GETESS: Constructing a Linguistic Search Index for an Internet Search Engine

Markus Becker[1], Jochen Bedersdorfer[1], Ilvio Bruder[2], Antje Düsterhöft[2], and Günter Neumann[1]

[1] DFKI GmbH
Stuhlsatzenhausweg 3
66123 Saarbrücken, Germany
{neumann,beders,mbecker}@dfki.de
[2] University of Rostock, Computer Science Department
A.-Einstein-Str. 21
18051 Rostock, Germany
{duest,ilr}@informatik.uni-rostock.de

Abstract. In this paper we illustrate how Internet documents can be automatically analyzed in order to capture the content of a document in a more detailed way than usually. The result of the document analysis is called **abstract** and will be used as a linguistic search index for the Internet search engine GETESS.
We show how the linguistic analysis system SMES can be used for a Harvest based search engine for constructing a linguistic search index. Further, we denote how the linguistic index can be exploited for answering user search inqueries.

1 Introduction

Growing amounts of information in cyberspace make it increasingly difficult for network users to locate specific information for certain themes. Even experts sometimes experience the "joy" of becoming "Lost in Hyberspace".

In contrast, a wide variety of tools and services exist that are useful for information searches in the Internet but whose efficiency is somewhat limited. The services tend to supply unsatisfactory search results, which are either too extensive, inapplicable or incomplete. The majority of tools used for information searches in the Internet concentrate primarily on so-called syntactical attributes, such as TITLE or DESCRIPTION, without considering the actual meaning of the information.

The bulk of available information in the Internet is provided in natural- language format and supplemented with graphics. Furthermore, user queries are typically formulated using natural-language words and phrases. However, none of the well-known search tools take advantage of the use of a natural language combined with graphics, pictures, icons and menus. This, despite the fact, that

the computer linguistic field during the past few years has developed a wide variety of tools and mechanisms for partially automatic, natural-language processing which could be employed as intelligent search support.

Internationally, the English language has established itself in the Internet. The majority of information in the Internet in Germany is presented in both German and English. Even in German-speaking areas, the German language plays an increasingly subordinate role in the Internet. Currently, a typical Internet user is generally either an information expert, student or computer freak with some command of the English language. However, as access to the Internet increases, the circle of users is also expected to become more multifaceted. At some point, knowledge of the English language can no longer be assumed .

The project GETESS (http://www.getess.de) [1] [10], [11] focus on the development of a Internet search engine which analyses information in German of a defined domain.
Therefore, the starting point is the idea to combine Internet techniques with database and knowledge representation methods as well as results from the computer linguists. The GETESS architecture integrates the different aspects in order to give more detailed information to the user.
The user can formulate his/her query using natural language phrases and will be supported by an ontology. The result of the query includes so-called *'abstracts'* which represents the content of the web documents as a kind of a summary.

Starting from the GETESS main ideas (*section 3*) we illustrate the functionality of the GETESS Gatherer and the architecture of the information extraction system SMES (*section 4*). In *section 5* we discuss the conceptual idea of the SMES integration into the GETESS-Gatherer and the actual implementation. In *section 6* we give a conclusion and consider future work.

2 The motivation - Issues to be solved

Internet search engines are actually able to answer inqueries (cf. [12], [13], [14]) like

- "hotel" or "Rostock" or "coast" - simple keywords or
- "hotel and Rostock" - simple keywords connected via boolean operators or
- "hotel between Rostock and Wismar" - strings.

[1] GETESS is funded by the German Bundesministerium für Bildung, Wissenschaft, Forschung und Technologie (BMBF) under grant number 01IN802. The partners of the project are AIFB, University of Karlsruhe, DFKI Saarbrücken, Gecko mbH Rostock and University of Rostock.

