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

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

[1] University of Rostock, Computer Science Department
A.-Einstein-Str. 21
18051 Rostock, Germany
{duest,ilr}@informatik.uni-rostock.de
[2] DFKI GmbH
Stuhlsatzenhausweg 3
66123 Saarbrücken, Germany
{neumann,beders,mbecker}@dfki.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 manner than usual. The result of the document analysis is called an *abstract*, and will it be used as a linguistic search index for the Internet search engine, GETESS.
We show how the linguistic analysis system SMES can be used with 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 inquiries.

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 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 (cf. [11]).

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

Internationally, the English language has established itself in the Internet. Most information in the Internet in Germany is also presented in English and

German. In German-speaking areas, the German language is playing an increasingly subordinate role in the Internet. Presently, a typical Internet user is either an information expert, student or computer freak with some knowledge 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] focuses on the development of an Internet search engine which analyzes information in German within a defined domain.
The project began with the idea of combining Internet techniques with database and knowledge representation methods, as well as results from the computer linguists. The GETESS architecture subsequently integrates these aspects in order to give more detailed information to the user.

Therefore, the starting point is the idea to combine Internet techniques with database and knowledge representation methods as well as results from the cmputer linguists. The GETESS architecture integrates the different aspects in order to give more detailed information to the user.
Users can formulate their queries using natural–language phrases, and they will be supported by an ontology. A query's result set consists of so–called *'abstracts'* that are summaries of the contents of web documents.

After motivating the main ideas behind GETESS in *section 2* we illustrate the functionality of the GETESS system, the GETESS Gatherer and the architecture of the information extraction system, SMES in *in section 4*. In *section 5* we discuss integrating SMES into the GETESS-Gatherer and its actual implementation.
In *section 6* we provide a conclusion and consider future work.

2 Motivation

Internet search engines are actually able to answer inquiries (cf. [11]) like

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

The possibilities outlined above for describing queries do not satisfactorily meet the users' needs. Users want an easy-to-formulate (natural) query language that allows to describe questions such as

- "ich suche eine Unterkunft an der Küste zwischen Rostock und Wismar" ("I'm looking for an accomodation on the coast between Rostock and Wismar")

  Additionally, a user may also want the following information:

[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, the University of Karlsruhe, DFKI Saarbrücken, Gecko mbH Rostock, and the University of Rostock.

  - an accomodation that can be a hotel or an appartment or a youth hotel or ...
  - an accomodation *between* Rostock and Wismar (Rostock and Wismar are cities)
  - an accomodation on the coast, in combination with the geographical information that Rostock and Wismar are cities on the Baltic Sea, one can assume that the user wants to be on the coast of the Baltic Sea

  The essential information behind the user inquery has to be extracted, because the user does not explicitly provide it. Consequently, it's important for a search engine to support a user in analyzing the user's inquery.
- "ich möchte ein Doppelzimmer unter 150 DM"
  ("I want to have a double room that costs no more than 150 DM")
  A search engine capable of answering this query must analyze the Web documents in such a way, so that prices can be read as integer values. In practice, most prices on the Web are integrated into text as strings. An indepth analysis of the text is necessary to classify the correct strings as prices. The same problem we have for addresses of institutions (e.g., hotels), special multi-valued sets (e.g., the set of credit cards), times (e.g., opening times), etc.
  However, the problem can be solved when a Web information in a particular domain (e.g., tourism) is analyzed and classified within the context of that domain. Important information can then be extracted and stored in the correct way (e.g., for comparisons).
- "ich suche ein Doppelzimmer mit Swimmingpool"
  ("I'm looking for a double room with a swimmingpool ")
  The user's inquery requests a double room in a hotel (or other establishment) where a swimming pool is located. A problem in answering this inquery because information about the rooms are often stored on different Web pages. Therefore, a keyword search would never find this complex information because the relevant keywords are not on the same Web page.
  This illustrates why information from a particular institution must be connected, i.e., its Web pages must be logically joined in order to answer such questions.

In the GETESS project, we use a domain-related approach in which all specific appearances of items are seen as representations of that particular item. So that specific words that are important in our domain are mapped onto concepts. For instance 'Master card' is a specific appearance of 'credit card'. The search engine subsequently uses these concepts in order to provide search results. The search engine also maps the natural-language user inquiries onto concepts and tries to match the concepts in the analyzed Web documents.

