Text Analysis Meets Computational Lexicography

Hannah Kermes
Institut für Maschinelle Sprachverarbeitung,
Azenbergstr. 12,
70174 Stuttgart,
Germany
kermes@ims.uni-stuttgart.de

Abstract

More and more text corpora are available electronically. They contain information about linguistic and lexicographic properties of words, and word combinations. The amount of data is too large to extract the information manually. Thus, we need means for a (semi-)automatic processing, i.e., we need to analyse the text to be able to extract the relevant information.

The question is what are the requirements for a text analysing tool, and do existing systems meet the needs of lexicographic acquisition. The hypothesis is that the better and more detailed the off-line annotation, the better and faster the on-line extraction. However, the more detailed the off-line annotation, the more complex the grammar, the more time consuming and difficult the grammar development, and the slower the parsing process.

For the application as an analyzing tool in computational lexicography a symbolic chunker with a hand-written grammar seems to be a good choice. The available chunkers for German, however, do not consider all of the additional information needed for this task such as head lemma, morpho-syntactic information, and lexical or semantic properties, which are useful if not necessary for extraction processes. Thus, we decided to build a recursive chunker for unrestricted German text within the framework of the IMS Corpus Workbench (CWB).

1 A corpus linguistic approach to lexicography

In order to meet the needs of the maintenance of consistency and completeness within a lexicon, lexicography is moving away from solely manually constructed dictionaries to computer-assisted methods. Lexical engineering aims towards a scalable lexicographic work process for large lexica. It ensures that the processes are reproducible on large amounts of data. Besides, quality standards that are valid for large lexica assure the quality of the product. Thus, automatic acquisition of linguistic and lexicographic knowledge can make lexicographic work easier, and faster, and it can help to maintain consistency and completeness within the lexicon (cf. (Underwood, 1998; Kilgarriff and Tugwell, 2001a; Kilgarriff and Tugwell, 2001b)).

The question is which depth of analysis fits best the needs for the application of extraction queries. We understand the crucial requirements for a useful tool for corpus linguistic applications to be the following: (i) It has to work on unrestricted text. (ii) Lacks in the grammar should not lead to a complete failure to parse. (iii) No manual checking should be required as it is not feasible for large quantities of text. (iv) The system should provide clearly defined and documented interfaces, where the extraction processes can attach.

What kind of information should a corpus annotation providing a useful basis for extractions include, beside of the information on token level: (i) the head lemma of annotated structures to determine the lemma of the lexical entry, (ii) morpho-syntactic information, to determine the grammatical function of the structure and to extract additional aspects about potential lexical entries, e.g., singular plural alternations for nouns, (iii) lexical or semantic information, e.g., temporal aspect, (iv) information about certain embeddings, text markers, or construction types, (v) hierarchical representations.

The best, richest and most complex, and usually most reliable (especially if manual checking is involved) basis is provided by a full parse. However, full parsers often lack robustness and/or provide ambiguous output except for (unreliable) statistical disambiguation. Besides, they often have a slow parsing speed, which makes working with large corpora tedious. A chunk analysis, being robust, fast and considerably reliable seems a good alternative for large scale corpus linguistic work. At the moment, chunking is the only technically feasible approach. Yet, the question remains: can a chunker provide the relevant information listed above.

As Kübler and Hinrichs (Kübler and Hinrichs, 2001) have pointed out, while chunking approaches have " focused on the recognition of partial constituent structures at the level of individual chunks [...], little or no attention has been paid to the question of how such partial analyses can be combined into larger structures for complete utterances."

In other words, combining chunk structures provided by most available chunkers often need complex rules or rules that are neither secure nor theoretically motivated. For German, this is even more so than for English, for which most chunkers are designed. This is due to the fact that German has a tendency for phrase structures including pre-head embedding of complex structures.