It's a fact, that these possibilities of describing the user problems are not fitting the user needs. The user wants to have an easy to formulate (natural) query language. He/She wants to describe questions[2] like

- "ich suche eine Unterkunft an der Küste zwischen Rostock und Wismar"
  ("I'm looking for an accomodation on the coast between Rostock and Wismar")
  The user wants to have with this inquery
  - an accomodation that can be a hotel or an appartment or a youth hotel or ...
  - Rostock and Wismar are cities; the user wants to have an accomodation *between* these cities
  - the user wants to have an accomodation on the coast; in combination with the geographical information that Rostock and Wismar are cities on the Baltic Sea one can sure that the user wants to be on the coast of the Baltic Sea

  This information behind the user inquery has to be extracted. The user is not really able to give that information explicitly. Thats why a search engine has to support the user in analyzing the user inquery.

- "ich möchte ein Doppelzimmer unter 150 DM"
  ("I want to have a double room which costs not more than 150 DM")
  A search engine that wants to answer this query has to analyze all the WWW documents in that way that prices must be read as integer values. In practice, most prices of Web information are integrated into text as strings. A deeper analyzes of the text is necessary to classify the right strings in the texts as prices. The same problem we have for addresses of instuitutions (e.g. hotels), special multi-valued sets (e.g. the set of credit cards), times (e.g. opening times), etc.
  The problem can be solved when a Web information of an domain (e.g. tourism) will be analyzed and classified in the context of that domain. Important information can be extracted and stored in the right way (e.g. for comparisons).

- "ich suche ein Doppelzimmer mit Swimmingpool"
  ("I'm looking for a double room with a swimmingpool ")
  The user give this inquery and wants to have a double room in a hotel (or something else) where the hotel has a swimmingpool. The problem in answering this inquery is that the information about rooms and about general equipment of an accomodation are mostly stored over different Web pages. A keyword search never would find that complex information because the relevant keywords are not in the same Web page.
  Thats why these information of one and the same institution must be connected - different Web pages have to be logical put together for answering such questions.

[2] These inqueries are real-life ones which we acquired at the Cebit fair in 1998

In the GETESS project we use a domain-related approach in which all specific appearances of things are seen as representation of that thing. That means, specific words which are important in our domain are mapped onto concepts; e.g. 'Master card' is a specific appearance of 'credit card'. Finally, the search engine uses these concepts in order to give search results. The search engine also maps the natural language user inqueries onto concepts and tries to match the concepts of the analyzed Web documents.

3 GETESS - GErman Text Exploitation and Search System

The GETESS project focuses on developing an intelligent Internet search site. It will offer the user an easy-to- operate system and will provide for description of search requirements in a natural language. Intelligent systems situated between search sites and information will be responsible for condensing extensive and complex data into language-independent, content-weighted summaries (*abstracts*). The sum total of *abstracts* will provide a base and be subsequently used as results for queries. Results will then be translated into the user's mother tongue.
A fundamental requirement for development of an Internet search site is an extremely large amount of data. Accordingly, the amount of information allocated to the *abstracts* will also be substantial. Therefore, databases will be used for efficient storage and quick access to *abstracts*.

The core of the system features a search site which (like other search services) will provide availability to conventional information (e.g. HTML-Documents, etc.) However, it will also have the capability to access *abstracts* in order to supply satisfactory responses to search queries. Therefore, the GETESS search engine must be capable of accessing database information. A response to a query will consist of a presentation of one or more results, as well as a ranking of the information to provide a plausible sequence for the user.

Linquistic and domain-specific information (e.g. in an ontology) will be employed when initiating a search query and for a response. Accordingly, one objective will be to guide users in preparing search descriptions. This means preparing search requests that specifiy parameters with regard to scope and quality of information. By offering content-dependent generic terms or concepts, it will be possible to limit the search domain. Additionally, a search process is better specified when relevant generic terms are negotiated. This is particularly important in situations where users have difficulty describing desired information or if they are unable to determine relevant search terms.

Through a specific linguistic and ontology-based analysis using information-extraction methods, structural links will be determined in natural- language

queries (e.g. modifications). This will make it possible for a search to consider relational circumstances between individual words in a natural-language query.

Language restrictions. The GETESS-System commences with the German language and notably offers users German-language WWW documents as search results. Furthermore, when users formulate natural-language queries, they are supported in the German language. [3] However, language autonomy will continue to be observed as much as possible during the development of the GETESS search site.
An additional step will consider making *abstracts* available for an English translation, through an existing WWW automatic- translating tool. Furthermore, a study of the japanese language is planned to examine the use of the GETESS search site for Japanese documents.

