The importance of precise tokenizing for deep grammars

Martin Forst*, **Ronald M. Kaplan****

*IMS, University of Stuttgart
Azenbergstr. 12
70174 Stuttgart, Germany
forst@ims.uni-stuttgart.de
**NLTT/ISL, Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, CA 94304, USA
kaplan@parc.com

Abstract

We present a non-deterministic finite-state transducer that acts as a tokenizer and normalizer for free text that is input to a broad-coverage LFG of German. We compare the basic tokenizer used in an earlier version of the grammar and the more sophisticated tokenizer that we now use. The revised tokenizer increases the coverage of the grammar in terms of full parses from 68.3% to 73.4% on sentences 8,001 through 10,000 of the TiGer Corpus.

1. Introduction

The work described in this paper was carried out in the DFG[1]-funded *DLFG*[2] project. Part of this project is to scale to newspaper corpora the German broad-coverage LFG developed in the *ParGram* parallel grammar development effort. This grammar used to be developed mostly in a phenomena-driven manner. As a consequence, grammar writers paid a lot of attention to interesting syntactic phenomena of German, but not so much to questions of low-level processing. In the following, we will show that high-quality low-level processing and, in particular, precise tokenization are crucial for scaling a hand-written grammar to free text.

The remainder of this paper is organized as follows: Section 2. presents the setup in which the tokenizer is used. Section 3. presents a selection of types of strings that used to be tokenized erroneously, which lead to parsing failure. Section 4. gives some evaluation figuresr. In section 5., we conclude.

2. Setup in which tokenizer is used

The tokenizer is used as part of a finite-state transducer (FST) cascade that preprocesses the input sentences before they are parsed by a large-scale LFG of German and the LFG parsing platform XLE. The setup is very similar to the one described in Kaplan et al. (2004), i.e. the input is tokenized and normalized, string-based multi-word identification is performed, morphological analysis is carried out, a guesser proposes analyses for otherwise unknown words and lexically based multi-word identification is performed. All steps are non-deterministic, since the goal is to allow all possible tokenizations (and morphological analyses etc.) as candidates for syntactic analysis, while sticking to the reasonable ones given the information available at the character level, in order not to blow up the downstream processing.

The grammar which is used for parsing this preprocessed input is a hand-written LFG. The grammar version we used for the evaluation presented at the end of this paper produces a full parse for 68.3% of sentences 8,001 through 10,000 of the TiGer Corpus when the original tokenizer is used. Thanks to a fallback mechanism, it produces partial (or fragment) parses for the remaining sentences, so that an overall coverage of 100% is achieved. In terms of quality (measured as the F-score on dependency triples, cf. Rohrer and Forst (2006)), full parses are, however, clearly superior to partial parses. It is thus desirable to produce full parses for as many sentences as possible.

3. Revising the tokenizer

The need to revise the tokenizer used as part of the grammar's FST cascade arose when we tried to enhance coverage (and parse quality). of the German ParGram LFG. When parsing the TiGer Corpus for the first time (instead of much smaller corpora or linguistic examples), we noticed that a considerable number of sentences could not be analyzed due to inappropriate tokenization.

3.1. Non-trivial tokenization issues not handled by the original tokenizer

The original tokenizer performed a very basic segmentation of the input sentences into tokens. For instance, all periods, except the ones at the end of a short list of common abbreviations and decimal/numerical points, were treated as separate tokens, which is clearly not intended in strings like the following (Instead of a translation, we indicate the intended tokenization in the third line of each example, where *TB* stands for 'token boundary'.):

(1) eine "K.o.-Tropfen-Bande"
a "k.o. drops gang"
eine TB " TB K.o.-Tropfen-Bande TB "

(2) in der Dominikus-Zimmermann-Str. 9
in the Dominikus Zimmermann street 9
in TB der TB Dominikus-Zimmermann-Str. TB 9

[1]Deutsche Forschungsgemeinschaft—'German Research Council'

