An Information Extraction Core System for Real World German Text Processing

Günter Neumann* Rolf Backofen† Judith Baur‡ Markus Becker§ Christian Braun¶

Abstract

This paper describes SMES, an information extraction core system for real world German text processing. The basic design criterion of the system is of providing a set of basic powerful, robust, and efficient natural language components and generic linguistic knowledge sources which can easily be customized for processing different tasks in a flexible manner.

1 Introduction

There is no doubt that the amount of textual information electronically available today has passed its critical mass leading to the emerging problem that the more electronic text data is available the more difficult it is to find or extract relevant information. In order to overcome this problem new technologies for future information management systems are explored by various researchers. One new line of such research is the investigation and development of *information extraction* (IE) systems. The goal of IE is to build systems that find and link relevant information from text data while ignoring extraneous and irrelevant information (Cowie and Lehnert, 1996).

Current IE systems are to be quite successfully in automatically processing large text collections with high speed and robustness (see (Sundheim, 1995), (Chinchor et al., 1993), and (Grishman and Sundheim, 1996)). This is due to the fact that they can provide a partial understanding of specific types of text with a certain degree of partial accuracy using fast and robust shallow processing strategies (basically finite state technology). They have been "made sensitive" to certain key pieces of information and thereby provide an easy means to skip text without deep analysis.

The majority of existing information systems are applied to English text. A major drawback of previous systems was their restrictive degree of portability towards new domains and tasks which was also caused by a restricted degree of re-usability of the knowledge sources. Consequently, the major goals which were identified during the sixth message understanding conference (MUC-6) were, on the one hand, to demonstrate task-independent component technologies of information extraction, and, on the other hand, to encourage work on increasing portability and "deeper understanding" (cf. (Grishman and Sundheim, 1996)).

In this paper we report on SMES an information extraction core system for real world German text processing. The main research topics we are concerned with include easy portability and adaptability of the core system to extraction tasks of different complexity and domains. In this paper we will concentrate on the technical and implementational aspects of the IE core technology used for achieving the desired portability. We will only briefly describe some of the current applications built on top of this core machinery (see section 7).

2 The overall architecture of SMES

The basic design criterion of the SMES system is to provide a set of basic powerful, robust, and efficient natural language components and generic linguistic knowledge sources which can easily be customized for processing different tasks in a flexible manner. Hence, we view SMES as a *core information extraction system.* Customization is achieved in the following directions:

*DFKI GmbH, Stuhlsatzenhausweg 3, 66123 Saarbrücken, Germany, neumann@dfki.uni-sb.de

†LMU, Oettingenstrasse 67, 80538 München, Germany, backofen@informatik.uni-muenchen.de

‡DFKI GmbH, baur@dfki.uni-sb.de

§DFKI GmbH, mbecker@dfki.uni-sb.de

¶DFKI GmbH, cbraun@dfki.uni-sb.de

- defining the flow of control between modules (e.g., cascaded and/or interleaved)
- selection of the linguistic knowledge sources
- specifying domain specific knowledge
- defining task-specific additional functionality