2 YAC - A recursive chunker for unrestricted German text

Thus, we decided that we need a chunker which is especially designed to meet the needs of extraction of lexicographic information. YAC is a fully automatic recursive chunker for unrestricted German text. It is based on a symbolic regular expression grammar written in the CQP query language (Christ et al., 1999) which is part of the IMS Corpus Workbench[1]. The chunker works on a corpus which is tokenized and part-of-speech tagged using the STTS-tagset (Schiller et al., 1999). For tokenization and PoS-tagging the TreeTagger[2] (Schmid, 1994; Schmid, 1995) is used. The German grammar additionally requires lemma and agreement information on token level, which is annotated using the IMSLex morphology (Lezius et al., 2000).

[1]http://www.ims.uni-stuttgart.de/projekte/Corpus Workbench

[2]http://www.ims.uni-stuttgart.de/projekte/corplex/ TreeTagger/DecisionTreeTagger.html

In order to meet the needs of an ensuing extraction process, the classic chunk definition (cf. (Abney, 1996; Abney, 1991)) is extended to include recursion in pre-head as well as in post-head positions excluding PP-attachement and sentential elements.

The structures annotated by YAC comprise the following lexical phrase categories:(i) adverbial phrases (AdvP), (ii) adjectival phrases (AP), (iii) noun phrases (NP), (iv) prepositional phrases (PP), (v) verbal complexes (VC), (vi) single verbs (V), (vii) subordinate clauses (CL).

Additionally, feature attributes specifying certain properties and characteristics of the chunks are annotated. The properties are classified and stored in different feature attributes to ease the access. One large disjunctive feature attribute holding all information would be unwieldy. As each chunk category has different characteristics, the annotated feature attributes vary from chunk category to chunk category. In other words, each chunk category has its own annotation scheme. For AdvP, e.g., only head lemma and lexical properties are annotated, while for NPs head lemma, lexical properties, and morpho-syntactic information is annotated. There are three general feature attributes, which are common to most of the chunks: (i) head lemma, (ii) morpho-syntactic information, (iii) lexical-semantic and structural properties (including temporal aspect, proper nouns, directional adverbs, deverbal adjectives).

The head lemma of the chunk is taken from the lemma value the head position. The head position is either specified in the rule using a target, or in the rule processing using a "fixed" position, i.e., a position that can be determined independently of the actual results relative to another position. Normally, the head lemma is a single token, derived from a single position. In some cases, however, the lemmas of several tokens have been subsumed to form the head lemma. Multi-word proper nouns, e.g., have a multi-token head lemma, as a single lemma cannot be filtered out. The head lemma of verbal complexes with separated prefixes is a single-token head, however, it has been taken from two different position. The head lemma of PPs, consists of two separate head lemma items: the lemma of the preposition, and the lemma of the embedded NP.

YAC gains morpho-syntactic information of chunks using the morpho-syntactic features of relevant elements of the chunk. Invariant elements (e.g., invariant APs such as *lila* (purple)) are not considered. The morpho-syntactic information does not have to be, and in most cases, is not unique. If there is more than one element relevant for agreement, it is possible to reduce the ambiguity. An intersection of the different value-sets is used to determine the morpho-syntactic information of the chunk. In contrast to probabilistic approaches, no guessing is involved, i.e., if the value is still ambiguous, it is left ambiguous. In the case that no value is returned, i.e., the relevant elements do not agree, the chunk is rejected as agreement is required within a chunk.

Lexical-semantic and structural properties are important for the parsing as well as for further applications. The properties can be triggers for specific internal structures, functions, and usages of chunks. Some of the properties are inherent in the corpus itself, i.e., they can be determined from the information already present in the corpus: (i) PoS-tags, (ii) text markers. PoS-tags and text markers Named entities, e.g., can be derived from the PoS-tag `NE` for proper noun (1).

(1) [$_{NP}$ Johann Sebastian Bach]
NE NE NE

Text markers such as quotation marks, parenthesis, and brackets indicate the special character of a chunk (e.g. as named entity or possible modifier)(2), and can function as a secure context in which the restrictions on the chunks are relaxed.