Domain restrictions. An indepth, automatic natural-language analysis and classification of WWW documents will initially require a defined domain. Tourism was chosen for the GETESS search site which will subsequently enlist tourism documents from Mecklenburg-Western Pomerania as data. Tourism data from other German states will also be analyzed in the next steps.
In order to investigate the adaptability and expense, the GETESS system will be applied to a second domain in a further step.

3.1 GETESS Architecture

The front end of the GETESS system (cf. a depictation of its architecture in Figure 1) provides a user interface that is embedded in a dialogue system controlling the history of interactions. Single interactions are handed to the query processor that selects the corresponding analysis method, viz. the natural language processing module or the information retrieval and database query mechanisms. While the latter ones can be directly used as input to the search system, the natural language processing module first translates the natural language query into a corresponding database query, before ist sends this formal query to the search system.

In order to process queries and search results, three kinds of resources are provided by the back end of the GETESS system. First, archieved information is available in several content databases (the *abstract* DB, the index DB and the DB repository), the function of which is explained below. Second, the lexicon and the ontology provide metaknowledge about the queries, viz. about the grammatical status of words and their conceptual denotations. Third, a database incorporating dialogue sequences and user profiles, gives control over dialogue

[3] In this case, natural language means a limitation to natural-language phrases, because experience indicates WWW users rarely use full or complete sentences.

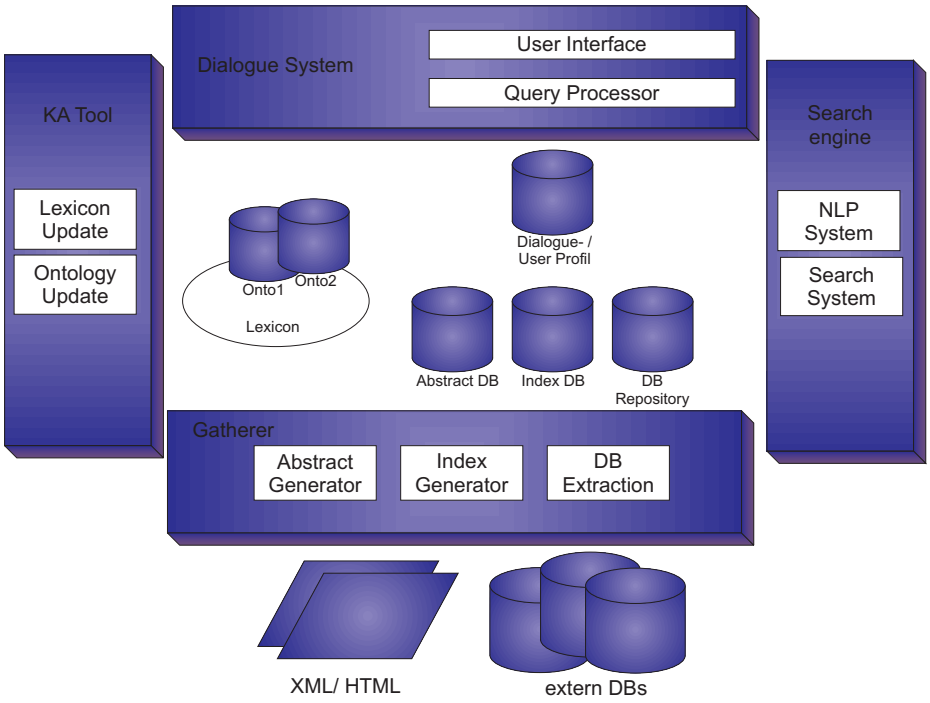

Fig. 1. GETESS Architecture

interactions.

While dialogue sequences and user profiles are acquired the course of interactions and the metaknowledge is provided by the human modeller with the help of knowledge acquisition tools (KA tools), the content databases must be filled automatically, since the content of typical web sites change almost on a daily basis. For this task the *gatherer* searches regularly through relevant XML/ HTML-pages and specified databases in order to generate corresponding entries in the *abstract* database, the index database and the database repository.