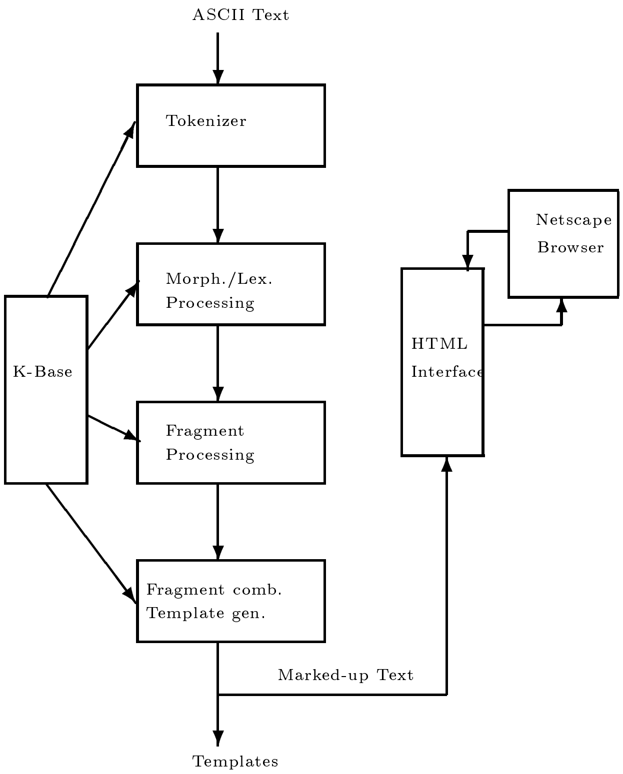

Figure 1: A blueprint of the core system

Figure 1 shows a blueprint of the core system (which roughly follows the design criteria of the generic information extraction system described in (Hobbs, 1992)) . The main components are:

A tokenizer based on regular expressions: it scans an ASCII text file for recognizing text structure, special tokens like date and time expressions, abbreviations and words.

A very efficient and robust German morphological component which performs morphological inflection and compound processing. For each analyzed word it returns a (set of) triple containing the stem (or a list of stems in case of a compound), the part of speech, and inflectional information. Disambiguation of the morphological output is performed by a set of word-case sensitive rules, and a Brill-based unsupervised tagger.

A declarative specification tool for expressing finite state grammars for handling word groups and phrasal entities (e.g., general NPs, PPs, or verb groups, complex time and date expressions, proper name expressions). A finite state grammar consists of a set of fragment extraction patterns defined as finite state transducers (FST), where modularity is achieved through a generic input/output device. FST are compiled to Lisp functions using an extended version of the compiler defined in (Krieger, 1987).

A bidirectional lexical-driven shallow parser for the combination of extracted fragments. Shallow parsing is basically directed through *fragment combination patterns* FCP of the form $\langle FST_{left}, anchor, FST_{right} \rangle$, where *anchor* is a lexical entry (e.g., a verb like "to meet") or a name of a class of lexical entries (e.g., "transitive-verb"). FCPs are attached to lexical entries (e.g., verbs), and are selected right after a corresponding lexical entry has been identified. They are applied to their left and right stream of tokens of recognized fragments. The fragment combiner is used for recognizing and extracting clause level expressions, as well as for the instantiation of templates.

An interface to TDL, a type description language for constraint-based grammars (Krieger and Schäfer, 1994). TDL is used in SMES for performing type-driven lexical retrieval, e.g., for concept-driven filtering, and for the evaluation of syntactic agreement tests during fragment processing and combination.

The knowledge base is the collection of different knowledge sources, viz. lexicon, subgrammars, clause-level expressions, and template patterns. Currently it includes 120.000 lexical root entries, subgrammars for simple and complex date and time expressions, person names, company names, currency expressions, as well as shallow grammars for general nominal phrases, prepositional phrases, and general verb-modifier expressions.

Additionally to the above mentioned components there also exists a generic graphical editor for text items and an HTML interface to the Netscape browser which performs marking of the relevant text parts by providing typed parentheses which also serve as links to the internal representation of the extracted information.

There are two important properties of the system for supporting portability:

- Each component outputs the resulting structures uniformly as feature value structures, together with its type and the corresponding start and end positions of the spanned input expressions. We call these output structures *text items*.

- All (un-filtered) resulting structures of each component are cached so that a component can take into account results of all previous components. This allows for the definition of cascaded as well as interleaved flow of control. The former case means that it is possible to apply a cascade of finite state expressions (comparable to that proposed in (Appelt et al., 1993)), and the latter supports the definition of finite state expressions which incrementally perform a mix of keyword spotting, fragment processing, and template instantiation.[1]

The system has already successfully been applied to classifying event announcements made via email, scheduling of meetings also sent via email, and extraction of company information from on-line newswires (see 7 for more details). In the next section, we are describing some of the components' properties in more detail.

3 Word level processing

Text scanning Each file is firstly preprocessed by the *text scanner*. Applying regular expressions (the text scanner is implemented in lex, the well-known Unix tool), the text scanner identifies some text structure (e.g., paragraphs, indentations), word, number, date and time tokens (e.g, "1.3.96", "12:00 h"), and expands abbreviations. The output of the text scanner is a stream of tokens, where each word is simply represented as a string of alphabetic characters (including delimiters, e.g. "Daimler-Benz"). Number, date and time expressions are normalized and represented as attribute values structures. For example the character stream "1.3.96" is represented as (:date ((:day 1)(:mon 3)(:year 96)), and "13:15 h" as (:time ((:hour 13)(:min 15))).

