

Grammar and Style Checking for German

Antje Schmidt-Wigger, MA.

IAI

Institut für angewandte Informationsforschung

Martin-Luther-Straße 14

D-66111 Saarbrücken

email: antje@iai.uni-sb.de

Abstract

As part of the MULTILINT project, a tool for grammar and style checking for Technical Documentation in German has been developed. The tool is based on a flat pattern matching approach. This approach has theoretical limits compared to parsing checkers, but still reaches acceptable results on real life corpora. We give some insights into the strategies for rule implementation, focussing on the differences between checking for grammatical errors and for stylistic errors. The influence of morphological ambiguity on the two checkers is underlined, and necessary and possible future enhancements are discussed.

1 Project context

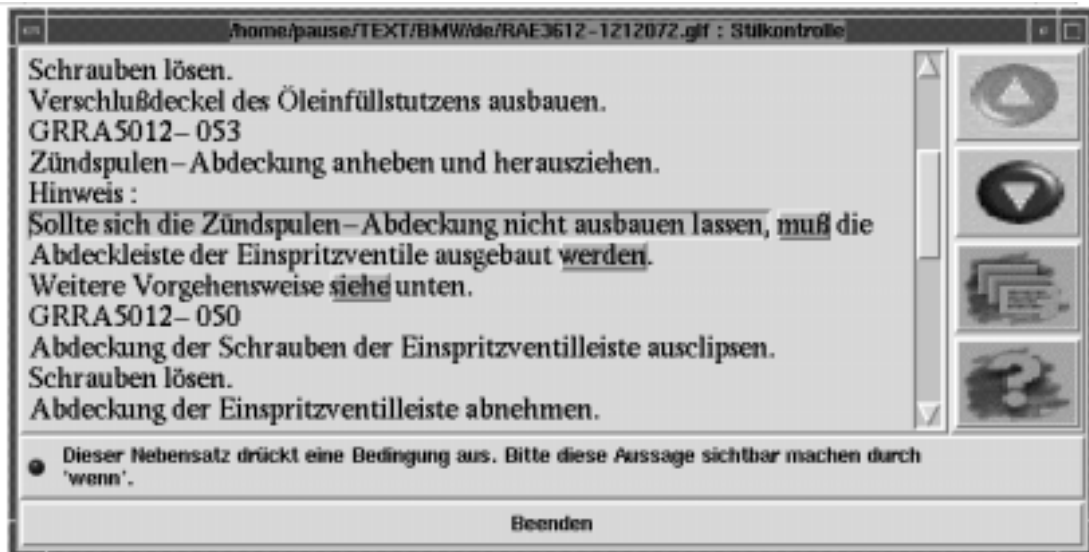
MULTILINT ([Haller(96)]) is an ongoing R&D project in the field of Technical Documentation¹. Its goal is to develop a text-oriented support toolbox for the technical writer who has to optimize his texts in terms of correctness, consistency and readability.

Part of the MULTILINT prototype is a user interface integrating the different tools into one system. It proposes a list of checking tools to the user which he can choose one by one: tools for spelling, grammar, terminology, abbreviation, consistency and style checking. This differentiation of error types is not common for commercial checkers². It was preferred, however, because it helps the user to become aware of the different problem areas and to pay more attention to these problems when writing. Also, he can personalize his environment, i.e. if he writes a text of a non-included text type, it would not make sense to use the style rules and he can concentrate on the other corrections.

In terms of machine-user interaction, it was decided that the whole text should be treated in one run, because the documents of the application domain are quite short (between 1 and 4 pages). However, results of each checking tool are presented separately to the user by a marker on the problematic parts in different copies of the treated document. This allows errors of different types to be marked in the same sentence. For each marking, the author can ask for a detailed error description by clicking on the marking itself.

¹The project is sponsored by the German Ministry of Economy (grant number II C 7-00 3048/5).

²The WORD checker, e.g., combines orthographic, grammatical and stylistic checking.



All the tools of the toolbox are based on a common linguistic analysis at morphological level ([Maas(96)]). The morphological analysis module has been developed at the IAI and covers a wide range of languages, with a treatment capacity of up to more than 95% correct analyses for an unknown German text. This basis allows for coherent treatment of the input texts.