The content in the *abstract* database is derived from a robust, through uncomplete natural language understanding module that parses documents and extracts semantic information building an *abstract* for a document or a set of documents. These *abstracts* are sets of facts, i.e. tupels, like hasChurch(Rostock, Church-1), that could be extracted from natural language text, like "Rostock's major church was build during mediaval times." The index generator builds access information for full text search with information retrieval methods, while

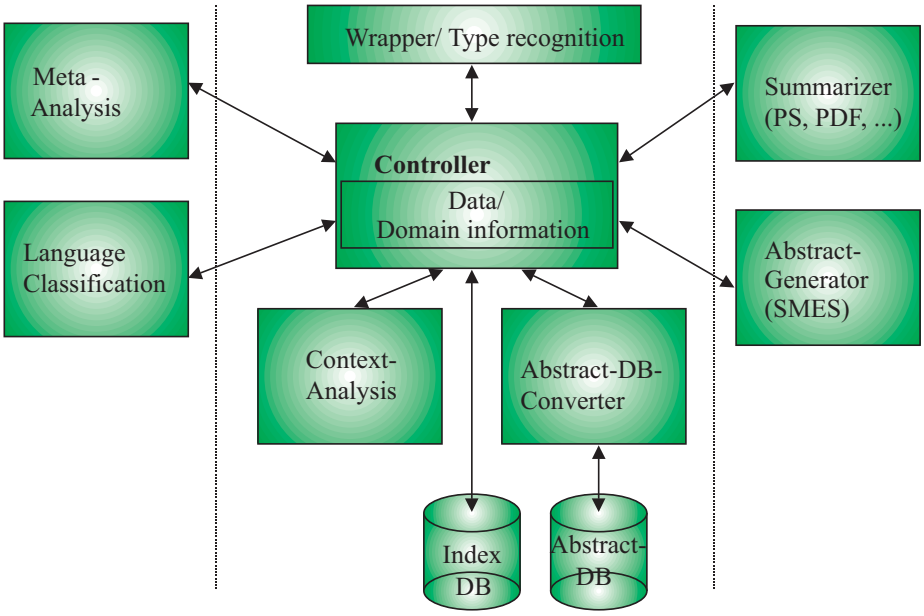

Fig. 2. GETESS Gatherer

DB repository offers relevant views onto extern databases.

The primary focus of the GETESS search site is the Internet search engine that makes it possible to store information in a database and then makes it quickly available for user inquiries. Above all, the storage of information in a database assumes the development of database solutions for the relevant data - in this case *abstracts*, a knowledge base and dictionaries. Furthermore, a search engine has to be developed with the capacity to initiate preparation of *abstracts* from Internet information and with the ability to access *abstracts* and relevant information from a database. The search engine also has the capacity for allocating user requests to database-stored *abstracts*.

3.2 The GETESS Gatherer

The GETESS-Gatherer (cf. Figure 2) is one of the main components of the GETESS architecture. The Gatherer works as an Internet agent that periodically collects data from the Internet. After a data type recognition the Gatherer-controller starts the summarizers and and the SMES-*abstract*-generator in order to analyze the documents (**information extraction**). The different summarizers parse e.g. HTML, XML, PS, RTF,... documents and extracting keywords

using information retrieval techniques (**keyword extraction**). The *abstract*-generator creates *abstracts* and uses the meta-analysis, where especially HTML-Meta-Tags and the HTML structure are analyzed, as well as the language classification and the information extraction where knowledge about the structure of an complex Internet site will be extracted.
The task of the controller is to coordinate the work of the agent together with the other Gatherer-tools.