[2]Disambiguierung einer Lexikalisch-Funktionalen Grammatik für das Deutsche—'Disambiguation of a Lexical Functional Grammar for German', grant *Ro 245/18-1*

Similarly, the original tokenizer treated basically all commas (except for decimal commas), apostrophes, parentheses, quotes and blanks as separate tokens and token boundaries respectively, which posed problems for strings like these:

(3) die 1,63-Meter-Frau
the 1.63-metre-woman
die TB 1,63-Meter-Frau

(4) Gibt's wieder Freikarten?
Are there again free tickets?
gibt TB 's TB wieder TB Freikarten?

(5) Veranstaltungsort ist Stiegl's Brauerei.
Event place is Stiegl's brewery.
Veranstaltungsort TB ist TB Stiegl's TB Brauerei TB .

(6) Karamanlis' Politik
Karamanlis's policy
Karamanlis' TB Politik

(7) die Kolleg(inn)en
the (both male and female) colleagues
die TB Kolleg(inn)en

(8) an zivilem (Verwaltungs-)Personal
of civil (administration) personnel
an TB zivilem TB (Verwaltungs-)Personal

(9) die "Aldi"-Brüder
the "Aldi" brothers
die TB "Aldi"-Brüder

(10) das "Soldaten sind Mörder"-Zitat
the "soldiers are murderers" citation
das TB "Soldaten sind Mörder"-Zitat

(11) rund 300 000 Quadratmeter
approximately 300,000 square metres
rund TB 300 000 TB Quadratmeter

(12) in eine angebliche ultima ratio-Situation
into an alleged ultima ratio situation
in TB eine TB angebliche TB ultima ratio-Situation

(13) ihre New York-Reise
their New York trip
ihre TB New York-Reise

Consider examples (4), (5) and (6), which concern the treatment of apostrophes. The apostrophe has a different function in each of them: In (4), it stands for the *e* of the contracted pronoun *es*; as *es* would be a separate token, we treat the sequence *'s* as such as well. In (5), the apostrophe is a common Anglicism in German, marking the genitive of a proper name; unlike the English *'s*, however, it is not a clitic, but only marks morphological case; we thus treat the sequence *'s* as part of the preceding word in this case. In (6), finally, the apostrophe marks the genitive of a proper name that ends in *s*, *x* or *z*; again, we treat it as part of the preceding word.

3.2. Text normalization

Another inadequacy of the original tokenizer was its insufficent ability to normalize text.

3.2.1. Decapitalization

The most important normalization when parsing free text is decapitalization at the beginning of a sentence, **but** also after opening quotes, brackets, colons and hyphens. The original tokenizer's inability to perform the latter caused strings like the following to be normalized insufficiently by the original tokenizer, i.e. *Die* in (14) and *Siehe* in (15) were not lower-cased. In order to remedy this situation, decapitalization was extended to capital letters after opening quotes, brackets, colons and hyphens.

(14) im Buch "Mystik Mythos Metaphysik –
in the book "mystics myth metaphysics –
... Mythos TB Metaphysik TB – TB
Die Spur des vermißten Gottes"
The trace of the missed God"
die TB Spur TB ...

(15) (Siehe auch Wirtschaft)
(See also economics)
(TB siehe TB auch TB Wirtschaft TB)

3.2.2. Normalization of variation in numbers

German exhibits considerable variation in the notation of numbers: Both the comma and the point appear as digital separators, and large numbers expressed in digits occur without any segmentation (cf. (16)), segmented by a point (cf. (17)), a blank (cf. (18)) or even an apostrophe (cf. (19)).