Morphological processing follows text scanning and performs inflection, and processing of compounds. The capability of efficiently processing compounds is crucial since compounding is a very productive process of the German language.

The morphological component called MONA is a descendant of MORPHIX, a fast classification-based morphology component for German (Finkler and Neumann, 1988). MONA improves MORPHIX in that the classification-based approach has been combined with the well-known two-level approach, originally developed by (Koskenniemi, 1983). Actually, the extensions concern

- the use of tries (see (Aho et al., 1983)) as the sole storage device for all sorts of lexical information in MONA (e.g., for lexical entries, prefix, inflectional endings) , and

- the analysis of compound expressions which is realized by means of a recursive trie traversal. During traversal two-level rules are applied for recognizing linguistically well-formed decompositions of the word form in question.

The output of MONA is the word form together with all its readings. A reading is a triple of the form $\langle stem, inflection, pos \rangle$, where *stem* is a string or a list of strings (in the case of compounds), $inflection$ is the inflectional information, and *pos* is the part of speech.

Currently, MONA is used for the German and Italian language. The German version has a very broad coverage (a lexicon of more then 120.000 stem entries), and an excellent speed (5000 words/sec without compound handling, 2800 words/sec with compound processing (where for each compound all lexically possible decompositions are computed).[2]

Part-of-speech disambiguation Morphological ambiguous readings are disambiguated wrt. part-of-speech using case-sensitive rules[3] and filtering rules which have been determined using Brill's unsupervised tagger (Brill, 1995). The filtering rules are also used for tagging unknown words.

The filtering rules are determined on the basis of unannotated corpora. Starting from untagged corpora, MONA is used for initial tagging, where unknown words are ambiguously tagged as noun, verb, and adjective. Then, using contextual information from unambiguously analysed word forms, filter rules are determined which are of the form *change tag of word form from noun or verb to noun if the previous word is a determiner*.

First experiments using a training set of 100.000 words and a set of about 280 learned filter rules

[1] Of course, it is also possible—and usually the case in our current applications—to combine both sorts of control flow.

[2] Measurement has been performed on a Sun 20 using an on-line lexicon of 120.000 entries.

[3] Generally, only nouns (and proper names) are written in standard German with an capitalized initial letter (e.g., "der Wagen" *the car* vs. "wir wagen" *we venture*). Since typing errors are relatively rare in press releases (or similar documents) the application of case-sensitive rules are a reliable and straightforward tagging means for the German language.

yields a tagging accuracy (including tagging of unknown words) of 91.4%.[4]

Note that the un-supervised tagger required no hand-tagged corpora and considered unknown words. We expect to increase the accuracy by improving the un-supervised tagger through the use of more linguistic information determined by MONA especially for the case of unknowns words.

4 Fragment processing

Word group recognition and extraction is performed through fragment extraction patterns which are expressed as finite state transducers (FST) and which are compiled to Lisp functions using a compiler based on (Krieger, 1987). An FST consists of a unique name, the recognition part, the output description, and a set of compiler parameters.

The recognition part An FST operates on a stream of tokens. The recognition part of an FST is used for describing regular patterns over such token streams. For supporting modularity the different possible kind of tokens are handled via *basic edges*, where a basic edge can be viewed as a predicate for a specific class of tokens. More precisely a basic edge is a tuple of the form $\langle name, test, variable \rangle$, where *name* is the name of the edge, *test* is a predicate, and *variable* holds the current token T_c , if *test* applied on T_c holds. For example the following basic edge (:mona-cat "partikel" pre) tests whether T_c produced by MONA is a particle, and if so binds the token to the variable *pre* (more precisely, each variable of a basic edge denotes a stack, so that the current token is actually pushed onto the stack).

We assume that for each component of the system for which fragment extraction patterns are to be defined, a set of basic edges exists. Furthermore, we assume that such a set of basic edges remains fix at some point in the development of the system and thus can be re-used as pre-specified basic building blocks to a grammar writer .

