

# A Generic Architectural Framework for Text Knowledge Acquisition\*

Alexander Mädche<sup>1</sup>, Günter Neumann<sup>2</sup> & Steffen Staab<sup>1</sup>

<sup>1</sup>AIFB, Univ. Karlsruhe  
{maedche, staab}@aifb.uni-karlsruhe.de

<sup>2</sup>DFKI, Saarbrücken  
neumann@dfki.de

May 31, 1999

## Abstract

As knowledge engineering matures and the amount of textual resources grows, the pressure for (semi-)automatic acquisition of knowledge structures from texts increases drastically. With automatic text understanding techniques still being far from perfect, the best solution one may offer is a framework that allows for flexible integration of a large variety of techniques into one text knowledge acquisition environment. Thus, we here present an architecture that integrates seminal, but rather isolated accounts for text knowledge acquisition into the MIKE knowledge engineering framework. Building on a survey of methods for text knowledge acquisition, we devise a scheme that lends itself to the acquisition problem, in general, and to the construction of natural-language applications, in particular.

## 1 Introduction

Methodologies and corresponding support tools for building knowledge-based systems along the lines of CommonKADS (Schreiber et al., 1994), MIKE (Angele et al., 1998), and their likes, have shown their usefulness in recent years. The philosophy underlying these approaches relies on lucent transitions between different steps of the knowledge-based system construction process — the transitions being easy as seen against the overall task of building the system. In particular, MIKE fosters even an intermediate, semi-formal model, the so-called *structure model*, against which the behavior of the system may be compared in subsequent steps of design and implementation. However, the transition that is often crucial is the one from texts containing expertise knowledge in form of protocols and expository texts towards the structure model.

This transition is handled in MIKE, for example, by the manual structuring of knowledge protocols, yielding the *elicitation model*. Dependencies between text fragments are explicitly given by the knowledge engineer in order to bridge the large gap between texts and (semi-)formal knowledge. Thus, the elicitation model helps with clarifying overall text structures and with identifying blocks of informal knowledge relevant to larger blocks of conceptual or task structures. The disadvantage is that it works on a purely manual basis and all but pinpoints to the target structures the knowledge engineer must discover and describe.

Hence, several statistics- and computational linguistics-based tools that support the knowledge engineer in the task of identifying important concepts and relations in texts have been proposed in recent years (cf., e.g., Assadi (1998), Biébow & Szulman (1999), Barker et al. (1998), Lapalut (1996), Hahn & Schnattinger (1997), Hahn & Schnattinger (1998), Szpakowicz (1990)) and the difficulty of the general problem we have just outlined has given them a considerable momentum. In spite of their seminal contribution on the methodological side, they lack a coherent integration into frameworks for building knowledge-based systems, such as CommonKADS or MIKE. Thus, they yield comparatively ideosyncratic approaches that do not fit smoothly into the knowledge-based system construction process nor do they collaborate easily with similar tools. This latter

---

\*Contact author: S. Staab, E-Mail: staab@aifb.uni-karlsruhe.de, Tel.: +49-721-6087363, Fax.: +49-721-693717

integration problem appears as a large drawback to us, since techniques in the computational linguistics community are constantly being improved, amended, or newly invented, while no single one of them seems to solve the general text understanding problem. Hence, we believe that a diversity of techniques should be employed to support the knowledge engineer with his task of identifying and defining relevant concepts, roles and tasks in text, *i.e.* with the *text knowledge acquisition process*.

For this purpose, we here present a general architectural framework that embeds tools employing statistics, linguistics and background knowledge in a cyclic process of discovering concepts and relations from texts. We offer integrated information facilities to the knowledge engineer, such as term extraction (Neumann et al., 1997), mixed linguistics and conceptual reasoning (Hahn & Schnattinger, 1998) and concept formulation (Michalski et al., 1986). For this "boosting" approach to text knowledge acquisition it is of especial importance that the structure model is built incrementally. The simplest techniques such as term extraction may be applied first and more refined algorithms such as ones for analogical reasoning may be used, once a significant amount of background knowledge exists. Our framework is integrated within the MIKE methodology and an integration into the MikeTool, which supports the construction of the knowledge-based system, is underway.

Finally, we want to consider a particular type of knowledge-based systems, *viz.* ones that handle natural language. The reason is that in the process of acquiring domain specific knowledge from texts it is necessary to extend our language understanding facilities towards an incorporation of new lexical, and possibly grammatical, knowledge. Thus, in the process of acquisition we also prepare the ground for improved natural language capabilities that may help the knowledge-based system in serving the natural language needs of its users – a "side product" that constitutes a considerable value on its own.

In this paper, we will first give a brief introduction to the MIKE approach, from which we will also derive some shortcomings (Section 2). Then we present a survey of tools for knowledge discovery from texts including methods based on linguistics, statistics and knowledge structures (Section 3). Subsequently, we describe a generic architectural framework that integrates these methods (Section 4). Thus, we allow a tool-supported construction of the structure model of MIKE, which will be illustrated in an example that enters in the midst of such a process (Section 5). Finally, we describe how the side products of text knowledge acquisition will be used for a system with natural language capabilities in a reversal of the knowledge-based system construction cycle in an *application cycle* (Section 6).

## 2 MIKE as a Methodology for Knowledge Engineering

MIKE (Model-based and Incremental Knowledge Engineering) is an approach for developing knowledge-based systems that integrates semi-formal and formal specification techniques and prototyping into a coherent engineering framework (cf. Angele et al. (1998)). All activities in the construction process of a knowledge-based system are embedded in a cyclic process model. Similar to software engineering where a specification of a system should be clearly separated from its design and implementation, the distinction between symbol level and knowledge level is necessary for the description of a knowledge-based system. While the symbol level model deals with design and implementation of a system, the knowledge level model specifies the functionality of a system and describes how the system provides functionality at a high level of abstraction. According to the CommonKADS approach (cf. Schreiber et al. (1994)) the knowledge level description of a knowledge-based system is called model of expertise. This model separates different kinds of knowledge at different layers. The *domain layer* contains knowledge about domain-specific concepts, their attributes and their relationships. The *inference layer* contains the knowledge about the functional behavior of the problem-solving process. The *task layer* (called *control layer* in MIKE) contains knowledge about the goals of a task and the control knowledge required to perform the task.

The MIKE approach splits the KADS model of expertise in a semi-formal and a formal part.