Besides word-focussed checking tools, the MULTILINT prototype integrates a grammar and a style checking tool, which are applied to word clusters. The present article describes these two related tools which are based on the same formalism by focussing on their differences which have been revealed during the practical implementation.

2 Deep vs. flat analysis

Many style, grammar and CL checkers³ are based on a parsing approach where syntactic analysis of the sentences of the text precedes recognition of the erroneous or avoidable structures. There are a number of different arguments for adopting this **full-parse** approach:

- **Precision** (quality oriented)
If a full parse has been found for a checked sentence, the chance of getting a wrong error judgment diminishes.
- **Extension towards a corrector functionality** (future oriented)
Mostly only a full parse allows for controlled generation of a correction, should the checker be further developed towards a combined checker/corrector.
- **Reuse of linguistic resources** (pragmatical oriented)
Once a research group has developed a syntactic grammar for a language, it will look for application fields: style checking seems a very promising one.

During the definition phase of MULTILINT, all these different arguments have been considered, also because the developing partner has strong experience in the field of syntactic treatment of German. The corpus on which the project is based has been

³See for example the SECC system ([Adriaens(94)]), the Boeing BSEC Checker ([Wojcik(90)]), the Gram-Check project ([Ramírez&Sanchez(96)]) ..

submitted to different in-house analysis tools based on full parse, including a grammar checker ([Chambers et al. (95)]) and some machine translation oriented analyzers ([Sharp&Streiter(95)]). However, results were not very satisfactory; the following main disadvantages could be revealed:

- **Completeness of results**

When a system works on a strong full parse approach, existing errors can only be flagged when all structures represented in an input sentence are covered by the grammar. The tested grammars and syntactic lexicons did not cover the sublanguage represented by the project's corpus to a sufficient extent. Indeed, the corpus is written in a very heterogeneous language including complete, complex sentences as well as short, elliptic constructions. The fact that the sublanguage in question differs considerably from the standard language assumed in the tested systems led to a prohibiting number of completely failed analyses. A failure in the analysis process could be avoided when a phrasal parse output is allowed (e.g. in [Sharp&Streiter(95)]), but quality of results in terms of precision then diminishes ([Adriaens(94)]).

- **Response time**

Calculation of the full parse can take an unacceptable time for the user, especially if backtracking strategies are used. This problem has to be seen in combination with the interest of the user not to get a large mass of comments, but just some relevant criticisms on his text. If a system is tuned in such a way towards high precision on a given corpus, it could result in a number of texts that produce no error messages at all. In this case, the waiting time becomes even more unacceptable, because the user has waited for nothing.

The effort estimated to extend the positive coverage of the existing in-house grammars accordingly was much bigger than initially expected. On the basis of the results, the decision was taken to develop KURD, a completely new formalism for recognition of erroneous structures ([Carl&Schmidt-Wigger(98)]). The description of the erroneous structures is based on simple morphosyntactic pattern description. No parse tree is constructed, the objects treated by the system are flat structures. For this reason, we call it a **flat formalism**.

Input for the system is a rule set and morphologically analyzed text in feature bundle format, output is morphologically analyzed text annotated by error codes on the flagged words and word groups. Rules describe parts of sentences by using the same feature bundle format, augmented by quantifiers and variable binding. In addition, the formalism proposes differentiation between ambiguous and non-ambiguous feature values and a word counting mechanism. Necessary disambiguation and the error annotation are performed by unification on selected word nodes. This implies that only one error can be marked on each word or word group. The rules are ordered by preference and no backtracking is used during their application.