Harvest - The Basis of the GETESS-Gatherer. The Basis of the GETESS-Gatherer is the Harvest gatherer (cf. [6]). Harvest is actual one of the most important search engines. "Currently, there are hundreds of Harvest applications on the Web (for example, the CIA, the NASA, the US National Academy of Science, and the US Government Printing Office), as this software is on the public domain." [2]
The GETESS internet agent, the keyword extraction mechanism and the summarizers are mainly based on the Harvest system.
In parallel to the keyword extraction the GETESS gatherer will initiate the information extraction (SMES). So, we have in every case results; on the basis of the lingustic analysis in form of abstracts or on the basis of the keyword analysis. In reason of that strategy on the one hand we have to integrate the information extraction system SMES into the internet agent. Otherwise, we have to make available all additional information which the specific GETESS tools are collected like e.g. Meta-information to the harvest components.

Meta-analysis. Within the meta-analysis typical structures of documents as well as significant structures of an Internet domain will be automatically analyzed. Therefore, an internet agent is working BEFORE the information extraction in order to find collections of similar documents and special URL-strings.

Document collections are created when documents have nearly the same HTML-structure. An approximation algorithm [4] is used to decide whether a document is similar to another.

The URL-analysis is employed to find hints of the Internet domain structure. So, e.g. strings like 'http://www...../english/...' or 'http://www...../en/...' are typical to find english documents. Furthermore, all directories (which have a file 'index.html') are stored in order to classify the according directory files as putting them logical together.

The meta-analysis agent collects these heuristic information and makes them available for the keyword and information extraction.

Language Classification. The language classification is used to define the document's language [5]. The linguistic analyses SMES combined with an Internet agent will get to know this document language when an *abstract* of a

document will be constructed.
Otherwise, documents with the same content but of different languages should be put together via the constructed *abstracts* in the database. So, equal *abstracts* representing documents with the same content but of different language. Lastly, the GETESS search engine takes only one of these *abstracts* as a query result - exactly the *abstract* of this document which is of the language the user want to have.

Abstract-Database Converter. Lastly, the GETESS-Gatherer has to store the constructed abstracts into a database. Because the abstracts are of XML format a special convertation into database structures is needed. A discussion about possible database systems and mechanisms for storing the abstracts can be found in [9], [15].

The GETESS-Gatherer runs periodically and collects data in the Internet. The result is a linguistical index which has the form of abstracts and is stored into a database. Further, a keyword index is constructed for the case that the linguistic analysis. Both indexes are used in the natural language dialogue system in order to answer user inqueries.
In the next sections we focus on the specific problem of integrating the information extraction system SMES into the Harvest Gatherer agent.

4 Information extraction with SMES

Linguistically based pre-processing of text documents is performed by SMES, an information extraction core system for real world German text processing (cf. [7], [8]). 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 essential abstract data types used in SMES are:

- dynamic tries for lexical processing:
  tries are used as the sole storage device for all sorts of lexical information (e.g., for stems, prefix, inflectional endings). Beside the usual functionality (insertion, retrieval, deletion), a number of more complex functions are available, most notably a regular trie matcher, and a robust recursive trie traversal which supports recognization of (longest matching) substrings. The latter is the basic algorithm for on-line decomposition of German compounds.
- weighted finite state transducers (WFST):
  WFST are used for representing cascades of grammar modules, e.g., proper name grammars (e.g. organizations, complex time/date expressions, person names), generic phrasal grammars (nominal and prepositional phrases, and