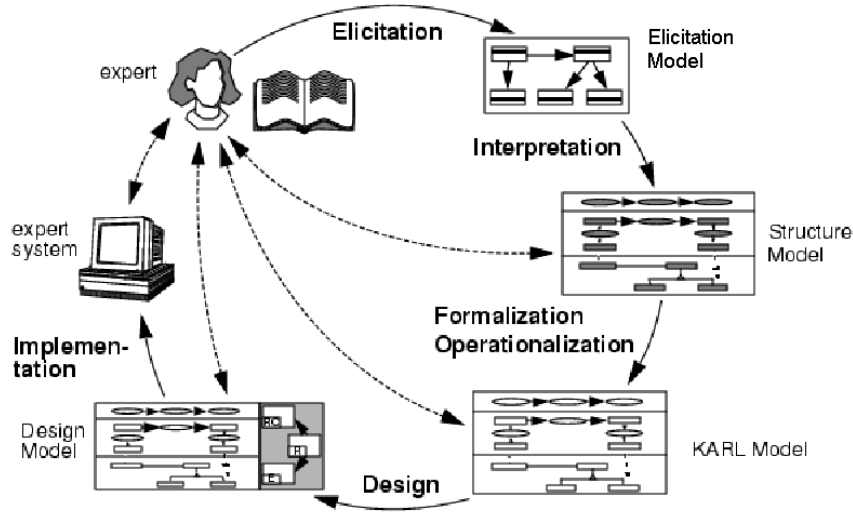


Figure 1: Architecture of the MIKE development process

For the semi-formal knowledge representation a hypertext-based formalism serves as a communication basis between expert and knowledge engineer during knowledge acquisition. The semi-formal knowledge representation is also the basis for formalization, resulting in a formal and executable model specified in the Knowledge Acquisition and Representation Language (KARL) (cf. Fensel et al. (1998)). Since KARL is executable, the formal model of expertise can be developed and validated by prototyping. A smooth transition from a semi-formal to a formal specification and further on to design is achieved because all the description techniques rely on the same conceptual model to describe the functional and non-functional aspects of the system. Thus, the system is thoroughly documented at different description levels, each level focuses on a distinct aspect of the entire development effort. Let us now consider the development process in more detail.

**The MIKE development process.** The different phases and resulting models in the development process have to bridge the large gap between the informal requirements and human knowledge on the one hand and the final realization of the knowledge-based system on the other hand. In order to reduce this complexity the entire development process is divided into a number of phases: elicitation, interpretation, formalization and operationalization, design, and implementation. Each of these phases results in different models that describe knowledge at different description levels. The development architecture distinguishes different models: *elicitation model*, *structure model*, *KARL model* and *design model*. Figure 1 depicts the whole MIKE development process.

The MIKE model of expertise is represented in three different layers starting with or following the structure model. Since our focus will be mainly on the steps between texts and the *structure model* we will neglect subsequent stages and briefly elaborate on the description primitives of the structure model.

The knowledge acquisition process starts with elicitation using methods like structured interviews, observation, structuring techniques etc. (cf. Eriksson (1992)). The resulting knowledge expressed in natural language is stored in so-called knowledge protocols. Knowledge protocols may be connected to each other by using two types of links: an ordering link specifies the temporal order in which two protocols have been collected, whereas a refinement link indicates that one protocol is a refinement of another. The result is called *elicitation model*. During the interpretation phase the knowledge structures that may be identified in the knowledge protocols are represented in the semi-formal variant of the model of expertise: the *structure model*. All structuring information

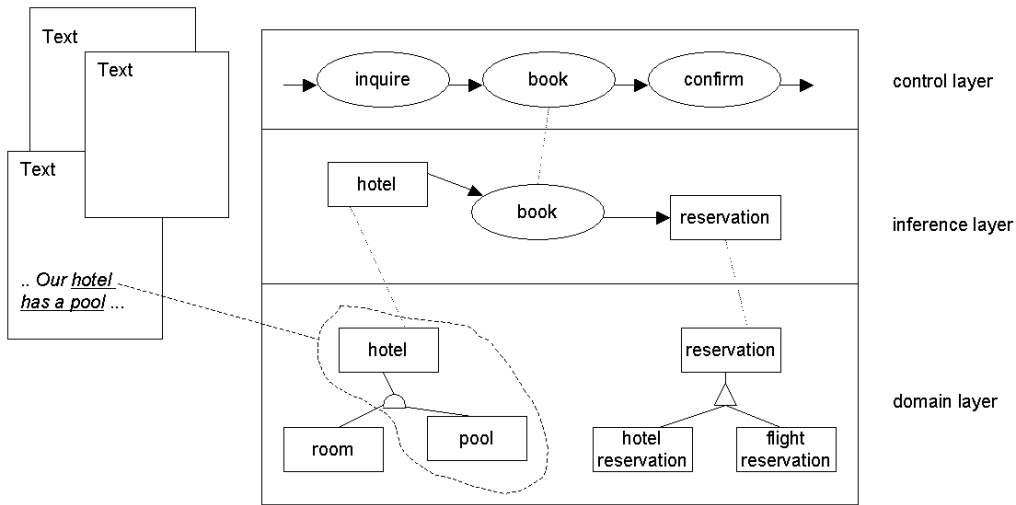


Figure 2: Example for the structure model in MIKE

in this model is expressed in a fixed, restricted language while the basic building blocks are represented by unrestricted texts. This representation provides an initial structured description of emerging knowledge structures. The structure model is described in a hypertext-based formalism offering various types of nodes and links. Each node is inscribed by a natural language text. On the *domain layer* nodes represent domain concepts and links represent various types of relationships between concepts, like generalization, aggregation and domain-specific relationships. On the *inference layer* nodes represent the inference steps and knowledge roles. The control between the inferences is defined at the *control layer*. In general, sequence, loop and alternative are used to specify the control flow. The semi-formal structure model is the foundation for the phases formalization and operationalization which result in the formal model of expertise: the *KARL model*.

Figure 2 depicts an example extract for a structure model in the tourism domain. The domain terminology and domain knowledge needed to organize and plan holidays are defined at the domain layer. The parts of a hotel are represented in a PARTOF-hierarchy. The different types of reservations are modeled within an ISA-hierarchy. On the inference layer nodes represent the inference steps and knowledge roles. In this example the global inference action "plan your holidays" is decomposed into several top-level inference actions including "book your hotel". This action is tied to a specific hotel and yields a specific reservation. The control flow of the global inference action is defined at the control layer. In our example the top-level inference actions are executed once in the order "inquire", "book", "confirm".

**Deficits.** MIKE is tuned to consider expert knowledge, but does not really support extracting information from available documents or from examples. The characteristics of the latter are that knowledge cannot easily be attributed to a single piece of text, but must rather be seen as being implicit in a collection of texts.

Also the step between natural language and a structural repository of a domain is still a rather large one. The meaning of a new concept often only appears when seen in connection with the context where it is used. Finding these places in the texts is a task that should be automated in order to ease the burden on the knowledge engineer. The hypertext-based representation helps in latter construction steps, but does not really facilitate the creation of the hypermedia structures right at the beginning.

### 3 Text Analysis Dimensions

If we want to define concepts on the basis of their use in natural texts we have to identify those textual fragments in a collection of texts which serve as *paraphrases* of these concepts. In other words we have to identify the set of text fragments that expresses the meaning of a concept. If successful we can (more or less) directly define a mapping from expressions in the texts to their corresponding concept representation. For this purpose, one must face several principal problems:

- In the initial phase of the text knowledge acquisition cycle no concepts are known. Thus the system should be able to identify (possibly disjunct) clusters of sets of paraphrases which describe concepts with regard to their similarities — or even better — with regard to their structural relationships. If such knowledge can be acquired incrementally, the text knowledge acquisition system may use this representation as additional guidance in the course of the further text knowledge acquisition process.
- A lot of paraphrases of a concept are not known to the system before or during the acquisition cycle. Thus we need some sort of prediction mechanism: if a system recognizes a text fragment to denote a concept it must decide to which of the known or to which new concept the fragment should be mapped.
- Text fragments are potentially ambiguous, and their meaning may only be understood with regard to a certain context or domain. Thus if the system detects a text fragment that previously denoted a particular concept this does not necessarily imply that it has found another paraphrase of this very same concept.