Using basic edges the recognition part of an FST is then defined as a regular expression using a functional notation. For example the recognition part for simple nominal phrases might be defined as follows:

```
(:conc
  (:star≤n (:mona-cat "det" det) 1)
  (:star (:mona-cat "adj" adj))
  (:mona-cat "n" noun))
```

Thus defined, a nominal phrase is the concatenation of one optional determiner (expressed by the loop operator :star≤n, where n starts from 0 and ends by 1), followed by zero or more adjectives followed by a noun.

Output description part The output structure of an FST is constructed by collecting together the variables of the recognition part's basic edges followed by some specific construction handlers. In order to support re-usability of FST to other applications, it is important to separate the construction handlers from the FST definition. Therefore, the output description part is realized through a function called BUILD-ITEM which receives as input the edge variables and a symbol denoting the class of the FST. For example, if :np is used as a type name for nominal phrases then the output description of the above NP-recognition part is

```
(build-item :type :np :out (list det adj noun)).
```

The function BUILD-ITEM then discriminates according to the specified type and constructs the desired output to some pre-defined requests (note, that in the above case the variables DET and ADJ might have received no token. In that case their default value NIL is used as an indication of this fact). Using this mechanism it is possible to define or re-define the output structure without changing the whole FST.

Special edges There exist some special basic edges namely (:var var), (:current-pos pos) and (:seek name var). The edge (:var var) is used for simply skipping or consuming a token without any checks. The edge :current-pos is used for storing the position of the current token in the variable pos, and the edge :seek is used for calling the FST named name, where var is used as a storage for the output of name. This is similar to the :seek edge known from Augmented Transition Networks with the notably distinction that in our system recursive calls are *disallowed.* Thus :seek can also be seen as a macro expanding operator. The :seek mechanism is very useful in defining modular grammars, since it allows for a hierarchical definition of finite state grammars, from general to specific constructions (or vice versa). The following example demonstrates the use of these special edges:

```
(compile-regexp
 (:conc
  (:current-pos start)
  (:alt
```

[4] Brill reports a 96% accuracy using a training set of 350.000 words and 1729 rules. However, he does not handle unknown words. In (Aone and Hausman, 1996), an extended version of Brill's tagger is used for tagging Spanish texts, which includes unknown words. They report an accuracy of 92.1%.

```
    (:seek time-phase time)
    (:conc
      (:star≤n (:seek time-expr-vorfield vorfield) 1)
      (:seek mona-time time)))
  (:current-pos end))
:name time-expr
:output-desc
  (build-item :type time-expr :start start
              :end end :out (list vorfield time))))
```

This FST recognizes expressions like "spätestens um 14:00 h" (*by two o'clock at the latest*) with the output description ((:out (:time-rel . "spaet") (:time-prep . "um") (:minute . 0) (:hour . 14)) (:end . 4) (:start . 0) (:type . time-expr))

Interface to TDL The interface to TDL, a typed feature-based language and inference system is also realized through basic edges. TDL allows the user to define hierarchically-ordered types consisting of type constraints and feature constraints, and has been originally developed for supporting high-level competence grammar development.

In SMES we are using TDL for two purposes:

1. defining domain-specific type lattices
2. expressing syntactic agreement constraints

The first knowledge is used for performing concept-based lexical retrieval (e.g., for extracting word forms which are compatible to a given super-type, or for filtering out lexical readings which are incompatible wrt. a given type), and the second knowledge is used for directing fragment processing and combination, e.g., for filtering out certain un-grammatical phrases or for extracting phrases of certain syntactic type.

The integration of TDL and finite state expressions is easily achieved through the definition of basic edges. For example the edge

(:mona-cat-type (:and "n" "device") var)

will accept a word form which has been analyzed as a noun and whose lexical entry type identifier is subsumed by "device". As an example of defining agreement test consider the basic edge

(:mona-cat-unify "det"
"[(num %1)(case %2 = gen-val) (gender %3)]"
agr det)

which checks whether the current token is a determiner and whether its inflection information (computed by MONA) unifies with the specified constraints (here, it is checked whether the determiner has a genitive reading, where structure sharing is expressed through variables like %1). If so, agr is bound to the result of the unifier and token is bound to det. If in the same FST a similar edge for noun tokens follows which also makes reference to the variable agr, the new value for agr is checked with its old value. In this way, agreement information is propagated through the whole FST.