Here follows a rule example for recognition of prefixed infinitives, where the prefix is wrongly separated from the verb stem (cf. 3). Each line describes a word, the first one stands for the prefix ($c=vpref$), the second one for the verb ($c=verb$). To avoid the rule being wrongly applied, both words are limited in the part of speech they can take, which is expressed by the 'a(11)' quantifier on the second condition ($a\{c\sim=verb\}$ resp. $a\{c\sim=adj\}$). The other features allow for an ambiguity of their values, as they are stated under the 'e(xistence)' quantifier. In addition, some prefixes are restricted in the lexicon for a limited

set of verbs. This information is controlled through variable binding with the ‘verb’ feature of the prefix and the ‘l(exical)u(nit)’ of the verb (*verb=_V* and *lu=_V*). The colon separates the description part from the action part; once the description matches two word analyses, the warning number is added to these marked words by unification (*Au{warning='311'}*). If more context was necessary to identify the error, supplementary word descriptors without the marker ‘A’ could be added to the description part.

- (1) `pref = Ae{c=vpref,lu~=zu,verb=_V}a{c~=verb}e{warning='311'},`
`Ae{c=verb,gra=small,lu=_V,vtyp=inf}a{c~=adj}e{warning='311'}`
`: Au{warning='311'}.`

The formalism is used for grammar checking as well as for style checking, each of these two functionalities involving a different set of rules.

3 Grammar Checking

At the present stage, the rule set for the grammar checker covers 55 grammatical error classes. The error classes were selected by manually checking a test corpus of about 7,500 sentences. After implementation of the rules, the grammar check was performed automatically, again on the same corpus. The error classes can be grouped into the following fields of spelling and grammatical errors committed by native writers:

	Error frequency
Punctuation errors	238 errors
Errors of capitalization ⁴	17 errors
Errors of separating or combining words	46 errors
Errors in agreement	44 errors
Less common syntactic errors (repeating words, wrong position, missing articles...)	18 errors

Throughout the test corpus, punctuation errors represent two thirds of the revealed errors. By application of the automatic checker to the test corpus, the number of revealed grammatical errors in the corpus could be augmented wrt. the manual check. The higher importance of word separation errors and agreement errors only then became clear. To make evident the complexity of the task, we will give examples for the main error classes:

- (2) Punctuation error:
Relative clauses have to be separated by commas from the rest of the sentence.
Verbesserte Materialqualität des Blechteiles an welchem Bowdenzüge eingehängt sind ab KW 40/95 in Serie.
‘Improved material quality of the sheet metal part to which bowden cables are attached from CW 40/95 in production.’

⁴In German, all nouns have to be written beginning with capital letters. Errors often occur with nominalizations, where the syntactic category is not obvious for the writer.

- (3) Word separation error:
Verbal prefixes have to be attached to the infinitive form of the verb.
Einschraubsechskant bis zum Anschlag in die Ölpumpe zurück drehen.
 ‘Tighten cap screw firmly home in oil pump.’
- (4) Agreement error:
Subject and verb have to agree in number.
Die Teilenummer der Drehfallenschlösser sind gleichgeblieben.
 ‘The part number of the rotary-striker locks have remained constant.’

To augment the coverage of the checker in a general way, it could be interesting to allow the flagging of more than one grammatical error in the same context; i.e. the flagging of missing commas for subordinates is done - for pedagogical reasons - on the whole subordinate phrase. If this phrase contains an internal error, a distribution of the flagging is not possible for the moment. A slight change in the formalism could allow for this; the marking of the subordinate phrase would in this case be interrupted by a marking of a different type.

4 Style Checking

The rule set for the style checker is smaller than that for the grammar checker; it covers 40 stylistic error classes. The elaboration of these classes was less easy than for the grammatical error classes, as a corpus study alone cannot give an answer about undesirable constructions among the constructions found in the corpus. To approach the rule set, the error classes were first formulated with the help of experts and lectors from the technical documentation department of the user. After their implementation, their validity was tested by comparing the automatic checking of a 30-page document to its correction by a human corrector. Lessons could mainly be learned about the rules involving a counting mechanism (e.g. maximum length of sentence, maximum length of coordination inside a sentence..) as the scores described in literature and thus taken as starting points were much too low compared to the corrective intuition of the human corrector: she rewrote very few of the sentences underlined by counting rules, and thus, the scores had to be raised accordingly.