How may we define useful criteria for recognizing paraphrases and how can we identify those text expressions which count as paraphrases of some conceptual structures? It seems obvious that we might consider different technical dimensions for this task that have been developed over the past decades. Hence, our tool set comprises

- the extraction of a plentitude of linguistic structure from real world text,
- the broad collection of statistical information for the determination of relevance at all levels of understanding,
- the extensive usage and accumulation of background knowledge that might restrict the choice of adequate conceptual structures in the further course of text knowledge acquisition.

We now give an overview of principal methods in each of these three categories and reflect on how they are employed in state-of-the-art text knowledge acquisition tools. The reader may become aware of the broad range of possibilities that may arise from integrations of techniques in this wide-spread spectrum — a range that urgently demands for a generic framework, such as will be given in Section 4.

#### 3.1 Shallow Text Processing

In theory, it might be possible to use an exhaustive and deep general text understanding system that would try to cope with the full complexities of a language and that would build all conceptual structures required for a knowledge-based system. However, even if there were the possibility to formalize and represent the complete grammar of a natural language, the system would still need a very high degree of robustness and efficiency in order to be able to process a large number of real-world texts, such as web pages. Past experiences have convinced most people that such a system is not feasible within the next few years.

However, in order to fulfill the ever increasing demands for improved processing of real-world texts, natural language researchers have turned from a theoretical view of the overall problem, which aimed at a complete solution in the distant future towards more practical approaches that are less

comprehensive. This has led to so-called “shallow” text processing approaches, which deliver the requested robustness and efficiency. These approaches neglect certain generic language regularities which are known to cause complexity problems (*e.g.*, instead of computing all possible readings only an underspecified structure is computed) or handle them very pragmatically, *e.g.*, by restricting the depth of recursion on the basis of a corpus analysis or by making use of heuristic rules, like “longest matching substrings”. This engineering view on language has led to a renaissance and improvement of well-known efficient techniques, most notably finite state technology for parsing.

Thus, we here rely on a general architecture for shallow text processing. The structure and functionality is drawn from common properties found in almost all recent approaches that deal with real-world text (see also Hobbs (1992), Chinchor et al. (1993), Appelt et al. (1993), Grishman & Sundheim (1996), MUC7 (1998)).

The basic design criterion of such a general 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 to process different domain-specific tasks in a flexible manner. We distinguish three primarily levels of processing, *viz.* the word level, the phrase level (which also includes the sentence level), and the text level. The three levels are dealt with by following shallow text processing tools listed below and including a preprocessing step at the beginning.

**Text Tokenizer.** Each text must first be preprocessed by the text scanner. Applying regular expressions, the text scanner identifies some text structures (*e.g.*, tables, 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-value structures.

**Lexical Analysis.** Each token that has been identified as a potential word-form is processed by the lexical analysis. Lexical analysis includes morphological analysis (*i.e.*, the identification of the canonical common stem of a set of related word forms), recognition of compounds, retrieval of lexical information (where retrieval supports robust processing through component-based substring matching), and part-of-speech tagging that performs word-based disambiguation. The capability for efficient processing of compounds is crucial since compounding is a very productive process in German, which we consider the principal target language for our system. The output returned from the morphological analysis comprises the word form together with all its readings.

If we consider the lexical analysis of words in isolation — as is usually done —, each word is potentially ambiguous. In order to decrease the set of possible candidates for the following components, local and very efficient disambiguation strategies are applied such that implausible readings are filtered out. This is usually performed through part-of-speech taggers as well as through the application of case-sensitive rules<sup>1</sup>. The task of a part-of-speech tagger is to determine the unique part-of-speech of a current word in its current context using local rules (see Brill (1993)).

**Chunk Parser.** Parsing is performed by a chunk parser (cf. Abney (1996), Neumann et al. (1997)) and is usually subdivided into two major components. In the first step, phrasal fragments are recognized, like general nominal phrases, verb groups or specialized complex expressions for time, date, and named entities. The structure of potential phrasal fragments is defined using (weighted) finite state transducers (WFST, cf. Neumann et al. (1997)). In the second stage, the dependency-based structures of the fragments of each sentence are analyzed using a set of specific sentence patterns. These patterns are also expressed by means of WFST. Finally, the grammatical functions (such as subject, direct-object) are determined for each dependency-based structure on the basis of a large subcategorization lexicon.

---

<sup>1</sup>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.

**Discourse Parser.** The discourse parser must detect the textual cohesion between sentences and, in particular, it must establish coreference between phrases. For instance, adjacent phrases such as (1) and (2)

- (1) The hotel offers 45 luxury rooms.
- (2) It also has a splendid bar overlooking the ocean.

should be analyzed such that the hotel is linked to the luxury rooms as well as to the splendid bar. For our toolset the discourse parser must yet be implemented, probably along the lines given by (Baldwin et al., 1998).

It is important that the system can store each partial result on each level of processing in order to maximize the contextual information available. We will call the knowledge pool that maintains all partial results computed by the shallow text processor a *text chart*. Each component outputs the resulting structures uniformly as feature value structures, together with its type (e.g., date token, noun (N), adjective (Adj), proper name (PN), nominal phrase (NP) or verb group (VG) etc.) and the corresponding start and end positions of the spanned input expressions.

The different kinds of index information computed by the individual components (e.g., text position, reduced word form, category, phrase) allow a uniform, flexible and efficient access to each individual partial result. This avoids redundant computations and it delivers rich contextual information for the purposes of disambiguation or the handling of unknown constructs. Furthermore, it provides the basis for the automatic computation of hyperlinks depicting text fragments and their corresponding internal linguistic information.

## 3.2 Statistical Approaches

The shallow text processor thus described can “only” compute linguistic information in the sense that it does not tell us what information is relevant for our domain modeling. However, we assume that the quantity and distribution of each information unit on every level of processing will tell us something about its relevance.

### 3.2.1 “Pure” Statistics

For a more lucid illustration of techniques we here consider (almost) “pure” statistical techniques such as often used in information retrieval (cf. Sparck-Jones & Willett (1997)). However one should be well aware that these methods are most often used in conjunction with simple linguistic methods, such as stemming.

**Frequency Measures** are often used to determine relevance. Though absolute frequencies may easily distort the picture towards closed word classes, weighted, relative counts like TF/IDF<sup>2</sup> (term frequency / inverse document frequency) face such issues successfully.

**Distributions** across a corpus of texts. Individual words exhibit a history of appearance. *I.e.*, information about when a certain word or phrase was introduced and how often it has been used in a certain context may help in defining corresponding conceptual structures. The assumption here is that a text fragment with a low frequency (relative to the complete set of documents) will have a high frequency for some time after it has been introduced. This information will be used as a basis for inducing domain-specific terms and conceptual regions.

---

<sup>2</sup>More precisely, the weight of a term is computed using  $w_{ij} = tf_{ij} * \log_2 \frac{N}{n}$ , where  $tf_{ij}$  is frequency of term  $T_j$  in document  $D_i$ ,  $N$  is the number of documents in the corpus, and  $n$  is the number of documents where term  $T_j$  occurs at least once (Salton, 1988).

**N-grams** or **collocations**. A collocation is a “recurrent combination of words that co-occur more often than expected by chance and that correspond to arbitrary word usages” (Smadja, 1993) and they are typically found by computing the probability of bigrams and trigrams (see also (Charniak, 1993)). Here, we are using N-grams in order to determine the mutual co-occurrence between text fragments (words or phrases) as a basis of finding contextual information.