An important advantage of using TDL in this way is that it supports the specification of very compact and modular finite expressions. However, one might argue that using TDL in this way could have dramatic effects on the efficiency of the whole system, if the whole power of TDL would be used. In some sense this is true. However, in our current system we only allow the use of type subsumption which is performed by TDL very efficiently, and constraints used very carefully and restrictively. Furthermore, the TDL interface opens up the possibility of integrating deeper processing components very straightforwardly.

Control parameters In order to obtain flexible control mechanisms for the matching phase it is possible to specify whether an exact match is requested or whether an FST should already succeed when the recognition part matches a prefix of the input string (or suffix, respectively). The prefix matching mechanism is used in conjunction with the Kleene :star and the identity edge :var, to allow for searching the whole input stream for extracting all matching expressions of an FST (e.g., extracting all NP's, or time expressions). For example the following FST extracts all genitive NPs found in the input stream and collects them in a list :

```
(compile-regexp
  (:star
    (:alt
      (:seek gen-phrase x)
      (:var dummy)))
  :output-desc (build-item :type list :out x )
  :prefix T
  :suffix NIL
  :name gen-star)
```

Additionally, a boolean parameter can be used to specify whether longest or shortest matches should be prefered (the default is longest match, see also (Appelt et al., 1993) where also longest subsuming phrases are prefered).

5 Fragment combination and template generation

Bidirectional shallow parsing The combination of extracted fragments is performed by a lexical-driven bidirectional shallow parser which operates on *fragment combination patterns* FCP which are attached to lexical entries (mainly verbs). We call these lexical entries *anchors*.

The input stream for the shallow parser consists of a *double-linked* list of all extracted fragments found in some input text, all punctuation tokens and text tokens (like newline or paragraph) and all found anchors (i.e., all other tokens of the input text are ignored). The shallow parser then applies for each anchor its associated FCP. An anchor can be viewed as splitting the input stream into a left and right input part. Application of an FCP then starts directly from the input position of the anchor and searches the left and right input parts for candidate fragments. Searching stops either if the beginning or the end of a text has been reached or if some punctuation, text tokens or other anchors defined as stop markers have been recognized.

General form of fragment combination patterns A FCP consists of a unique name, an recognition part applied on the left input part and one for the right input part, an output description part and a set of constraints on the type and number of collected fragments. As an prototypical case, consider the following FCP defined for intransitive verbs like *to come* or *to begin*:

```
(compile-anchored-regexp
 ((:set (cdr (assoc :start ?*)) anchor-pos)
  (:set ((:np (1 1) (nom-val (1 1)))) nec)
   (:set ((:tmp (0 2))) opt))
 ((:dl-list-left
   (:star
    (:alt
     (:ignore-token (“,” “;”))
     (:ignore-fragment :type (:time-phase :pp))
     (:add-nec (:np :name-np)
              :np nec lcompl)
     (:add-opt (:time-expr :date-expr)
              :tmp opt lcompl))))
 (:dl-list-right
   (:star
    (:alt
     (:ignore-token (“,” “;”))
     (:add-nec (:np) :np nec rcompl)
     (:add-opt (:time-expr :date-expr)
              :tmp opt rcompl)))))
 :name intrans
 :output-desc (build-item :type :intrans
              :out (list anchor-pos lcompl rcompl)))
```

The first list remembers the position of the active anchor and introduces two sets of constraints, which are used to define restrictions on the type and number of necessary and optional fragments, e.g., the first constraint says that exactly one :np fragment (expressed by the lower and upper bound in (1 1)) in nominative case must be collected, where the second constraint says that at most two optional fragments of type :tmp can be collected. The two constraints are maintained by the basic edges :add-nec and :add-opt. :add-nec performs as follows. If the current token is a fragment of type :np or :name-np then inspect the set named nec and select the constraint set typed :np . If the current token agrees in case (which is tested by type subsumption) then push it to lcompl and reduce the upper bound by 1. Since next time the upper bound is 0 no more fragments will be considered for the set nec.[5] In a similar manner :add-opt is processed.