Another test which was important for the development of the rule set was the convergence test⁵ of the system. For helping the author in reformulating the sentence, he can ask in addition to the rule formulation for a reformulated example. The examples have been chosen from among the automatically retrieved style errors in the corpora; the reformulation has been proposed by the experts. By processing these examples, lessons could be learned about the status of some rules. In fact, in 25% of the reformulations, the system still revealed a stylistic error. In some cases, a reediting of the examples became necessary, but also the interaction of the rules involved had to be questioned again, i.e. rules should not propose a correction that would be filtered out by another rule. The test reflects in fact the

⁵ A Convergence Test has been first introduced in [Adriaens(95)]: The automatic correction produced by SECC is resubmitted to the checking process where no new correction is expected. Our convergence test is different, because we submit a manually corrected example to be rechecked. Thus, it involves a control of the ability of the experts to produce errorless sentences, or, in other words, a control of the consistency of the controlled language itself. But we would still argue also that the convergence test of SECC is NOT ONLY testing the tool, as is done by the Recall/Precision test, but also the CL itself. Indeed, when Adriaens classifies divergence problems wrt. their cause, most cases discussed in “Lingware Problems” are in fact problems of cyclic definition of the CL.

enormous work, collection of data and introspection that would be necessary to construct a non-contradictory rule set for just the syntactic part of a Controlled Language.

To give an idea of the rule set implemented up to now, we will give some of the most frequently occurring error messages as examples:

- (5) **This sentence contains too many units of meaning. Please rewrite it.**
Es dient bei evtl. Reklamationen mit dem nummerierten Arbeitsauftrag als Nachweis der im einzelnen durchgeführten Arbeiten und schützt den ausführenden Betrieb vor unberechtigten Werkstatt-, Gewährleistungs- oder sonstigen Regreßansprüchen.
‘With possible complaints, it serves as proof of the separate tasks carried out and protects the executing plant from unauthorized workshop, warranty or miscellaneous regress claims.’
- (6) **Two (or more) independent sentences are joined together. Please separate the sentences.**
Kaltstartprobleme, DDD-Kontrolllampe leuchtet, Motor läuft im Notprogramm.
‘Cold start problems, DDD indicator light shows, engine runs in the emergency program.’
- (7) **The verb (or the verb group) consists of two parts which are separated. Please rewrite the sentence so that the parts are closer together.**
Dieser stellt sich beim Beschleunigen aus ca. 1500 U/min. insbesondere im zweiten Gang unter hoher Last bzw. Vollast als inhomogenes Beschleunigungsverhalten dar.
‘This is shown to be inhomogeneous acceleration when accelerating from approximately 1500 U/min especially in the second gear under high load or full load.’
- (8) **This subordinate clause expresses a condition. Please make this clear by using ‘wenn’.**
Wird Korrosion festgestellt, sind die betroffenen Bauteile auszutauschen.
‘When corrosion is detected, the components concerned have to be replaced.’

As a reminder: a marking covers the words which are affected by the error. This implies that no overlapping errors can be communicated so that the choice of the marked words for a certain error has to be made with respect to the other errors implemented. This limitation, however, had positive effects for the design of the style checker. Here, the error classes were regrouped by the experts into three classes of importance. In this way, it should be avoided that a minor error prevents the flagging of larger parts of the sentence to be reformulated. Also diagnoses that allow for more specific guiding of the author should be preferred to general flagging of stylistically odd sentences. To exemplify this: the following sentence contains more than six nouns (cf. 5). But the message displayed tells the author to separate the two sentences contained (cf. 6), because this last rule is classified before the first rule.

- (9) *Bei seitlicher Belastung der Rückenlehne verformt das Rückenlehnensegment elastisch, dadurch tritt Relativbewegung zwischen Segment und Rückenlehne im unteren Schachtbereich auf.*
‘During lateral loading of the backrest, the backrest deck becomes elastically misshapen, and there thereby appears a relative movement between deck and backrest in the bottom cage area.’

The classification into three classes of importance can be related to the frequency of the error classes: by considering the most frequent and the less frequent occurring error classes

as two different groups, about all error classes of priority I are to be found between the most frequent, while most of the error classes of priority III are to be found between the less frequent.

In the next step, the rule set was divided into two sets for a different text type each. The corpus used in the project is composed of informative and instructive texts, and some of the stylistic rules are only relevant for one of the two text types.