**Text Clustering.** Pure statistical approaches will also be used for unsupervised classification. For example, AutoClass (Cheeseman & Stutz, 1995) is an unsupervised Bayesian classification system that seeks a maximum posterior probability classification. The inputs consist of a database of word vectors (*i.e.*, the frequencies of all words from the model in each document), and a class model. The system finds the set of classes that is maximally probable with respect to the data and model. The output is a set of class descriptions, and partial membership of the documents in the classes. This approach seems very promising for term extraction, in case linguistic structure computed by the shallow text processor is used for defining the attribute set and class model (also cf. Lapalut (1996)).

### 3.2.2 Combination of Statistics with Linguistic Methods

The computed structures on each level can uniformly be treated as streams of units. For example, the text scanner computes a stream of tokens, the morphology a stream of morphological readings, and the phrasal component of the chunk parser returns a stream of phrases. Each unit is represented as an item in the text chart (see also Section 3.1). At each level of processing, statistics may successfully amend linguistic information represented by these different types of streams of items.

The wide range of useful combinations includes frequency counts or terms instead of tokens. There may be a combination of TF/IDF techniques with morphology; statistics may also permeate the analysis of *linguistic context*. As the latter case is one of the more elaborate ones we may illustrate it here to indicate the range we intend to cover with our framework.

The algorithm is based on (Buitelaar, 1998) and computes the number of occurrences of nouns relative to their occurrence in the head-modifier structures determined by the chunk parser. This means that terms are classified strictly according to their linguistic context, that is, according to their distributions within certain linguistic constructions. Hence, one may find indicators for similarity between “Victorian” and “Neo-Classic” based on the observation that both terms modify “houses”. The algorithm runs as follows:

First, the shallow text analysis is performed. For each document tokenization, morphological analysis (including processing of nominal compounds), proper name extraction, and a syntactical analysis of general nominal and prepositional phrases are performed. All extracted nominal expressions are represented uniformly in form of head-modifier structures: for instance,

(3) “Victorian Neo-Classic houses”

yields:

(4) ((:head “houses”)(:mods “Victorian” “Neo-Classic”)(:quantifier “def-det”))

In a second step, TF-/IDF techniques may be applied in order to determine relevant nouns. Finally, the elements of the relevance list are further classified according to structural similarity.

A prototypical implementation of this approach has been realized in cooperation with our colleagues from the EU-funded project MIETTA, and has been applied in the domain of tourism. The quality of the relevance list directly corresponds to the quality of the extracted nominal terms computed by the shallow text processor so that the result is only a first approximation. However, it has already become apparent that the list of relevant terms can be used in order to improve the parsing of nominal terms in subsequent cycles of the text knowledge acquisition system. Hence, we hope that through a cyclic approach we might further improve the quality of the relevance of

nominal terms, *e.g.*, by integrating unsupervised statistical clustering approaches (like AutoClass, see above).

### 3.3 Knowledge Structures

Irrespective of linguistic patterns or statistical characteristics, the information given in the texts can only be understood within a conceptual framework that integrates all features into a comprehensive knowledge model. The necessity for the semi-automatic extension of the scope of this knowledge model given partial information has been widely accounted for — mostly in the area of machine learning. We here consider very briefly several approaches — without the intention of a complete enumeration — that contribute to this goal (Section 3.3.1), in particular when exploited in combination with other means (Section 3.3.2).

#### 3.3.1 Knowledge Structures as Seen in Isolation

The methods that are listed here cover very different aspects of concept formation and structuring, all of them seeming appropriate for the text knowledge acquisition task. The ones we list in this subsection have in common that they are restricted to the knowledge level only:

**Description Logics** facilitates the construction of an ISA-hierarchy by way of automatic *subsumption*. Description Logics<sup>3</sup> exploits logic inference in order to determine sub- or superconcept relationships. For instance, given the knowledge that **trains** travel on railroads and an **Intercity Express (ICE)** travels on railroads, but stops only in cities with a population of more than 50,000 then the reasoning engine may infer that **ICE** is a subconcept of **train** (cf. (Woods & Schmolze, 1992) for a survey).

**Conceptual Clustering** pursues a similar goal by constructing a hierarchy of clusters (cf. (Michalski et al., 1986) for a survey). In contrast to description logics, the mechanism does not build on logic implications, rather it clusters concepts that share many features together in the lower, more specific parts of the hierarchy, while dissimilar concepts only meet at the upper, more general level of concepts. For instance, a **local train** and an **ICE** may be clustered together in a superconcept (that could be named **train**), while **car** and **ICE** only appear together in a more general cluster (that could be named **vehicle**).

**Analogical Reasoning** may be used in two ways (cf. Veloso (1994)). On the one hand, one may use parallel relational structures to determine ISA- or PARTOF-hierarchies. On the other hand, one may use the knowledge about analogy to establish parallel relational structures. For instance, given the knowledge that **IBM** provides **AIX** and **SUN** provides **Solaris**, **AIX** is an operating system and **IBM** and **SUN** are both computer companies then this may indicate that **Solaris** is an operating system, too. Conversely, given that the relation between **IBM** and **AIX** is analogous to the one between **SUN** and **Solaris**, and that one may infer that **IBM** provides **AIX** then this is a strong indication that **SUN** provides **Solaris**.

Regarding our setting of text knowledge acquisition, these different techniques may hardly be applied without any further modification and integration into the knowledge engineering environment. In particular, they may only bear their full benefits within a full-fledged integration with linguistics and statistics.

#### 3.3.2 Combination of Knowledge Structures with Linguistics and Statistics

There is a plentitude of possibilities for integrating shallow text processing and statistics into knowledge level approaches such that we refrain from a general account and give two examples from the literature that we plan to integrate and build on:

---

<sup>3</sup>Often the term “terminological logic” is used instead.

Assadi (1998) describes a clustering approach that combines linguistic and conceptual criteria. As an example he gives the pattern <NP, line> which results in two categorizations by modifiers. The first categorization is motivated by the `function_of_structure` modifiers, resulting in a clustering of `connection line`, `dispatching line` and `transport line` (cf. Table 1). For the other concepts the background knowledge lacks adequate specifications such that further categorizations could have been proposed.

A proposal categorization	The other candidate terms
connection line	mountain line
dispatching line	telecommunication line
transport line	input line

Table 1: Example categorization

Hahn & Schnattinger (1998) combine different linguistic and conceptual criteria that are also weighted according to their different strengths and frequencies. For example, the parallel relational structures

(5) IBM delivers DOS.

(6) IBM delivers OS/2.

indicate that OS/2 belongs to the same concept as DOS — both are operating systems. Two criteria of this kind are favored against one competing indication of the same type, but there are stronger indicators, such as the apposition in (7), which explicitly states that OS/2 is an `operating system`.

(7) OS/2, a modern operating system, is suited for ...