The edges :ignore-token and :ignore-fragment are used to explicitly specify what sort of tokens will not be considered by :add-nec or :add-opt. In other words this means, that each token which is not mentioned in the FCP will stop the application of the FCP on the current input part (left or right).

Complex verb constructions In our current system, FCPs are attached to main verb entries. Expressions which contain modal, auxiliary verbs or separated verb prefixes are handled by lexical rules which are applied after fragment processing and before shallow processing. Although this mechanism turned out to be practical enough for our current applications, we have defined also complex verb group fragments VGF. A VGF is applied after fragment processing took place. It collects all verb forms used in a sentence, and returns the underlying dependency-based structure. Such an VGF is then used as a *complex anchor* for the selection of appropriate fragment combination patterns as described above. The advantage of verb group fragments is that they help to handle more complex constructions (e.g., time or speech act) in a more systematic (but still shallow) way.

Template generation An FCP expresses restrictions on the set of candidate fragments to be collected by the anchor. If successful the set of found fragments together with the anchor builds up an instantiated template or frame. In general a template is a record-like structure consisting of features and their values, where each collected fragment and the anchor builds up a feature/value pair. An FCP also defines which sort of fragments are necessary or optional for building up the whole template. FCPs are used for defining linguistically oriented general head-

[5] In some sense this mechanism behaves like the subcategorization principle employed in constraint-based lexical grammars.

modifier construction (linguistically based on dependency theory) and application-specific database entries. The "shallowness" of the template construction/instantiation process depends on the weakness of the defined FST of an FCP.

A major drawback of our current approach is that necessary and optional constraints are defined together in one FCP. For example, if an FCP is used for defining generic clause expressions, where complements are defined through necessary constraints and adjuncts through optional constraints then it has been shown that the constraints on the adjuncts can change for different applications. Thus we actually lack some modularity concerning this issue. A better solution would be to attach optional constraints directly with lexical entries and to "splice" them into an FCP after its selection.

6 Coverage of knowledge sources

The lexicon in use contains more than 120.000 stem entries (concerning morpho-syntactic information).

The time and date subgrammar covers a wide range of expressions including nominal, prepositional, and coordinated expressions, as well as combined date-time expressions (e.g., "vom 19. (8.00 h) bis einschl. 21. Oktober (18.00 h)" yields: (:pp (from :np (day . 19) (hour . 8) (minute . 0)) (to :np (day . 21) (month . 10) (hour . 18) (minute . 0))))

The NP/PP subgrammars cover e.g., coordinate NPs, different forms of adjective constructions, genitive expressions, pronouns. The output structures reflects the underlying head-modifier relations (e.g., " Die neuartige und vielfältige Gesellschaft " yields: (((:sem (:head "gesellschaft") (:mods "neuartig" "vielfaeltig") (:quantifier "d-det")) (:agr nom-acc-val) (:end . 6) (:start . 1) (:type . :np)))

30 generic syntactic verb subcategorization frames are defined by fragment combination patterns (e.g, for transitive verb frame). Currently, these verb frames are handled by the shallow parser with no ordering restriction, which is reasonably because German is a language with relative free word order. However, in future work we will investigate the integration of shallow linear precedence constraints.

The specification of the current data has been performed on a tagged corpora of about 250 texts (ranging in size from a third to one page) which are about event announcement, appointment scheduling and business news following a bottom-up grammar development approach.

7 Current applications

On top of SMES three application systems have been implemented:

1. appointment scheduling via email: extraction of co-operate act, duration, range, appointment, sender, receiver, topic

2. classification of event announcements sent via email: extraction of speaker, title, time, and location

3. extraction of company information from newspaper articles: company name, date, turnover, revenue, quality, difference

For these applications the main architecture (as described above), the scanner, morphology, the set of basic edges, the subgrammars for time/date and phrasal expressions could be used basically unchanged.

In (1) SMES is embedded in the COSMA system, a German language server for existing appointment scheduling agent systems (see (Busemann et al., 1997), this volume, for more information). In case (2) additional FST for the text structure have been added, since the text structure is an important source for the location of relevant information. However, since the form of event announcements is usually not standardized, shallow NLP mechanisms are necessary. Hence, the main strategy realized is a mix of text structure recognition and restricted shallow analysis. For application (3), new subgrammars for company names and currency expressions have to be defined, as well as a task-specific reference resolution method.

