checking NP agreement in German designed for robustness and broad coverage. The system uses a new combination of existing techniques to improve recall and precision. This is achieved by a combination of constraint relaxation with parse ranking based on optimality theory.

One interesting question is, how well the system scales up when additional grammatical phenomena are added. We believe that the combination of constraint relaxation and parse ranking offers a very good basis to add new phenomena without losing performance. Using this technique allows to describe the constraints locally, but still merging them into a overall ranking according to their severeness.

As all checking is done locally inside the NPs so far, one important improvement that we have planned is the addition of a mechanism for checking subcategorization. This would be very helpful to predict the number and the required cases for NPs in a sentence.

It is also planned in the near future to implement a commercial grammar checking system for foreign language learners for languages other than German

based on the techniques and findings of this system. Especially beginners tend to find such checking systems helpful, even if they are not perfect (Bernth, 2000). This should give us a good idea of how well our ideas and implementation transfer to other languages.

References

- Arendse Bernth. 2000. EasyEnglish: Grammar Checking for Non-Native Speakers. In *CLAW 2000*, pages 33–42, Seattle, Wash.
- Andrew Bredekamp, Berthold Crysmann, and Mirela Petrea. 2000. Looking for Errors: A Declarative Formalism for Resource-adaptive Language Checking. In *LREC 2000*, Athens.
- EAGLES. 1995. EAGLES. Evaluation of Natural Language Processing Systems. Eagles Document EAG-EWG-PR.2, Expert Advisory Group on Language Engineering Standards. URL: <http://issco-www.unige.ch/projects/ewg96/ewg96.html> (15/05/02).
- Damien Genthial, Jacques Courtin, and Jacques Ménézo. 1994. Towards a More User-friendly Correction. In *COLING-94*, pages 1083–1088, Kyōtō.
- John E. Hopcroft and Jeffrey D. Ullman. 1979. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, Reading.
- Karen Jensen, George E. Heidorn, and Stephen D. Richardson, editors. 1993. *Natural Language Processing: The PLNLP Approach*. Kluwer, Boston.
- René Kager. 1999. *Optimality Theory*. Cambridge University Press, Cambridge et al.
- Karel Oliva. 1997. Techniques for Accelerating a Grammar-Checker. In *5th ANLP*, pages 155–158, Washington, D. C.
- Claus Povlsen, Anna Sångvall Hein, and Koeraad de Smedt. 1999. Scarrie Final Report. URL: http://www.hltcentral.org/usr_docs/project-source/scarrie/ScarrieFinal.doc (15/05/02).
- Olga Wedbjer Rambell. 1999. A Study of Three Commercial Grammar Checkers. Working Papers in CL & Language Engineering 9, Universitet Uppsala. URL: <http://www.ling.uu.se/wp/wp9.pdf> (15/05/02).
- W. A. Woods. 1970. Transition Network Grammars for Natural Language Analysis. *Communications of the ACM*, 13(10):591–606.