(16) Insgesamt 130000 DM
In total 13,000 DM
insgesamt TB 130000 TB DM

(17) Die rund 18.400 Ladiner
The approximately 18,400 Ladinians
die TB rund TB 18400 TB Ladiner

(18) 35 000 Menschen protestierten
35,000 people protested
35000 TB Menschen TB protestierten

(19) 300'000 Anleger wollen Swisscom-Aktien
300,000 investors want Swisscom shares
300000 TB Anleger TB wollen TB Swisscom-Aktien

Since applications that make use of the grammar do not want to bother about this type of variation, we decided to map all the segmented variants onto their unsegmented counterpart and to use the comma as the "normal" digital separator. Moreover, all of these strings can thus later be analyzed adequately by our FST morphology, whereas most of the segmented variants could not.

3.2.3. Haplology

Our grammar is punctuation-sensitive, which means that commas, hyphens, periods etc. are parsed just like words and appear in the resulting trees. (See, e.g., figure 1.) In general, it is not very difficult to handle punctuation in the syntactic rules of the grammar, in particular in a language like German, where punctuation is syntax-driven to a large extent. However, there are some low-level typographical conventions that complicate this picture: In German, as in many other languages, certain punctuation marks that would show up sentence-internally disappear when they would be adjacent to another punctuation mark. In other words, they are merged into the following punctuation mark (haplology). This is the case of, e.g., hyphens and commas before periods.

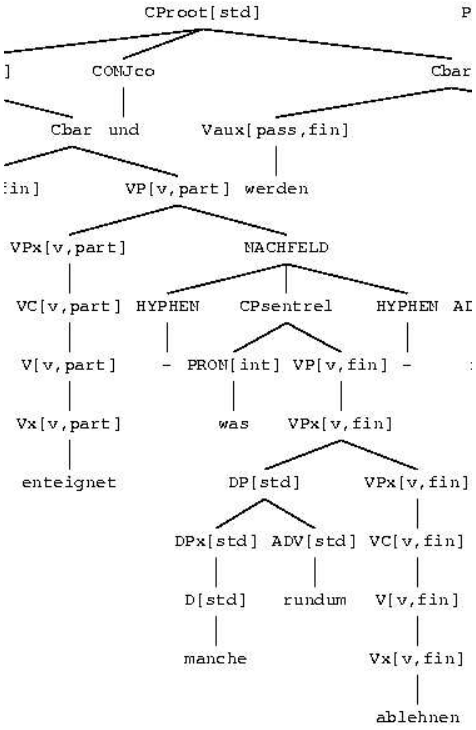

Figure 1: excerpt from the c-structure of (20a)

One possible solution to this problem would be to write distinct syntactic rules for constituents that are surrounded (or preceded or followed) by commas or hyphens for each context they can appear in. This solution would seriously hurt the modularity and maintainability of the grammar, however, since it would necessitate a multitude of almost identical rules that only differ with respect to punctuation.
We thus prefer a different solution, which, at least in part, was already implemented in the original tokenizer: During text normalization, additional optional commas are inserted before visible commas, periods, question marks and exclamation marks. What needed to be revised with respect to this insertion, were two things: First, other punctuation marks, e.g. hyphens, have to be taken into account as well, as they also take part in this type of interaction. Second, it is desirable to distinguish commas that are actually present in the surface string from the additional ones provided by the tokenizer.

(20) (a) Sie werden enteignet – was
They are expropriated – which
sie TB werden TB enteignet TB – TB was TB
manche rundum ablehnen – und
some altogether reject – and
manche TB rundum TB ablehnen TB – TB und TB
werden nur unzureichend entschädigt.
are only insufficiently compensated.
werden TB nur TB unzureichend TB entschädigt TB .

(b) Sie werden enteignet – was
They are expropriated – which
sie TB werden TB enteignet TB – TB was TB
manche rundum ablehnen.
some altogether reject.
manche TB rundum TB ablehnen TB _, TB .