Processing is very robust and fast (between 1 and 10 CPU seconds (Sun UltraSparc) depending on the size of the text which ranges from very short texts (a few sentences) upto short texts (one page)). In all of the three applications we obtained high coverage and good results. Because of the lack of comparable existing IE systems defined for handling German texts in similar domains and the lack of evaluation standards for the German language (comparable to that of MUC), we cannot claim that these results are comparable.

However, we have now started the implementation of a new application together with a commercial partner, where a more systematic evaluation of the system is carried out. Here, SMES is applied on a quite different domain, namely news items concerning the German IFOR mission in former Yugoslavia. Our task is to identify those messages which are about violations of the peace treaty and to extract the information about location, aggressor, defender and victims.

The corpus consists of a set of monthly reports (Jan. 1996 to Aug. 1996) each consisting of about 25 messages from which 2 to 8 messages are about fighting actions. These messages have been hand-tagged with respect to the relevant information. Although we are still in the development phase we will briefly describe our experience of adapting SMES to this new domain. Starting from the assumption that the core machinery can be used un-changed we first measured the coverage of the existing linguistic knowledge sources. Concerning the above mentioned corpus the lexicon covers about 90%. However, from the 10% of unrecognized words about 70% are proper names (which we will handle without a lexicon) and 1.5% are spelling errors, so that the lexicon actually covers more then 95% of this unseen text corpus. The same "blind" test was also carried out for the date, time, and location subgrammar, i.e., they have been run on the new corpus without any adaption to the specific domain knowledge. For the date-/time expressions we obtained a recall of 77% and a precision of 88%, and for the location expressions we obtained 66% and 87%, respectively. In the latter case, most of the unrecognized expressions concern expressions like "nach Taszar/Ungarn", "im serbischen bzw. kroatischen Teil Bosniens", or "in der Moslemisch-kroatischen Föderation". For the general NP and PP subgrammars we obtained a recall of 55% and a precision of 60% (concerning correct head-modifier structure). The small recall is due to some lexical gap (including proper names) and unforeseen complex expressions like "die Mehrzahl der auf 140.000 geschätzten moslemischen Flüchtlinge". But note that these grammars have been written on the basis of different corpora.

In order to measure the coverage of the fragment combination patterns FCP, the relevant main verbs of the tagged corpora have been associated with the corresponding FCP (e.g., the FCP for transitive verbs), without changing the original definition of the FCPs. The only major change to be done concerned the extension of the output description function BUILD-ITEM for building up the new template structure. After a first trial run we obtained an unsatisfactory recognition rate of about 25%. One major problem we identified was the frequent use of passive constructions which the shallow parser was not able to process. Consequently, as a first actual extension of SMES to the new domain we extended the shallow parser to cope with passive constructions. Using this extension we obtained an recognition of about 40% after a new trial run.

After the analysis of the (partially) unrecognized messages (including the misclassified ones), we identified the following major bottlenecks of our current system. First, many of the partially recognized templates are part of coordinations (including enumerations), in which case several (local) templates share the same slot, however this slot is only mentioned one time. Resolving this kind of "slot sharing" requires processing of elliptic expressions of different kinds as well as the need of domain-specific inference rules which we have not yet foreseen as part of the core system. Second, the wrong recognition of messages is often due to the lack of semantic constraints which would be applied during shallow parsing in a similar way as the subcategorization constraints.

Although these current results should and can be improved we are convinced that the idea of developing a core IE-engine is a worthwhile venture.

8 Related work

In Germany, IE based on innovative language technology is still a novelty. The only groups which we are aware of which also consider NLP-based IE are (Hahn, 1992; Bayer et al., 1994). None of them make use of such sophisticated components, as we do in SMES. Our work is mostly influence by the work of (Hobbs, 1992; Appelt et al., 1993; Grishman, 1995) as well as by the work described in (Anderson et al., 1992; Dowding et al., 1993).