## 4 A Framework for Text Knowledge Acquisition

We have presented a number of different techniques that may be used for text knowledge acquisition. As became clear these techniques do not carry far, if applied in isolation, since they address only very specific, though overlapping, parts of the general text knowledge acquisition problem. Furthermore, the survey has shown that each of these different mechanisms may add further bits of knowledge to the repositories and each bit of knowledge may enhance other methods in carrying on with text knowledge acquisition. Hence, we aim at a flexible integration of all methodological dimensions just described. In order to bring about its full benefit, the integration model is applied incrementally and iteratively in a text knowledge acquisition cycle (cf. Fig. 3), which is embedded in the general MIKE cycle (cf. Fig. 1), such that the maximal amount of knowledge may be extracted out of given texts. Let us now elaborate on the text knowledge acquisition cycle (cf. Fig. 3).

**Text.** The subcycle of text knowledge acquisition starts with documents such as available from tutorial texts, from protocols of interviews, or from example texts of the domain to be covered in the knowledge-based system. The latter type of texts is especially interesting to semi-automatic text knowledge acquisition. With the spread of internet and intranets, the number of texts of this type tend to grow much faster than that of other types and, thus, should be exploited as far as possible. An additional problem derives from the fact that the knowledge contained in this type of texts is only implicitly available and, thus, explicit linkage between parts of the knowledge-based system specification and the texts, such as required by the MIKE methodology, may not be established easily. In contradistinction, semi-automatic text knowledge acquisition is amenable to recurring implicit information.

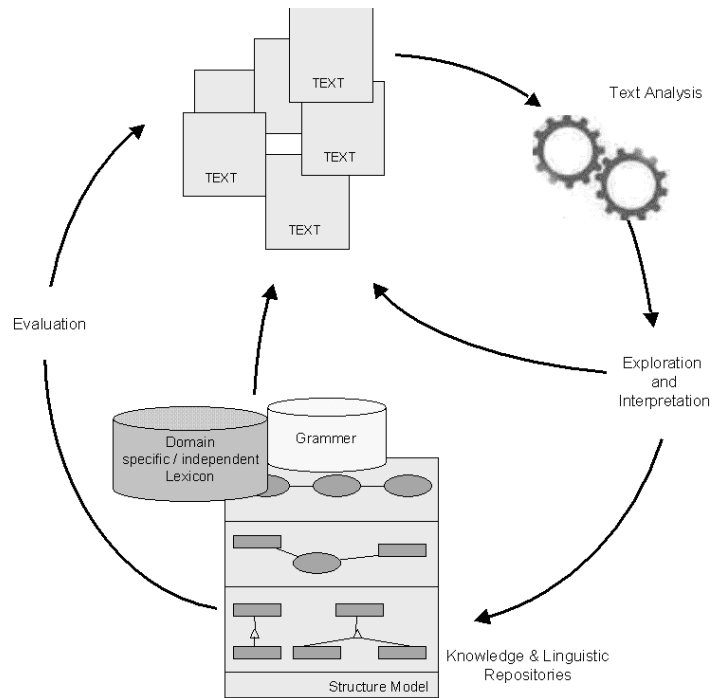


Figure 3: The text knowledge acquisition cycle

**Text Analysis.** In the second stage we provide methods and tools for manually or automatically structuring and analysing input texts. While we envision “a high ceiling” for the future, *i.e.* the integration of techniques from the most complex end of the text understanding spectrum, such as text structure analysis (cf. Mann & Thompson (1988)), we consider a “low point of entry” for our framework that is of immediate use to the knowledge engineer with the limited natural language processing capabilities available as of today. This low point of entry means simple counts of token or term frequencies, such as TF/IDF described in Section 3.2. It also includes manual structuring of the input texts as it has been introduced by common knowledge engineering methodologies. In the text analysis phase, we identify four “pairwise orthogonal” dimensions of information that may guide the acquisition of knowledge from texts:

1. Linguistic information, *e.g.* morphological analyses (cf. Section 3.1).
2. Statistical information, *e.g.* distribution of terms (cf. Section 3.2).
3. Guiding background knowledge, *e.g.* similarity of envisaged concepts (cf. Section 3.3).
4. Hand-crafted information, such as manually marked in the elicitation model (cf. Section 2).

A generalized framework must blend these different types of information into a common picture readily accessible to the knowledge engineer.

**Exploration and Interpretation.** This blending happens at the next level where the raw data resulting from analysis is re-organized to deliver a clearer picture. For this purpose, we introduce *equivalence classes*. Equivalence classes allow for the exploration of tokens, items and phrases in light of different linguistic, statistical, conceptual, or manually-shaped information contexts. The motivation underlying the notion of equivalence classes stems from the requirement of *combining* and *normalizing contexts* of terms or phrases at different levels of understanding. Let us give a brief elaboration of the three italicized notions we have used here.

*Context*, in our understanding, is a rather broad notion that includes not only the spatial collocation of terms, but also the linguistic context, such as given through dependency relations, the statistic context, which may be applied on top of any linguistic analysis, or the text structure context, such as elucidated manually, to name but a few. We show an example below, but we also think it is clear that all these types of contexts may help to clarify the semantics of a domain.<sup>4</sup>

*Normalization* is the process of linking together different representation structures with the same implicit content. On the morphology side this boils down to mapping tokens onto terms. On the grammatical part this may involve the mapping from syntactic cases with possibilities like active or passive voice onto voice-neutral case frames. On the logical side, this may happen by establishing a common normal form like Neo-Davidsonian event semantics (cf., e.g., (Pinkal, 1993) on event semantics).

Finally, a *combination*, i.e. a simultaneous application of analyses results towards guidance for text knowledge acquisition, is ultimately necessary, since no single dimension yields enough clues in order to give sufficiently sophisticated engineering support. For instance, neither grammatical information, such as a phrase forming the subject, nor statistics, e.g. terms a and b often cooccurring in sentences, nor background knowledge may be of much help if seen in isolation. However, in combination a result, which may be verbalized by (8), may strongly indicate a relationship between two concepts.<sup>5</sup>

- (8) Often, the tokens denoting **hotels** and **rooms** co-occur in the subject and direct-object slot, respectively, or in the direct-object and subject slot, respectively.

The central data structure that we use for extracting equivalence classes from text analysis results is an *extended text chart*. The extended text chart does not only contain the linguistic knowledge resulting from linguistic processing (cf. Section 3.1), it also incorporates manually-added information and, in particular, knowledge hypotheses. Thus, e.g., on the linguistic level a head-modifier relationship may be captured. The corresponding *text item* describes the text fragment concerned, the type of information being captured (e.g., head-modifier relationship or part-of-speech property) and, possibly, further specifying information. On the knowledge and manual level the same structure applies, but with different types of text items being added into the chart, examples for which are listed in Table 2.<sup>6</sup>

Level	Relevant Categories
Linguistics	Word stem, head-modifier relationships, co-reference, . . .
Knowledge	Conceptual denotation, conceptual relations
Manual	Topics, textual function, . . .

Table 2: Description levels in the extended text chart

The construction of such an extended text chart is not a new idea, but it has been applied in deep natural language processing for a long time. In contradistinction to these well-known approaches, we do not aim at acquiring knowledge by percolating pieces of information up from the lowest level of text understanding towards more knowledgeable levels. Rather, we pursue an explorative approach that allows the definition of equivalence classes by patterns that may bridge between the linguistics, knowledge and manual levels. With more background knowledge available the number of text items and, thus, their categorization into equivalence classes increases. If the knowledge level is only sparsely populated due to lacking conceptual descriptions, this does not pose a major problem to our approach, only the number of useful hints given to the knowledge engineer is reduced. With regard to matters of ambiguity, we must follow similar strategies as employed for shallow text processing — keep the text chart simple by applying heuristics or by

<sup>4</sup>Though this list is not exhaustive; one could, e.g. consider pragmatic contexts, which we abstract from.