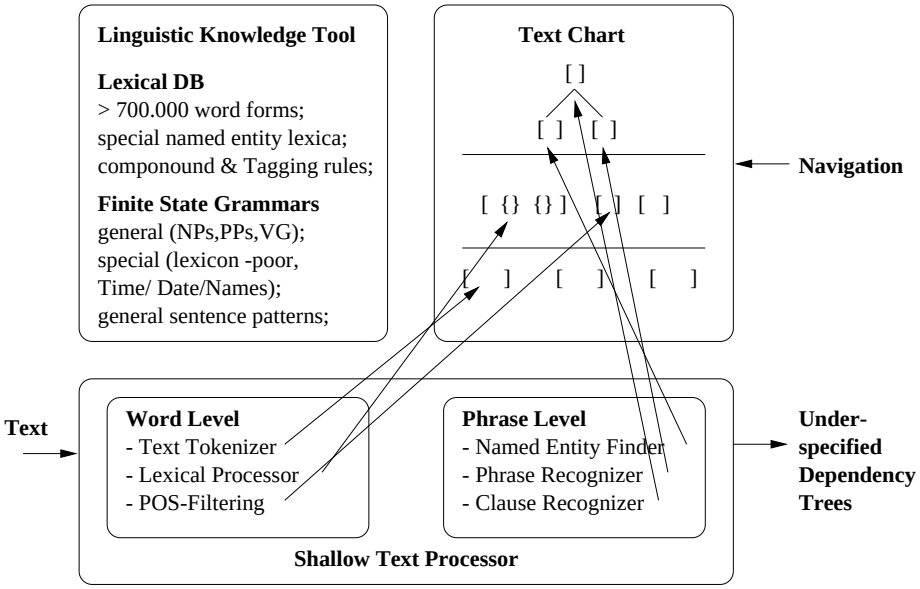

Fig. 3. The blue print of the SMES system architecture

verb groups) and clause level grammars. Using WFST supports efficient and robust representation of each individual grammar modul.

4.1 The SMES Architecture

The overall architecture of the SMES system (shown in figure 3) consists of two major components, a) the Linguistic Knowledge Pool (LKP) and b) STP, the core shallow text processor of SMES. STP consists of three major components: the tokenizer, the lexical processor and the chunk parser.

Text tokenizer. Each file is firstly preprocessed by the *text scanner*. Applying regular expressions, 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))).

Lexical processor. Each token, which is identified as a potential wordform is lexically processed by the LEXICAL PROCESSOR. Lexical processing includes

morphological analysis, recognition of compounds, retrieval of lexical information (where retrieval allows fuzzy matching), and tagging, which performs word-based disambiguation. Morphological analysis 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 output after morpholigy is the word form together with all its readings. A reading is a triple of the form tuple{stem,inflection,pos}, 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, the morpholgical analyzer 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).

Chunk parser. The chunk parser is subdivided into three components.In a first step phrasal fragments are recognized, like general nominal expressions and verb groups or specialized expressions for time, date, and named entity. The structure of potential phrasal fragments is defined using weighted finite state transducers WFST. In the second step, the dependency-based structure of the fragments of each sentence is analysed using a set of specific sentence patterns. These patterns are also expresssed by means of finite state transducers FST, so that step one and three are uniformly realized by the same underlying mechanism. In the final fourth step, the grammatical functions are determined for each dependency-based structure on the basis of a large subcategorization lexicon.

SMES has 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 broad coverage special subgrammars for recognizing unknow proper name on the fly and general grammars for nominal phrases, prepositional phrases and verb groups. Complete processing of a text with 400 words takes about 1 second. Very importantly in the context of this paper, is SMES's high degree of modularity: each component can be used in isolation. Thus it is possible to run only a subset of the components, e.g., for peforming term extraction using only the specialized subgrammars or/and the phrasal grammars.

4.2 Creating domain-related abstracts

The content of a document is represented in an abstract. An abstract is composed of tuples where the elements are terms resulting from the term extraction. Heuristics determine which terms are paired on the base of the text and document structure. The employed keys for pairing two terms may be both linguistic

or extra-linguistic.

The application of heuristics is done in two stages. First, a set of positive heuristics suggests possible tuples. Currently implemented are the following positive heuristics:

- The *title heuristics* combines the terms between the starting and ending title tags with those from the rest of the document.
- The *sentence heuristic* conjoins all terms of a sentence with each other.
- The *NP-PP-heuristic* couples all directly consecutive sequences of nominal and prepositional phrases. It thus models minimal PP-attachment.

A merger takes care that every suggested pairing appears only once in the set of tuples, but it keeps track by which heuristics a tuple was suggested. Thus, if a tuple was combined by more than one heuristic, it can be regarded as more relevant.

In a subsequent step, negative heuristics act as filters on the output of the positive heuristics. There are two negative heuristics.

- The *instance heuristics* rejects all tuples which bear the same instance name. This would be the case for pairings of identical place names.
- The *coordination heuristics* rules out tuples that stem from one coordinated noun phrase. In the example sentence "Das Hotel hat einen Swimmingpool und eine Dachterasse.", the sentence heuristic would come up with three tuples, of which we reject <Swimmingpool, Dachterasse>.