(20b) illustrates the insertion of a an additional optional comma (_,); (20a) is a sentence where no additional comma needs to be inserted. This tokenizer output for both sentences can be processed by the following rule and lexical entries:

```
-   HYPHEN  *  .
,   COMMA   *  .
_,  HAP-COMMA  *  .

NACHFELD  -->  HYPHEN
               CPsentrel
               { HYPHEN
               | HAP-COMMA
               }.
```

The structures produced by this rule are illustrated in figures 1 and 2. Whereas in the former, the `CPsentrel` is enclosed by `HYPHEN`s on both sides, the latter is preceded by a `HYPHEN` and followed by a `HAP-COMMA`. Note that we do not allow sequences of `HYPHEN`, `CPsentrel` and regular `COMMA` to be analyzed as `NACHFELD`.

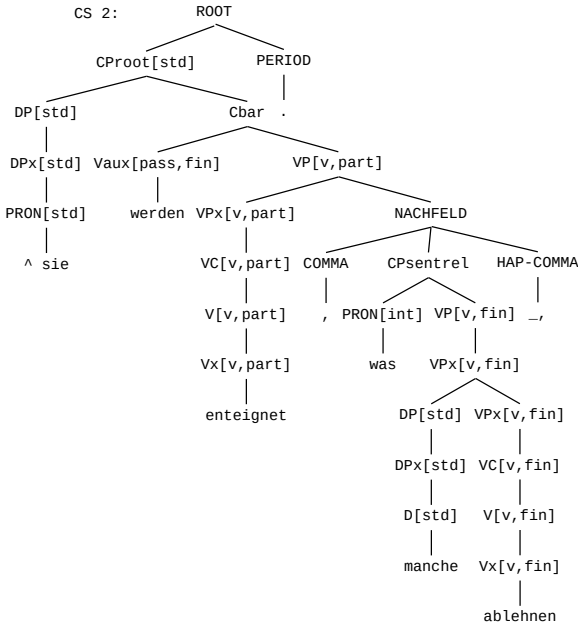

Figure 2: c-structure of (20b)

3.3. Interpreting quotes as potential "markup" for foreign material

One major obstacle for full coverage of German newspaper texts is the common occurrence of material from foreign languages in the text. However, this material is often "marked"by quotes for the human reader, and it seems wise to take advantage of these quotes for parsing. One way this can be done is the following: All strings between balanced quotes are optionally considered as one single token and marked as *+QuotedString*. (21) illustrates this solution.

(21) des zweiten Gegenkongresses
of the second anti-congress
des TB zweiten TB Gegenkongresses TB
"The other Economic Summit"
"The other Economic Summit"
The other Economic Summit TB +QuotedString

The output is then "collected" by the following rule and lexical entries:

```
-unknown  NE  *  (^PRED)='%stem'.

+QuotedString  QS_SFX  *  .

NAMEP  -->  NE
            QS_SFX.
```

`-unknown`, as it is shown here, is a guessing mechanism for any kind of token provided by the tokenizer; it allows the grammar to treat *The other Economic Summit* (as well as any other string) as an `NE`. This `NE` can then be combined with a `QS-SFX` and form a `NAMEP`. The resulting c-structure then looks as follows:

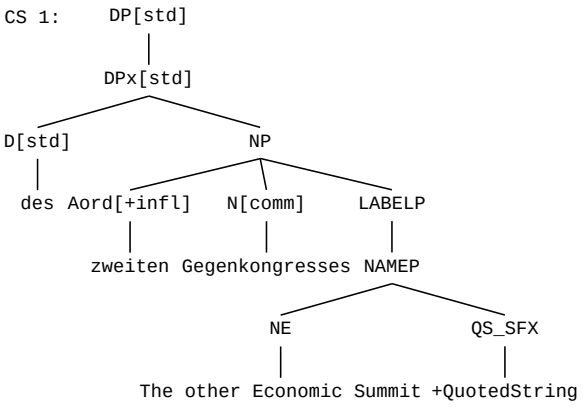

Figure 3: c-structure of (21)

4. Evaluation

For a comparative evaluation of the original and the new tokenizer, we parsed sentences 8,001 through 10,000 of the TiGer Corpus with grammar versions that only differed in the version of the tokenizer being used. The changes obtained with the new tokenizer were the following: Of the 2000 sentences, 301 vs. 423 sentences could not be parsed, 232 vs. 211 sentences timed out.[3] In total, this represents a gain of 101 sentences for which we get a full parse, which is an increase of 5.1 points (or 7.5%) in coverage. Given that full parses almost always result in a noticeably higher F-score than partial parses (cf. Riezler et al. (2002), Rohrer and Forst (2006)), we conjecture that the revision of the tokenizer contributes considerably to the overall parse quality of our grammar.