(2) "Wilhelm Meisters Lehrjahre"

Other properties are determined by external knowledge sources, such as lexicons and ontologies. Local adverbs (3), e.g., are identified according to manually prepared word lists.

(3) hier (here); dort (there)

Another possibility to derive properties is from the chunking process itself. In this case, specific embeddings are indicated as properties of the embedding chunk. Complex AP embedding PPs (4a) and NPs (4b) are marked by a respective feature indicating the embedded structure.

(4) a. [$_{AP}$ [$_{PP}$ über die Köpfe der
above the heads of the
Apostel] gesetzten]
apostles set
'set above the heads of the apostles'

b. [$_{AP}$ [$_{NP}$ der "Inkatha"-Partei
to the Inkatha-Partei
] angehörenden]
belonging
'belonging to the Inkatha-Partei'

The grammar rules of YAC are written in an efficient query language (CQP), and then are post-processed by separate Perl-scripts. This allows: (i) efficient work even with large corpora, (ii) modular query language, (iii) interactive grammar development, (iv) powerful post-processing of rules.

Powerful compression algorithms allow one to work with large corpora. Even complex queries can be efficiently evaluated and processed. It is possible to work with corpora of 200-300 million tokens. The query language is modular, i.e., it allows to split complex rules into different blocks. The fact that the grammar rules are written in a query language allows an interactive development and testing of the rules. The same formalism used for the grammar rules can be used for interactive querying of the final results. Templates covering structures which are found to be relevant can be easily included in the parsing process if desired. Lexical information can be added without multiplying the grammar size unnecessarily. Different output formats can be provided, and hierarchical structures can be built.

The chunking process is divided intro three levels, which serve different purposes:

- First Level: (i) annotates base-chunks, (ii) annotates chunks with a specific internal structure, (iii) introduces lexical-semantic properties
- Second Level: (i) main parsing level, (ii) iterative application of general phrase structure rules to build recursive chunks
- Third Level: finishing level

There are several advantages of annotating base-chunks with specific internal structures and introducing lexical and semantic information in the first level: (i) the specific rules do not interact with the main parsing rules, (ii) the rules for chunks which do not involve complex (recursive) embedding have to be applied only once, (iii) the additional rules which are necessary to cover specific phenomena of specialized text domains can be included easily without affecting the main parsing process, (iv) the rules of the main parsing process can be kept relatively simple and general, as most special cases are already covered, (v) only a relatively small number of "general" rules is needed for the main parsing process.

In order to test real performance, we decided to build a manually annotated corpus as reference[3]. We extracted 400 sentences from the NEGRA corpus at random. The resulting reference contains 1920 NPs. We think that the size of the reference is large enough to make valid statements about precision and recall of the system. We calculated the precision as the number of true positives divided by the number of structures found by YAC, and the recall by dividing the number of true positives by the number of structures in the reference.

Evaluation was performed on ideal PoS-tags taken from the NEGRA corpus as well as on automatically annotated PoS-tags produced by the TreeTagger. In both cases, the morpho-syntactic information was not ideal, but automatically added and left ambiguous.

Table 1 gives the evaluation figures of YAC on ideal PoS-tags.

	all chunks		maximal chunks	
	precision	recall	precision	recall
NP	96.36	96.51	95.55	96.47
PP	98.08	96.51	98.07	96.50
AP	96.39	97.50	96.12	97.45
VC	–	–	99.01	98.59

Table 1: Evaluation figures of YAC on ideal PoS-tags

[3]We also tried to extract a gold standard from syntax annotation of NEGRA. However, it was impossible to cut-out chunks corresponding to our chunk definition. The problems we had are discussed in detail in (Kermes, 2003)

Precision figures for the evaluation of all chunks range from approximately 96% for APs and NPs to 98% for PPs. As VCs are not recursive, they do not occur in this category. Recall is slightly lower ranging from 96.5% for APs and NPs to 97.5% for PPs. The figures are slightly lower for the evaluation of maximal chunks. The reason for the decrease of figures can be explained by the fact that some structures are not combined to one large structure but left separate instead. In the case of NPs, e.g., certain post-head modifiers are not identified as such. Consequently, the maximal chunk cannot be identified correctly. However, embedded chunks can still be correctly identified. In the case of nominal post-head modifiers, the two NP structures are annotated as two separate maximal chunks as in (5a) instead of as one maximal chunk as in (5b).