<sup>5</sup>This criterion is similar to the criterion “cross-supported” proposed by (Hahn & Schnattinger, 1998).

<sup>6</sup>A concrete example is given in Section 5.

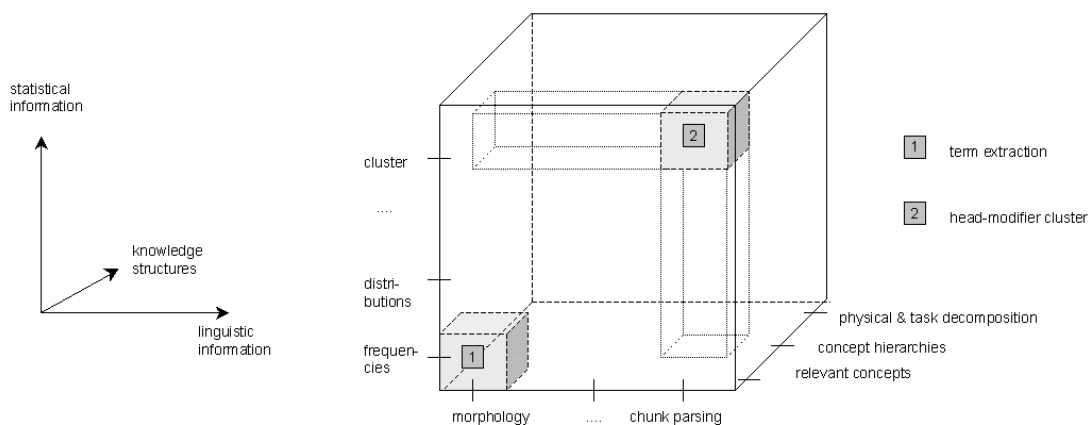


Figure 4: The orthogonal dimensions: linguistics, statistics, knowledge structures

refraining from the treatment of too difficult structures. Thus only, we may avoid the pitfalls that approaches for deep natural-language understanding fell prey to, *viz.* the explosion of the numbers of items in the text chart due to ambiguity and underspecification.

Given a definition of equivalence classes on the text chart, we may now exploit all kinds of statistical methods, too. Instead of simple statistics about term frequencies, etc., we may now also consider frequency or distribution counts on complex equivalence classes, such as the one for subject-object dependencies on certain pairs of concepts (cf. the example in Section 5.). Of course, the question remains of which combinations should be used for implementation in the actual system. Building on state-of-the-art tools that deliver information in the four different dimensions (cf. Figure 4, which depicts the three major dimensions), well-working approaches like (Biébow & Szulman, 1999; Assadi, 1998; Hahn & Schnattinger, 1998; Barker et al., 1998; Lapalut, 1996) can be taken of the shelf and reformulated according to our framework as a first starting point, while the approach remains very extensible towards new approaches being explored in one of the different communities.

Designing and testing presentation strategies, one encounters a natural trade-off between the complexity of indications and the gain derived from a certain number of examples. Low level information, like term frequencies, often needs much more actual support from the texts in order to indicate the same degree of relevance as complex information, providing a much more concise picture of some text content.

**Knowledge & Linguistic Repositories.** Depending on the methodology of equivalence classes, the knowledge engineer may navigate the contextual information sources and consider statistical information in order to make modeling decisions. In addition, equivalence classes may be used to propose reasonable concept definitions or at least to restrict the place in the ontology where a concept should be modeled down to a small region that is easy to oversee by the knowledge engineer.

Thus, knowledge repositories are filled up in the course of semi-automatic acquisition of knowledge from text input. However, many decisions that are to be made are not restricted to the knowledge level, but may also affect the linguistic repositories and may enhance text analysis in the further course of text knowledge acquisition. While we assume that core, domain-independent linguistic repositories for lexical and grammar descriptions are specified explicitly (*e.g.*, we build on a system from a former project), domain-specific linguistic descriptions may also be amended in the knowledge acquisition process. A prime example for this linguistic-conceptual repository extension is found to be the domain lexicon that contains the domain-specific descriptions of lexical entries. It may be extended when the knowledge engineer decides about mappings from words to

concepts, *e.g.*, formerly unknown words may be associated with linguistic word class information or with links to their conceptual denotations.

In addition, the ultimate goal may be the construction of a knowledge-based system with natural language capabilities. Then, the inclusion of domain-specific linguistic information into the text knowledge acquisition cycle does not only improve knowledge acquisition, but also contributes to the linguistic capabilities of the system to be constructed. Hence, our approach provides a systematic way of embedding generic natural language processing tools into applications from a particular domain, their linguistic repositories being specialized with regard to domain-specific linguistics and ontology (cf. Section 6).

**Evaluation.** The evaluation of the text knowledge acquisition cycle we propose is still an open issue. It will have to deal with comparisons of different text knowledge acquisition heuristics on the same data — which up till now is an almost untackled problem — and which will only become possible by a framework that allows for the formulation of different heuristics working on the same background knowledge and text input. However, even then the methodology for the evaluation will be a crucial point of further research.

## 5 An Example Text Knowledge Acquisition Process

Let us now consider a scenario in which we assume that we are in the midst of a text knowledge acquisition process. We are given a text corpus about a tourism domain describing actual objects, such as hotels in a city, administrative information, cultural events, etc. The purpose is to extend an ontology from the example descriptions given in the text corpus.

**Text.** As relevant corpus we used a WWW provider (located at <http://www.all-in-all.com>) for tourist information of Mecklenburg-Vorpommern<sup>7</sup>. The corpus contains 542 HTML documents about regional tourist information. For our example acquisition steps we here consider the simplified fragment “Hotel ‘Five Seasons’: The hotel offers 45 luxury rooms. It also has ...” (cf. Fig. 5). The only knowledge added manually is that the example text contains information about hotels (cf. the manual level in Fig. 5).

**Text Analysis.** At the linguistic level we implemented a first version of a shallow text processor, such as described in Section 3.1. Currently we focus on the German language, but we have started to consider English as well, in order to support multi-lingual shallow text processing. The current system (called STP) has a very broad linguistic knowledge source, *i.e.* a huge lexical data base (more than 120.000 stem entries, more than 12,000 subcategorization frames as well as basic lexica for proper names). It has a broad coverage of special subgrammers for recognizing name entities (including proper names, company names, complex time/date expression) and general grammars for nominal, prepositional and verb phrases. Complete processing of a text with 400 words takes about 1 second. STP has a high degree of modularity: each component can be used in isolation. Thus, it is possible to run the STP only on a subset of the components.

Tokenization, lexical analysis and chunk parsing of our example yield syntactic results such as “the hotel” and “45 luxury rooms” being nominal phrases.<sup>8</sup> For our example we depict the grammatical roles of “the hotel” and “45 luxury rooms” (subject and direct object, respectively) that are returned by the chunk parser (cf. Fig. 5), too. The text parser must establish the coreference between “hotel” and “it”.

At the knowledge level we start now with an ontology consisting of 235 concepts, which are only weakly related via 48 slot descriptions. The concepts `hotel` and `room` are already known and recognized from the corresponding text fragments by the linguistic analysis. However, a detailed concept description linking these two concepts together is still missing in the ontology.