4.3 SMES: Example

Let us consider the information on the Internet page
http://www.all-in-all.com/1165.htm
(cf. also appendix A). There, we have a hotel page where a hotel gives it's description.
The construction of an according abstract is shown in the following steps. The step's output data can be seen in the appendix.

Text filtering and term analysis. First, the HTML text will be filtered and has only some attributes than. There are only tags like table tags which are needed for keeping clustering information or which are usable for the interpretation in general (like the title tag).
We did a term analysis on the basis of the filtered text. The output is in XML format and will be used for constructing the abstracts.

Constructing an abstract. In the next step an abstract will be constructed on the basis of the term analysis. Ttuple pairs will be extracted and

coordinated.
The final abstract can be found in appendix A. Equal pairs are put together. This abstract will be stored into the database and used for answering user inqueries.

5 Integrating SMES into Harvest

The aim of the GETESS Gatherer is to construct a linguistic search index in combination with a keyword index [3]. In reason of using Harvest for constructing the keyword index we have to analyze the possibilities of integrating the SMES information extraction.

Demands. For the use of the SMES system we defined the ffollowing demands:

- modularity/ flexibility: In reason of changes of the SMES system during the development of the project it is necessary to have a great modularity and flexibility. Internal changes of the SMES system should not have directly consequences for the GETESS Gatherer. On the other hand changegs of the GETESS Gatherer should not have directly consequences for the SMES system.
- parallelism: The gathering process, especially the work of the Internet agent, and the SMES analysis should work in parallel. Because the process of gathering keywords is actually a time consuming process the SMES analysis should be possible in parallel to all the other GETESS Gatherer processes.
- the use of a database: The results of the SMES analysis should be stored into a database. The kind of the database can be different. So it is possible to have a full-text database system (like e.g. the database system Fullcrum), an object-relational database system (like e.g. DB2) or only a file system. The resulting abstracts of the SMES analysis should be stored in one of these systems. The SMES analysis should not be connected to the different database systems.

Connecting the GETESS Gatherer and SMES. The approach of connecting the GETESS Gatherer with the SMES system is shown in figure 4. We have two generel parts: the SMES client and the integration as Harvest summarizer.

- The SMES client. Resulting from the demands the GETESS gatherer acts with the SMES system via a client-server-architecture.

- Connecting the GETESS Gatherer and SMES using the Essence system. An important part of the Harvest Gatherer tool is the Essence system (cf. [6]).. The aim of the Essence system is to summarize information. Starting from a known information type (e.g. HTML, PS, ...) special summarizer are

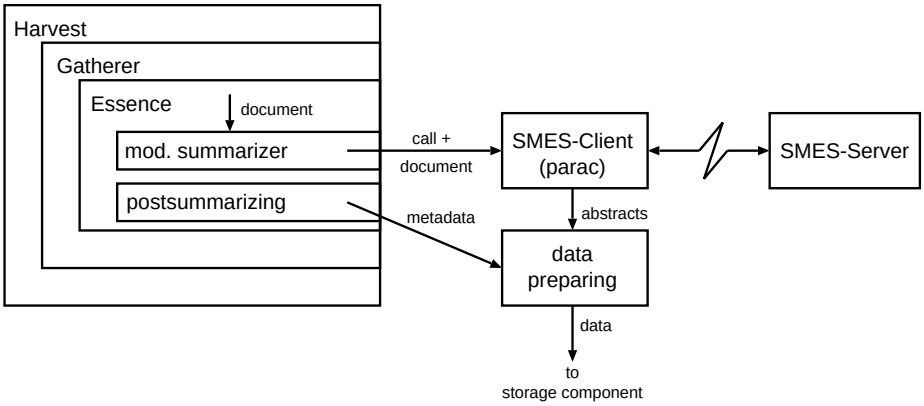

Fig. 4. Integration of SMES into Harvest

analyzing the formated text in order to find keywords.
We decided to integrate the SMES analysis as a special summarizer. So, an HTML text will first analyzed through the HTML summarizer and then analyzed with the SMES system. The results of the SMES analysis (in XML format) are stored and than converted into a database entry.