(5) a. [$_{NP}$ eine Art] [$_{NP}$ Gesundheitspolizei]
a kind of health police

b. [$_{NP}$ eine Art Gesundheitspolizei]

In order to test the performance of YAC under real-life conditions, we evaluated its performance on automatically annotated PoS-tags as well. The evaluation figures are given in Table 2.

	all chunks		maximal chunks	
	precision	recall	precision	recall
NP	89.93	91.67	89.43	91.68
PP	94.05	89.67	94.04	89.65
AP	84.24	89.25	83.67	89.59
VC	–	–	97.72	96.62

Table 2: Evaluation figures of YAC on automatic PoS-tags

As can be observed, precision and recall figures for NPs and APs drop significantly if YAC is applied to a corpus PoS-tagged by the TreeTagger. The recall, although it is also significantly lower, is not affected to the same extent. Looking at the false analyses, it can be observed that two factors are responsible for most of the errors: (i) proper nouns, and (ii) capitalized words.

The first major factor for false analyses were proper nouns. In contrast to many other languages, German capitalizes both common nouns and proper nouns. Thus, it is difficult to distinguish between the two categories. The tagger has to guess whether an unknown capitalized word is a noun or a proper noun. In many cases, proper nouns are mistakenly tagged as common nouns. Thus, special rules for proper nouns cannot fire in all necessary cases. The consequence is that multi-word proper nouns cannot be assembled, and are left as separate NPs (6a). This is often punished not only once but several times, especially, if the proper nouns are embedded in larger structures. The correct analysis with flat annotation of post-head modifiers is given in (6b).

(6) a.

[$_{NP}$ Darstellung	[$_{NP}$ des
description	of the
Dortmunder	Sportmediziners]
Dortmund	sports physician
[$_{NP}$ Professor Klaus]]	[$_{NP}$ Völker]
Professor Klaus	Völker

b.

[$_{NP}$ Darstellung	[$_{NP}$ des
description	of the
Dortmunder	Sportmediziners]
Dortmund	sports physician
[$_{NP}$ Professor Klaus Völker]]	
Professor Klaus Völker	

In this case, the tagger failed to identify the surname *Völker* as a proper noun. Consequently, both the maximal NP and the NP *Professor Klaus Völker* could not be identified correctly. This results in two false negatives and three false positives. Thus, one false PoS-tag entails several faults in the chunker output.

The second major factor for errors in the annotation are capitalized words. The tagger did not seem to take sentence boundaries into account. Thus, in many cases, capitalized words at the beginning of sentences were erroneously tagged as nouns. Most of these wrongly tagged capitalized words were adjectives, which explains the low precision figures of APs on automatically tagged text. Again, similar to the proper nouns, one false tag entails several false structures. As the adjective is mistakenly analysed as separate NP as in (7).

(7)

[$_{NP}$ Seelische]	[$_{NP}$ Gesundheit]
psychological	health

The result of this multiple punishment is that the precision for APs and NPs drops quite dramatically. The recall of APs is also largely affected as many adjectives are erroneously tagged as nouns. The recall for NPs drops less dramatically as the tagging errors affect fewer structures than the mere precision figures would suggest.

PPs are affected less by tagging errors. The recall figures, which are lower in comparison to the recall figures for ideally tagged text, suggest that more correct PPs failed to be identified. The reason is that words which can function as prepositions, can have other functions, and thus, other PoS-tags as well (e.g., conjunction). However, if a preposition is erroneously annotated with another PoS-tag, the PP can no longer be identified. Additionally, the precision of PPs can be affected by wrong NP assignment as well.