<sup>7</sup>Mecklenburg-Vorpommern is a region in the north east of Germany.

<sup>8</sup>In English the compound analysis of “luxury rooms” is delegated to the chunk parser, while in German the corresponding task for “Luxuszimmer” is handled by the lexical analysis.

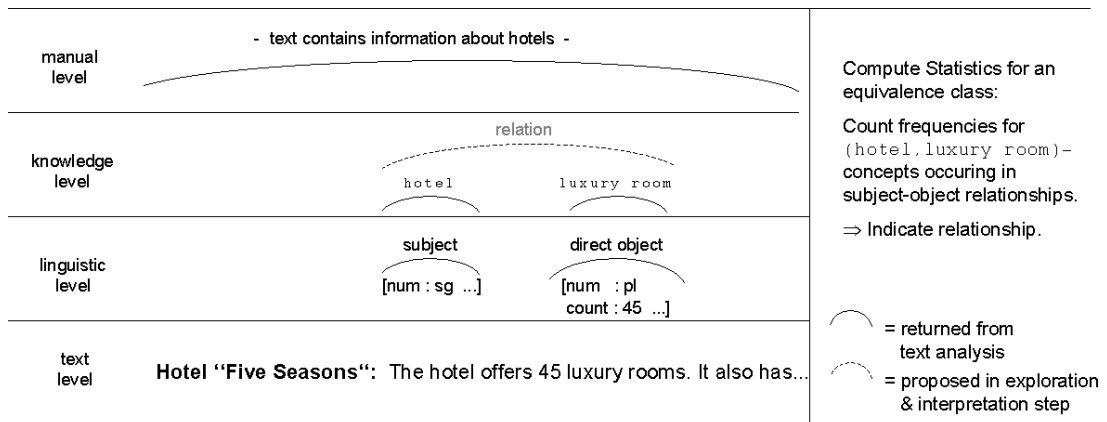


Figure 5: An example for a step in the text knowledge acquisition process

**Exploration and Interpretation** Based on the results of text analysis we may now explore and interpret regularities, associations, clusters, etc. by matching our equivalence class definitions on the data available on the extended text chart (cf. Section 4). An example definition for acquiring relationships between concepts is given as follows

- (9) search all pairs (concept<sub>1</sub>, concept<sub>2</sub>) such that the paraphrases for these concepts occur in subject–direct object relationships

We may also consider statistics on the equivalence class just defined. In our small example (hotel, luxury room) turn out to be concepts that should be related.

**Filling the Repositories.** With the results of the exploration and interpretation step the knowledge engineer may now decide about whether and how to introduce the relationship. Eventually, he may decide to establish a *has-part* relation between hotel and luxury–room.

## 6 The Application Cycle

As depicted in Figure 6, the *application cycle* reverses the direction of processing found in the knowledge acquisition cycle yielding a generic approach for a system with natural language capabilities. The acquired knowledge & linguistic repositories may be exploited to improve the functionality of a knowledge-based system with NL capabilities. The acquired repositories (ontology, domain lexicon and, possibly grammar) that are integrated in the application cycle of the domain-specific NL system provide the linguistic & conceptual background for partial text understanding. Text understanding may be evaluated by the inclusion of evaluation strategies, which might combine the evaluation techniques in the area of information retrieval (cf. (TREC, 1998)) and in the area of information extraction (cf. (MUC7, 1998)).

We have in mind a particular application scenario: GETESS<sup>9</sup> (Staab et al., 1999) is an intelligent information agent that uses semantic methods and natural language processing in order to gather tourist information from the WWW and present it to the human user in an intuitive, user-friendly way. Natural language processing is applied for linguistically analyzing user queries and creating a summary of facts from NL documents. A fail-soft approach is pursued that is based on extracting knowledge from text with a robust parser, but that also integrates and falls back onto common information retrieval mechanism once the linguistic mechanisms fail to deliver due to the complexity of language or due to underspecification.

<sup>9</sup>German Text Exploitation and Search System

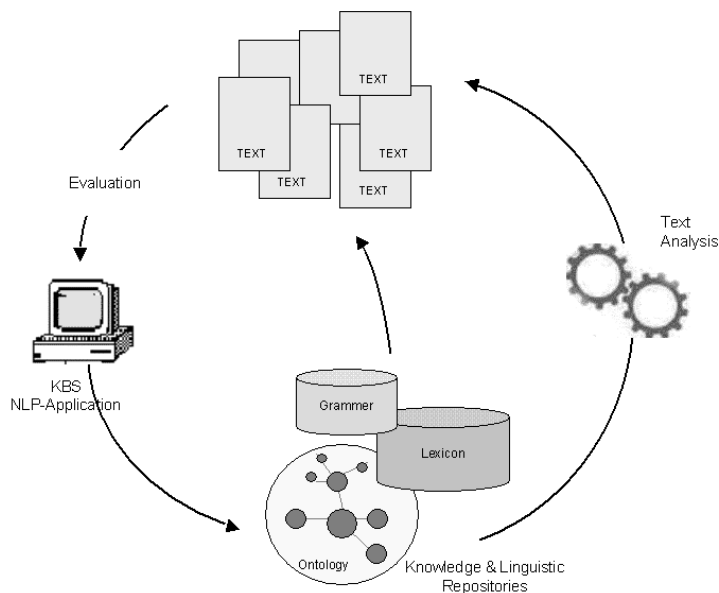


Figure 6: The NLP application cycle

Since GETESS is partially based on semantic methods, it is necessary to use knowledge acquisition tools to build up a domain lexicon and an ontology. The ontology is used as a semantic reference model for the intelligent information agent: it mediates between the different modules and it supports the different retrieval tasks. In particular, it offers inference facilities that are exploited by the other modules. Furthermore, the ontology allows the exploration of the conceptual structures via an innovative user interface (cf. Lamping & Rao (1996)), thus providing concise, intuitive access to facts extracted from input texts.

## 7 Conclusion

We have presented an architectural framework for integrating text knowledge acquisition techniques into the knowledge-based system construction of the MIKE process. With this framework we bring together formerly isolated approaches and provide a cyclic approach that allows for incremental extensions of conceptual models and linguistic background knowledge, thus serving not only the needs of the knowledge engineer, but also the acquisition of natural language capabilities.

Thereby, our approach as applied in Sections 5 and 6 reflects our understanding of text mining in general and text knowledge acquisition in particular: We believe that text mining is not a fully automatic process. Rather, we conceive of a change of perspective that may be compared to the one that has taken place in data mining applications. There, the original idea was the exclusive usage of machine intelligence for the detection of relevant interdependencies. However, while data mining has matured as a discipline, the widespread conviction that has evolved, now considers data mining as being a two-stage process. The process is set up by a programmer with a particular application in mind, before the actual user may analyze his data and discover the interesting pieces of knowledge. Similarly, we here conceive of semi-automatic acquisition of knowledge which is guided by the knowledge engineer and which results in a domain-specific application — that may be a domain-specific information extraction system.

Our next steps will be the incorporation of parsing and text knowledge acquisition technologies as described in (Neumann et al., 1997; Hahn & Schnattinger, 1997; 1998; Biébow & Szulman, 1999; Barker et al., 1998; Assadi, 1998; Lapalut, 1996; Szpakowicz, 1990) into the MikeTool (Steinberg, 1999). Thus, we hope to facilitate the further extension and maintenance of our application, GETESS (Staab et al., 1999).