(10) For informative texts only:

Always write complete sentences. Do not suppress the verb here.

Wärmetauscher undicht?

'Heat exchanger leaky?'

(11) For instructive texts only:

Avoid retrospective relations. Always put the instructions in the order of execution.

Anlageflächen von Schaumresten vorher reinigen.

'Clean contact surface of foam residue first.'

Up to now, more than 5,000 stylistically odd sentences have been automatically detected in the complete project's corpus of about 76,000 sentences.

5 Performance testing

The quality of both checkers was judged on the basis of recall and precision tests⁶ concerning the corpora described above. Precision was about 92% for the style checker on a corpus of about 750 complex sentences randomly selected among all sentences revealed automatically by the checker⁷. Results of comparison on the 30 pages document gave rise to about 65% recall. For the grammar checker, precision on the test corpus of 465 sentences was about 81%. Recall was about 57%. These last numbers will be augmented when implementation time equals that of the style checker, i.e. when the whole corpus has been taken into account for the grammar checker development.

To sum up, the present checkers can be compared to other systems publishing their results concerning the precision; concerning the recall, however, the results lie clearly behind them:

	Recall	Precision
MULTILINT grammar	57%	81%
MULTILINT style	65%	92%
SECC ([Adriaens(94)])	87%	93%
BSEC ([Wojcik(90)])	89%	79%

⁶Recall and Precision are concepts defined initially for retrieval problems. They are in common use for checker evaluation today. The formulas are:

Recall = Number of retrieved errors / Number of existing errors

Precision = Number of retrieved errors / Number of all retrieved cases

⁷This selection even did not contain style problems identifiable by one word nodes which do not involve syntactic complexity. For them, precision should be considered even higher.

We should not enter into a discussion about the reliability and the comparability of these figures here, which are subject to differences in the technical approach, development time, development and test corpora, rule sets and languages treated. What should be understood from these figures is that the MULTILINT checkers concentrate on a good precision rate rather than on recall. By that they follow the assumption that users accept more easily a system helping them in some cases than a system irritating them by wrong messages.

However, it seems that precision of a grammar checker can never reach that of a style checker. Indeed, a rule set for style checking has to cope with the sublanguage of the corpus, while a rule set for grammar checking has to cope with the sublanguage of the corpus AND with the erroneous structures it wants to check⁸. Thus, more possible structures presupposed by the rules give rise to more possible analyses of a sentence, and thus more chances to find a wrong one. Some parse grammar checkers try to handle this problem by separating the rule sets for allowed and erroneous structures. However, they can in fact reduce the problem in this way only for errorless sentences. If the sentence contains a grammatical error, the problem remains. For our flat formalism, this problem also exists; even if no parse tree has to be constructed, all syntactic constructions occurring in the corpus have to be taken into account. If a rule describes an error, it has to make sure that the patterns found by this rule in a text are not in fact part of other, permitted constructions.

6 Introduction of disambiguation rules

The difference between style and grammar checking tools described above is also linked to the possibility of using disambiguation rules before applying the error flagging rules. Error flagging rules try to identify syntactic patterns connected to a set of selected problem classes. Disambiguation rules are independent from the formulation of specific problem classes. Their role is to get rid of morphological ambiguity on the word level by means of the context of the ambiguous word. This role is necessary, because ambiguous input has a strong impact on the recall and/or the precision of a rule. If an error flagging rule evokes an ambiguous feature value to handle ambiguous input, it will yield wrong results in cases where other values should be the right ones. On the other hand, if the same rule evokes non-ambiguous feature values to restrain these undesirable applications, it cannot apply to words ambiguous for that feature. To be able to make maximal use of non-ambiguous feature value constraints, input should be disambiguated to a maximum.

In the output of our morphological system, ambiguity is expressed by multiple analyses, by value disjunction or by underspecification. To exemplify the degree of ambiguity, morphological analysis of the test corpus already gives rise to about 40% of ambiguity by multiple analyses only⁹. In the case of the style checker, a set of context implying disambiguation rules also written in the KURD formalism precedes the application of the style rules. This post-morphological processing reduces the ambiguity to about 10%. The reduction is probably even bigger concerning value disjunction and underspecification, i.e. in the recognition and disambiguation of noun phrase agreement. The following example gives the feature bundles for *in den Rückspiegeln* (= *in the rear-view mirrors*) before and after disambiguation.