VCs are affected least by automatic tagging. The tagger can obviously identify verbal elements with a good precision. If the tagger makes errors, it is with respect to the character of the verbal element (e.g., finite vs. infinite). However, as the rules for VC leave the verbal elements underspecified, the chunking performance is not affected by assignment of the wrong kind of verbal element.

The good precision figures prove the quality of the chunks annotated by YAC. The precision figures of YAC are almost as high as the figures of other state-of-the-art systems for German (Schiehlen, 2002; Skut and Brants, 1998; Brants, 1999). However, a real comparison is difficult because the single systems do not share the same theoretical background. Schiehlen (Schiehlen, 2002), e.g., applies his chunker to an ideally tokenized text, which includes the results of named entity recognition. That is, he takes multi-word proper nouns as given, and treats them as single classified tokens. We, however, try to identify named entities and other multi-word units on the basis of rules. The reason is, that we want to present a tool for extraction purposes. If we apply YAC to a large unknown corpus, we cannot expect to have a perfect lexicon including all multi-word units. Thus, we have to find a way to identify them on the basis of rules, and although we are successful in many cases, named entities are nevertheless one of the major error factors in the annotation of YAC. Thus, we can expect to obtain better results, if YAC were applied on a perfect tokenization in the sense of Schiehlen.

Brants (Brants, 1999) presents evaluation figures for his Cascaded Markov Model approach. He reports precision and recall for minimal (base) chunks (after 1 layer of annotation), and for maximal chunks (after 9 layers of annotation). Accordingly, the precision for maximal chunks is 91.4% and the recall 84.8%. The figures are based on automatically tagged text. The precision is comparable to that reported by Schiehlen, thus slightly higher than the precision of YAC, however, the recall is considerably lower. The recall of YAC is as high as its precision, which proves that YAC has a good coverage of phenomena. Other state-of-the-art systems have lower recall figures. A good recall, however, is important for the extraction of phenomena with relatively low frequency such as adjectives in predicative(-like) constructions. In this case we need large amounts of parsed text to be able to extract enough relevant data. Besides, it has to be taken into account that the tool of Brants was both trained and tested on the NEGRA corpus, albeit however, on different parts. Thus, the tool was specifically trained for this text.

3 Conclusion

YAC is a text analyzing tool for German text, which is both robust and efficient, and a good basis for the extraction of linguistic and lexicographic information. The goal is to relieve extraction tasks from parts of the linguistic analysis. The main principle behind YAC is that it tries to achieve a maximum of output with a minimum of input. In other words, based on a limited amount of prerequisite knowledge, we want to be able to extract large amounts of information.

We use a relatively small and simple set of rules for the annotation. Nevertheless, we are able to provide a useful basis for extraction processes. The annotated structures are relatively flat in comparison to a full parse. However, in comparison to a classic chunker, the structures are more complex, including complex (recursive) embedding. We try to find a compromise between the goal of providing sufficient structural information, and of annotating reliable structures. The principle behind the underlying chunk definition of YAC is to provide structures which are large enough to provide sufficient information, which are easily combinable into larger structures, and do not involve highly ambiguous attachment decisions (e.g., PP-attachment).

Simple rules and simple structural annotation means also implementing few theoretical assumptions. This does not make the annotation theory-independent, but at least less bound to a given linguistic theory. Consequently, YAC can be useful for applications from various theoretical backgrounds.

The grammar rules of YAC rely only on lexical information such as PoS-tags, lemma and morpho-syntactic information. The latter is usually ambiguous, and lists all possible options for a word. We do not make use of information about subcategorization frames, selectional preferences, cooccurrences. As we want to extract this kind of information, we cannot depend on it as input to the system. The lexical-semantic information we use is limited to small lists of words. These lists together with simple pattern matching strategies, and certain heuristics can be used to annotate a number of chunks with different lexical-semantic properties.