## References

- Abney, S. (1996). Partial parsing via finite-state cascades. In *Proceedings of the European Summer School in Logic, Language and Information (ESSLLI'96) – Robust Parsing Workshop*.
- Angele, J., Fensel, D., Landes, D., & Studer, R. (1998). Developing knowledge-based systems with MIKE. *Journal of Automated Software Engineering*, 5(4):389–418.
- Appelt, D., Hobbs, J., Bear, J., Israel, D., & Tyson, M. (1993). FASTUS: A finite state processor for information extraction from real world text. In *IJCAI-93: Proceedings of the 13<sup>th</sup> International Joint Conference on Artificial Intelligence. Chambery, France, August 28 - September 3, 1993*, Chambery, France.
- Assadi, H. (1998). Construction of a regional ontology from text and its use within a documentary system. In *Proceedings of the International Conference on Formal Ontology and Information Systems - FOIS'98*, Trento, Italy.
- Baldwin, B., Morton, T., Bagga, A., Baldrige, J., Chandraseker, R., Dimitriadis, A., Snyder, K., & Wolska, M. (1998). Description of the UPENN CAMP system as used for coreference. In (MUC7, 1998).
- Barker, K., Delisle, S., & Szpakowicz, S. (1998). Test-driving TANKA: Evaluating a semi-automatic system of text analysis for knowledge acquisition. In Mercer, R. & Neufeld, E. (Eds.), *Advances in Artificial Intelligence. Proceedings of the 12<sup>th</sup> Biennial Conference of the Canadian Society for Computational Studies of Intelligence (AI '98). Vancouver, Canada, June 18-20, 1998*, LNAI 1418, Berlin. Springer.
- Biébow, B. & Szulman, S. (1999). TERMINAE: A linguistics-based tool for the building of a domain ontology. In *EKAW '99 - Proceedings of the 11th European Workshop on Knowledge Acquisition, Modeling, and Management. Dagstuhl, Germany*, LNCS, pages 49–66, Berlin. Springer.
- Brill, E. (1993). Automatic grammar induction and parsing free text: A transformation-based approach. In *ACL'93 — Proceedings of the Annual Meeting of the Association for Computational Linguistics*, Ohio.
- Buitelaar, P. (1998). *CORELEX: Systematic Polysemy and Underspecification*. PhD thesis, Brandeis University, Department of Computer Science.
- Charniak, E. (1993). *Statistical Language Learning*. MIT - Press, Cambridge, MA.
- Cheeseman, P. & Stutz, J. (1995). Bayesian classification (AutoClass): Theory and results. In Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P., & Uthurusamy, R. (Eds.), *Advances in Knowledge Discovery and Data Mining*. The AAAI Press, Menlo Park.
- Chinchor, N., Hirschman, L., & Lewis, D. (1993). Evaluating message understanding systems: An analysis of the third message understanding conference (MUC-3). *Computational Linguistics*, 19(3).
- Eriksson, H. (1992). A survey of knowledge acquisition techniques and tools and their relationship to software engineering. *Journal of Systems and Software*, 19:97–107.
- Fensel, D., Angele, J., & Studer, R. (1998). The knowledge acquisition and representation language KARL. *IEEE Transactions on Knowledge and Data Engineering*, 10(4):527–550.
- Grishman, R. & Sundheim, B. (1996). Message Understanding Conference – 6: A Brief History. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING'96)*, pages 466–471, Copenhagen, Denmark, Europe.

- Hahn, U. & Schnattinger, K. (1997). An empirical evaluation of a system for text knowledge acquisition. In *EKAW'97 - Advances in Knowledge Acquisition. Proceedings of the 10th European Knowledge Acquisition Workshop*, LNCS, pages 129–144, Berlin. Springer.
- Hahn, U. & Schnattinger, K. (1998). Towards text knowledge engineering. In *AAAI '98 - Proceedings of the 15th National Conference on Artificial Intelligence. Madison, Wisconsin, July 26-30, 1998*, pages 129–144, Cambridge/Menlo Park. MIT Press/AAAI Press.
- Hobbs, J. (1992). The generic information extraction system. In (MUC4, 1992).
- Lamping, J. & Rao, R. (1996). The hyperbolic browser: A focus + context technique for visualizing large hierarchies. *Journal of Visual Languages & Computing*, 7.
- Lapalut, S. (1996). Text clustering to help knowledge acquisition from documents. In *EKAW '96 — Proceedings of the 9th European Workshop on Knowledge Acquisition. Nottingham, United Kingdom*, LNCS, pages 115–130, Berlin. Springer.
- Mann, W. & Thompson, S. (1988). Rhetorical structure theory: Toward a functional theory of text organization. *Text*, 3(8):243–281.
- Michalski, R. S., Carbonell, J. G., & Mitchell, T. M. (1986). *Machine Learning, an Artificial Intelligence approach*. Morgan Kaufmann, San Mateo, California.
- MUC4 (1992). *MUC-4 — Proceedings of the 4th Message Understanding Conference*.
- MUC7 (1998). *MUC-7 — Proceedings of the 7th Message Understanding Conference*. <http://www.muc.saic.com/>.
- Neumann, G., Backofen, R., Baur, J., Becker, M., & Braun, C. (1997). An information extraction core system for real world german text processing. In *ANLP'97 — Proceedings of the Conference on Applied Natural Language Processing*, pages 208–215, Washington, USA.
- Pinkal, M. (1993). Semantik. In Görz, G. (Ed.), *Einführung in die künstliche Intelligenz*, pages 425–498. Addison-Wesley, Bonn.
- Salton, G. (1988). *Automatic Text Processing*. Addison-Wesley.
- Schreiber, A. T., Wielinga, B. J., Akkermans, J. M., de Velde, W. V., & de Hoog, R. (1994). CommonKADS: A comprehensive methodology for KBS development. *IEEE Expert*, 9(6):28–37.
- Smadja, F. (1993). Retrieving collocations from text: Xtract. *Computational Linguistics*, 19(3).
- Sparck-Jones, K. & Willett, P. (Eds.) (1997). *Readings in Information Retrieval*. Morgan Kaufmann.
- Staab, S., Braun, C., Düsterhöft, A., Heuer, A., Klettke, M., Melzig, S., Neumann, G., Prager, B., Pretzel, J., Schnurr, H.-P., Studer, R., Uszkoreit, H., & Wrenger, B. (1999). GETESS — searching the web exploiting german texts. In *CIA'99 — Proceedings of the 3rd Workshop on Cooperative Information Agents*, LNCS, page (to appear), Berlin. Springer.
- Steinberg, U. (1999). MIKE tool II: Ein CAKE Werkzeug. Master's thesis, Karlsruhe University. (In German).
- Szpakowicz, S. (1990). Semi-automatic acquisition of conceptual structure from technical texts. *International Journal of Man-Machine Studies*, 33.
- TREC (1998). *TREC — Proceedings of the 7th Text REtrieval Conference*. <http://trec.nist.gov/>.
- Veloso, M. (1994). *Planning and Learning by Analogical Reasoning*. Springer, Berlin.
- Woods, W. A. & Schmolze, J. G. (1992). The KL-ONE family. *Computers & Mathematics with Applications*, 23(2-5):133–177.