⁸The present style checker is also able to handle ungrammatical structures in cases where the ungrammaticality does not interfere with the information used to detect the stylistic problem.

⁹This mainly involves differences in syntactic class and in verbal form.

```
(12) {ori=in,c=w,sc=p,ehhead={case=dat;acc}},
      {ori=den,c=w,sc=art,ehhead={nb=plu,case=dat};{g=m,nb=sg,case=acc}};
      {ori=den,c=w,sc=rel,ehhead={g=m,nb=sg,case=acc},ref={c=noun,g=m,nb=sg}},
      {ori=Ru1ckspiegeln,c=noun,s=process,ehhead={case=nom;dat;acc,nb=sg,g=n}};
      {ori=Ru1ckspiegeln,c=noun,s=instr,ehhead={case=dat,nb=plu,g=m}}
=>
      {ori=in,c=w,sc=p,ehhead={nb=plu,case=dat,g=m}},
      {ori=den,c=w,sc=art,ehhead={nb=plu,case=dat,g=m}},
      {ori=Ru1ckspiegeln,c=noun,s=instr,ehhead={case=dat,nb=plu,g=m}}
```

All kinds of ambiguities are represented: multiple analyses are given for the article/relative pronoun ambiguity on *den* and for the noun which can be analyzed as plural *rear-view mirrors* or singular *the reflecting*, value disjunction is given for the case ambiguity on the preposition or the 'ehhead' ambiguity for the article, and underspecification is given for the gender of the preposition and the article. All these ambiguities are resolved by the noun phrase rule.

In the case of grammar checking, however, context implying disambiguation has to be handled with greater attention. Grammar checking is applied to erroneous texts. Thus, if the context used for the disambiguation decision already contains a grammatical error, the decision taken by the rule can be wrong. Our decision, however, was not to do without disambiguation rules, but to find their best place of application by ordering carefully both the checking rules and the disambiguation rules. At the moment, order is as follows:

1. Identification of capitalization errors
2. Identification of wrong separation (cf. 3)
3. Disambiguation of agreement and category in noun and prepositional phrases
4. Disambiguation of category in general, of some verbal forms, prefixes etc.
5. Identification of punctuation errors (cf. 2)
6. Disambiguation of some verbal forms, relative pronouns etc.
7. Identification of agreement errors (cf. 4)
8. Identification of minor errors

In this way, wrong interpretation of the context influences as little as possible the application of the grammar checking rules. If for example the identification of wrong separation was not placed first, the information about the prefix analysis would have been suppressed by the rules for prefix disambiguation just on the basis of the position of the prefix candidate: the separated prefix can only stand at the end of the phrase, thus followed by the final point, or by y comma, if another clause should be extraposed at the end. The following example shows the analysis of *Anschlüsse gerade biegen* (= *straighten connections*, but not *just bend connections*) before and after prefix disambiguation. It can be easily seen that rule 3 can successfully apply only to the first analysis.

```
(13) {ori=Anschlüsse,c=noun,ehhead={case=nom;acc,nb=plu,g=m}},
      {ori=gerade,c=vpref,pctr=no};
      {ori=gerade,c=w,sc=part,pctr=no},
      {ori=biegen,c=verb,vtyp=inf}
=>
      {ori=Anschlüsse,c=noun,ehhead={case=nom;acc,nb=plu,g=m}},
      {ori=gerade,c=w,sc=part,pctr=no},
      {ori=biegen,c=verb,vtyp=inf}
```

Some experiments were undertaken to correct an error once it has been detected. In theory, this would ameliorate the result of later rules, because they could apply to an errorless context. In our corpus study, however, we encountered no case of wrong error marking because of misinterpretation due to a non-corrected error elsewhere in the sentence. Also, a complete correction at least on the feature level would be in conflict with the few context dependent features of the morphological analysis such as *wordnumber* and *last word*. Such an enhancement of the system would be further studied in combination with a corrector version of the checker.