We do not only annotate extended chunk structures, but enrich the structures additionally with information relevant for extraction. This information includes the head lemma, information about morpho-syntax, and lexical-semantic properties of the head. State-of-the-art chunkers - if at all - provide only some of the relevant information. This information is important, if not necessary for extraction processes.

Despite the relatively simple and flat annotation, YAC provides a basis for fine-grained distinctions among the extracted data. The additional information allows for a powerful filtering and grouping of the results. We do not only want to extract simple subcategorization frames, but are interested in selectional preferences, cooccurrences, distributional variations, and relations among the different phenomena. Available systems do not allow to make such a fine-grained distinction

References

Steven Abney. 1991. Parsing by chunks. In Robert Berwick, Steven Abney, and Carol Tenny, editors, *Principle-Based Parsing.* Kluwer Academic Publishers.

Steven Abney. 1996. Chunk stylebook. Working draft.

Thorsten Brants. 1999. Cascaded markov models. In *Proceedings of 9th Conference of the European Chapter of the Association for Computational Linguistics EACL-99*, Bergen, Norway.

Oliver Christ, Bruno M. Schulze, Anja Hofmann, and Esther König, 1999. *The IMS Cropus Workbench: Corpus Query Processor (CQP): User's Manual.* University of Stuttgart: Institute for Natural Language Processing, Azenbergstr.12, 70174 Stuttgart, Germany, August.

Hannah Kermes. 2003. *Off-line (and Online) Text Analysis for Computational Lexicography.* Ph.D. thesis, Institut für Maschinelle Sprachverarbeitung, Universität Stuttgart. Submitted.

Adam Kilgarriff and David Tugwell. 2001a. WASP-Bench: an MT lexicographers' workstation supporting state-of-the-art lexical disambiguation. In *Proceedings of MT Summit VII*, pages 187–190, Santiago de Compostela.

Adam Kilgarriff and David Tugwell. 2001b. WORD SKETCH: Extraction and display of significant collocations for lexicography. In *In Proceedings of the workshop "COLLOCATION: Computational Extraction, Analysis and Exploitation", 39th ACL & 10th EACL*, pages 32–38, Toulouse, July.

Sandra Kübler and Erhard W. Hinrichs. 2001. TüSBL: A similarity-based chunk parser for robust syntactic processing. In *Proceedings of HLT 2001*, San Diego, California, March.

Wolfgang Lezius, Stefanie Dipper, and Arne Fitschen. 2000. IMSLex – representing morphological and syntactical information in a relational database. In Ulrich Heid, Stefan Evert, Egbert Lehmann, and Christian Rohrer, editors, *Proceedings of the 9th EURALEX International Congress,Stuttgart, Germany*, pages 133–139, Stuttgart, Germany.

Michael Schiehlen. 2002. Experiments in German noun chunking. In *Proceedings of the 19th International Conference on Computational Linguistics.*

Anne Schiller, Simone Teufel, Christine Stöckert, and Christine Thielen. 1999. Guidelines für das Tagging deutscher Textcorpora mit STTS. Technical report, Universität Stuttgart, IMS and Universität Tübingen, November.

Helmut Schmid. 1994. Probabilistic part-of-speech tagging using decision trees. In *International Conference on New Methods in Language Processing*, pages 44–49, Manchester, UK.

Helmut Schmid. 1995. Improvements in part-of-speech tagging with an application to German. In *Proceedings of the ACL SIGDAT-Workshop.*

Wojciech Skut and Thorsten Brants. 1998. A maximum-entropy partial parser for unrestricted text. In *Sixth Workshop on Very Large Corpora*, pages 143–151, Montreal, Canada.

Nancy Underwood. 1998. Issues in designing a flexible validation methodology for nlp lexica. In *Proceedings of the First International Conference on Language Resources and Evaluation*, volume 1, pages 129–134, Granada.