For a more direct evaluation of the tokenizer, we randomly selected 100 out of the 301 sentences that can still not be parsed and determined whether parsing failure was due to erroneous tokenization. For 88 of them, this is not the case. Among the 12 that failed due to tokenization errors, 5 contain foreign multi-word units, mostly names, that are not marked in any way, as in example (22)[4]; 3 sentences contain the upper-cased string *FRANKFURT A. M.*, where the *A.* is not lower-cased at the moment; 2 sentences have so-called *Aktenzeichen* in them, which are identifiers of court sentences etc. that consist of letters, digits and spaces (cf. example (23)); finally, 2 sentences contain foreign material between single quotes, which are not (yet) recognized as markup, as in example (24). The problem of the multi-words is difficult to address without having complete lists of place, organization and product names available. The remaining problems will be addressed in future work on the tokenizer.

(22) im Palazzo Reale
in the Palazzo Reale
im TB Palazzo Reale
im TB Palazzo TB Reale

(23) [...] (AZ: 9 C 73.95).
[...] (Aktenzeichen: 9 C 73.95).
[...] (TB AZ TB : TB 9 C 73.95 TB) TB .
[...] (TB AZ TB : TB 9 TB C TB 73.95 TB) TB .

(24) das ‚faire de la philosophie'
the 'faire de la philosophie'
das TB faire de la philosophie TB +QuotedString
das TB , TB faire TB de TB la TB philosophie TB '

5. Conclusion

We have shown that precise preprocessing, in this case tokenization and normalization, is important when handcrafted deep linguistic grammars are to be ported to free text. Revising these components may thus prove just as fruitful in terms of both coverage and parse quality as grammar extension "stricto sensu".

6. References

Miriam Butt, Helge Dyvik, Tracy H. King, Hiroshi Masuichi, and Christian Rohrer. 2002. The Parallel Grammar Project. In *Proceedings of COLING-2002 Workshop on Grammar Engineering and Evaluation*, pages 1–7.

Stefanie Dipper. 2003. *Implementing and Documenting Large-scale Grammars – German LFG*. Ph.D. thesis, IMS, University of Stuttgart. Arbeitspapiere des Instituts f¨ur Maschinelle Sprachverarbeitung (AIMS), Volume 9, Number 1.

Ronald M. Kaplan, John T. Maxwell, Tracy H. King, and Richard Crouch. 2004. Integrating Finite-state Technology with Deep LFG Grammars. In *Proceedings of the ESSLLI 2004 Workshop on Combining Shallow and Deep Processing for NLP*, Nancy, France.

Ronald M. Kaplan. 2005. A Method for Tokenizing Text. In *Festschrift in Honor of Kimmo Koskenniemi's 60th anniversary*. CSLI Publications.

Stefan Riezler, Tracy Holloway King, Ronald M. Kaplan, Richard Crouch, John T. Maxwell III, and Mark Johnson. 2002. Parsing the Wall Street Journal using a Lexical-Functional Grammar and Discriminative Estimation Techniques. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics 2002*, Philadelphia.

Christian Rohrer and Martin Forst. 2006. Improving coverage and parsing quality of a large-scale LFG for German. In *Proceedings of the Language Resources and Evaluation Conference (LREC-2006)*, Genoa, Italy.

[3] Both for uncovered sentences and for timeouts, XLE provides a fallback mechanism that enables us to produce partial parses for those sentences.

[4] We indicate the correct tokenization in the third line and the actual tokenization in the fourth line.