7 Future work

In the future of the project, we wish to improve the style checker and the grammar checker described. We will concentrate on different areas for each of the checkers; as has been shown in the previous sections, differences exist between these two applications, which lead to other follow-up ideas.

Concerning the grammar checker, work will first of all concentrate on improving its capacities for the rules already implemented, because less time has been spent on its development. At the same time, the possibility of combining different error markings on one text part will probably be realized. A much bigger step would be the integration of automatic correction. Some experiments are in preparation which will follow the way adopted for checking the errors. But it seems already limited to some grammatical errors, as the amelioration of style often involves more complicated structures and more involvement of the author himself.

Concerning the style checker, the quality of results is already satisfactory. Some experiments will focus on the role of the disambiguation rules for this application; especially experiments will be made to see if cyclic application of the rules achieve better results. Other reflections will focus on the integration of checking a Controlled Language's lexicon. But already for the syntactic part of the rule set, work still can be continued to capture all structures which really disturb the readers attention.

8 Conclusion

MULTILINT integrates a flat checking system for grammatical and stylistic errors. The treatment of the two types of error is possible with the same mechanism, but has to cope with different problems. Grammar checking presents more difficulties for the implementation than style checking; style checking on the other hand presents more difficulties for defining the rule set. For both checkers, implementation work concentrated more on good precision, which has been proved in the test results. The experiment was successful: the MULTILINT prototype contains today, in the final phase of the project, a grammar and a style checker for German which will help the pilot users in their daily work.

References

- [Adriaens(94)] Adriaens, G. *Simplified English Grammar and Style Correction in an MT Framework: The LRE SECC Project*, in: Proceedings of the 16th Conference on Translating and the Computer (London), 78-88. 1994
Also in: Aslib Proceedings, 47 (3), March 1995, 73-82.
- [Adriaens(95)] Adriaens, G. and Macken, L., *Technological evaluation of a controlled language application: precision, recall, and convergence tests for SECC*, in: Proceedings of the 6th International Conference on Theoretical and Methodological Issues in Machine Translation (TMI95), Center for Computational Linguistics, Leuven 5-7/7/95, 123-141, 1995.
- [Ramírez&Sanchez(96)] Ramírez Bustamante, F. and Sánchez León, F., *Is linguistic information enough for grammar checking?*, in: Proceedings of the First International Workshop on Controlled Language Applications, Leuven, 26-27/3/96, 1996.
- [Carl&Schmidt-Wigger(98)] Carl, M. and Schmidt-Wigger, A., *Shallow Post Morphological Processing with KURD*, in: Proceedings of NeMLaP, Sydney, 1998.
- [Chambers et al. (95)] Chambers, C., Malnati, G., Tesio, R., Soma, E., *JDII – The Linguistic Syntax Checker*, in: Proceedings of Language Engineering '95, Montpellier, 121 - 129, 1995.
- [Fottner-Top(96)] Fottner-Top, C., *Workshop: Erstellung von verständlicher und benutzerfreundlicher technischer Dokumentation*, itl, 1996.
- [Haller(96)] Haller, J., *MULTITLINT, A Technical Documentation System with Multilingual Intelligence*, in: Translating and the Computer 18, London, Aslib, The Association for Information Management, Information House, 1996.
- [Maas(96)] Maas, H.-D., *MPRO - Ein System zur Analyse und Synthese deutscher Wörter*, in: Hausser, R. (ed.), *Linguistische Verifikation, Sprache und Information*, Max Niemeyer Verlag, Tübingen, 1996.
- [Sharp&Streiter(95)] Sharp, R. and Streiter, O., *Applications in Multilingual Machine Translation*, in: Proceedings of the Third International Conference of Prolog, Paris, 4.-7. April, 1995.
- [Wojcik(90)] Wojcik, R.H., Hoard, J.E and Holzhauser, K.C., *The Boeing Simplified English Checker*, in: Proceedings of the International Conference, Human Machine Interaction and Artificial Intelligence in Aeronautics and Space. Toulouse, Centre d'Etudes et de Recherche de Toulouse, 43-57, 1990.