Implementational aspects. The GETESS gatherer with the SMES system is implemented and is running periodically.
The system needs for analyzing a HTML page 40 seconds. This time includes the time of the Internet agent, the keyword analysis, the SMES analysis and the time of storing the abstracts into the database.
The system is analyzing the Internet site www.all-in-all.com completely (about 4000 pages in German and English) and is actually tested with other Internet sites of the tourism domain.

6 Conclusion

In this paper we have shown how an Internet search engine can use linguistic and domain-related knowledge. W have illustrated the GETESS search engine and especially the GETESS gatherer. The GETESS gatherer collects data from the Internet in order to them for the search engine. In GETESS, in contrast to other search engines, not only keywords are used to answer user inqueries. GETESS focus on a linguistic and domain-related approach in which so called *abstracts* are constructed. These *abstracts* are an conceptual description of a specific Internet information and are the basis for answering the user queries.
Future work will consider on clustering algorithms which can be exploited for

clustering logical related Web pages which should be analyzed together.

References

1. Bradley, N.: The XML companion. Addison Wesley,Harlow, England, 1998
2. R. Baeza-Yates, B. Rebeiro-Neto: Modern Information Retrieval. ACM Presse, Addison Wesley, New York, 1999
3. I. Bruder: Integration von Harvest und SMES in GETESS. Studienarbeit, Universität Rostock, Fachbereich Informatik, 1999 (in German)
4. I. Bruder: Integration von Kontextwissen in den GETESS-Gatherer. Diplomarbeit, Universität Rostock, Fachbereich Informatik, 2000 (in German)
5. A. Düsterhöft, S. Gröticke: A Heuristic Approach for Recognizing a Document's Language Used for the Internet Search Engine GETESS. In: Proc. of the 2nd International Workshop on Natural Language and Information Systems, September 2000, Greenwich, UK
6. Harvest-System: http://harvest.transarc.com
7. G. Neumann, R. Backofen, J. Baur, M. Becker, C. Braun: An Information Extraction Core System for Real World German Text Processing. In: Proc. of the ANLP'97, March 1997, Washington, USA, 208-215,
8. G. Neumann, J. Piskorski, C. Braun: A Divide-and-Conquer Strategy for Shallow Parsing of German Free Texts. In: Proc. of the 6th International Conference of Applied Natural Language, April 2000, Seattle, USA
9. M. Klettke, H. Meyer: Managing XML documents in object-relational databases. University of Rostock, Rostocker Informatik-Berichte, No. 24, 1999
10. S. Staab, C. Braun, I. Bruder, A. Düsterhöft, A. Heuer, M. Klettke, G. Neumann, B. Prager, J. Pretzel, H.-P. Schnurr, R. Studer, H. Uszkoreit, B. Wrenger, "A System for Facilitating and Enhancing Web Search", In: *IWANN'99 - Proceedings of International Working Conference on Artificial and Natural Neural Networks*, Alicante, ES, 1999, LNCS, Berlin, Heidelberg, Springer
11. S. Staab, C. Braun, I. Bruder, A. Düsterhöft, A. Heuer, M. Klettke, G. Neumann, B. Prager, J. Pretzel, H.-P. Schnurr, R. Studer, H. Uszkoreit, B. Wrenger, "GETESS - Searching the Web Exploiting German Texts", In: *CIA'99 - Proc. of the 3rd Int. Workshop on Cooperating Information Agents.* Upsala, Schweden, 1999, LNCS, Berlin, Heidelberg, Springer
12. Search Engine Watch: Tips About Internet Search Engines. http://searchenginewatch.com/
13. Understanding and comparing web search tools http://web.hamline.edu/administration/libraries/search/comparisons.html
14. Search engines shoot-out - Top engines compared. http://coverage.cnet.com/Content/Reviews/Compare/Search2/
15. J. Timm: Speicherung von XML-Dateien in Objekt-relationalen Datenbanken. Diplomarbeit, Universität Rostock, Fachbereich Informatik, 1999 (in German)