9 Conclusion

We have described an information extraction core system for real world German text processing. The basic design criterion of the system is of providing a set of basic powerful, robust, and efficient natural language components and generic linguistic knowledge sources which can easily be customized for processing different tasks in a flexible manner. The main features are: a very efficient and robust morphological component, a powerful tool for expressing finite state expressions, a flexible bidirectional shallow parser, as well as a flexible interface to an advanced formalism for typed feature formalisms. The system has been fully implemented in Common Lisp and C.

Future research will focus towards automatic adaption and acquisition methods, e.g., automatic extraction of subgrammars from a competence base and learning methods for domain-specific extraction patterns.

10 Acknowledgement

The research underlying this paper was supported by research grants from the German Bundesministerium für Bildung, Wissenschaft, Forschung und Technologie (BMBF) to the DFKI projects PARADICE, FKZ ITW 9403 and PARADIME, FKZ ITW 9704. We would like to thank the following people for fruitful discussions: Hans Uszkoreit, Gregor Erbach, and Luca Dini.

References

A. Aho, J. Hopcraft, and J. Ullmann. 1983. *Data structures and algorithms*. Addison Wesley, Reading, Mass.

P. Anderson, P. Hays, A. Huettner, L. Schmandt, I. Nirenburg, and S. Weinstein. 1992. Automatic extraction of facts from press releases to generate news stories. In *3rd ANLP*, pages 170–177, Trento, Italy.

C. Aone and K. Hausman. 1996. Unsupervised learning of a rule-based Spanish part of speech tagger. In *Proceedings of COLING-96*, pages 53–58, Kopenhagen, Denmark, Europe.

D. Appelt, J. Hobbs, J. Bear, D. Israel, and M. Tyson. 1993. Fastus: A finite state processor for information extraction from real world text. In *Proceedings of the 13th IJCAI*, Chambery, France, August.

T. Bayer, U. Bohnacker, and H. Mogg-Schneider. 1994. Infoportlab – an experimental document understanding system. In *Proceedings of the 1st DAS*.

E. Brill. 1995. Unsupervised learning of disambiguation rules for part of speech tagging. In *Very Large Corpora Workshop*.

S. Busemann, T. Declereck, A. Diagne, L. Dini, J. Klein, and S. Schmeier. 1997. Natural language dialogue service for appointment scheduling agents. This volume.

N. Chinchor, L. Hirschman, and D. Lewis. 1993. Evaluating message understanding systems: An analysis of the third message understanding conference (muc-3). *Computational linguistics*, 19(3).

J. Cowie and W. Lehnert. 1996. Information extraction. *Communications of the ACM*, 39(1):51–87.

J. Dowding, J. Gawron, D. Appelt, J. Bear, L. Cherny, R. Moore, and D. Moran. 1993. Gemini: A natural language system for spoken-language understanding. In *31th ACL*, Ohio.

W. Finkler and G. Neumann. 1988. Morphix: A fast realization of a classification–based approach to morphology. In H. Trost, editor, *Proceedings of 4th ÖFAI*, Berlin, August. Springer.

R. Grishman and B. Sundheim. 1996. Message Understanding Conference – 6: A Brief History. In *Proceedings of COLING-96*, pages 466–471, Kopenhagen, Denmark, Europe.

R. Grishman. 1995. The NYU MUC-6 System or Where's the Syntax? In *Sixth Message Understanding Conference (MUC-6)*. Morgan Kaufmann, November.

U. Hahn. 1992. On text coherence parsing. In *Proceedings of COLING-92*, pages 25–31, Nantes, France, Europe.

J. Hobbs. 1992. The generic information extraction system. In B. Sundheim, editor, *Fourth Message Understanding Conference (MUC-4)*, McLean, Virginia, June. Distributed by Morgan Kaufmann Publishers, Inc.,San Mateo, California.

K. Koskenniemi. 1983. Two-level model for morphological analysis. In *8th IJCAI*, pages 683–685, Karlsruhe.

Hans-Ulrich Krieger and Ulrich Schäfer. 1994. *TDL*—a type description language for constraint-based grammars. In *Proceedings of COLING-94*, pages 893–899.

Hans-Ulrich Krieger. 1987. Nil—eine Lisp-basierte natürlichsprachliche Schnittstelle zu Ramses. Unterstützung der NMR-Diagnostik von Hirn- und Mammatumoren. Master's thesis, RWTH Aachen.

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