3 Collecting data from the Web - The GETESS Gatherer

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 the description of search requirements in a natural-language. Intelligent systems situated

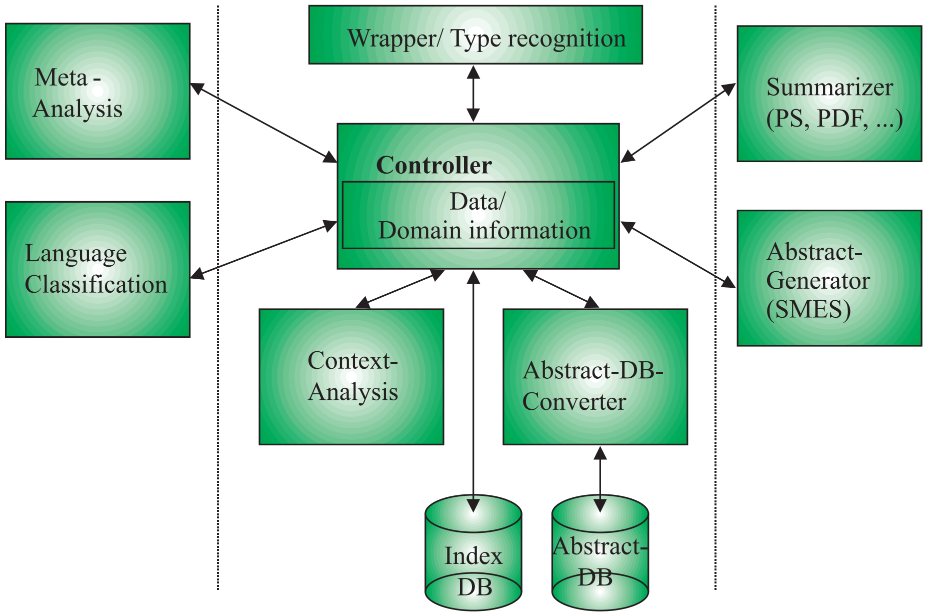

Fig. 1. GETESS Gatherer

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 will subsequently be used as results for queries. Results will then be translated into the user's mother tongue.
A fundamental requirement for developing 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 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 controller's task is to coordinate the work of the agent with the other Gatherer tools.

Harvest - The Basis of the GETESS-Gatherer. Harvest is actually one of the most imortant search engines, and it serves as the foundation for the GETESS Gatherer[2]. The GETESS Internet agent, the keyword extraction mechanism, and the summarizers are largely based on the Harvest system.
Parallel to the keyword extraction the GETESS gatherer initiates the information extraction (SMES). In any case, we have results either in the form of abstracts based on a linguistic analysis or results based on a keyword analysis. Because of this, we want to integrate the information extraction system, SMES, into the Internet agent. Otherwise additional information collected by specific GETESS tools has to be made available to the harvest components, e.g., meta-information.

Meta-analysis. During 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 prior to the information extraction in order to find collections of similar documents and special URL-strings.
Document collections are created when documents have comparable HTML structures. An approximation algorithm [3] is used to decide whether a similarity exists betweeb documents.
The meta-analysis agent collects this heuristic information and makes it available for the keyword and information extraction.

Language Classification. The language classification is used to define the document's language [4]. 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 in XML format, a special conversion into database structures is needed. A discussion about possible database systems and mechanisms for storing the abstracts can be found in [8].

The GETESS Gatherer runs periodically and collects data in the Internet. The result is a linguistic index in abstract form that is stored into a database. Furthermore, a keyword index is constructed for the linguistic analysis. Both indexes are used in the natural language dialogue system in order to answer user inquiries.

---

[2] "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." [1]

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. [6], [7]). The system's design provides a set of basic powerful, robust, and efficient natural-language components and generic linguistic knowledge sources that 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., stems, prefix, inflectional endings). Besides 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 that supports recognition 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, such as proper name grammars (e.g., organizations, complex time/date expressions, person names), generic phrasal grammars (nominal and prepositional phrases, and verb groups), and clause level grammars. Using WFST supports efficient and robust representation of each grammar modul.

4.1 The SMES Architecture

The architecture of the SMES system (shown in figure 2) consists of two major components, the Linguistic Knowledge Pool (LKP) and 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 first processed by the *text scanner*. Applying regular expressions, the text scanner identifies some text structures (e.g., paragraphs, indentations) and 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, that is identified as a potential word form 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

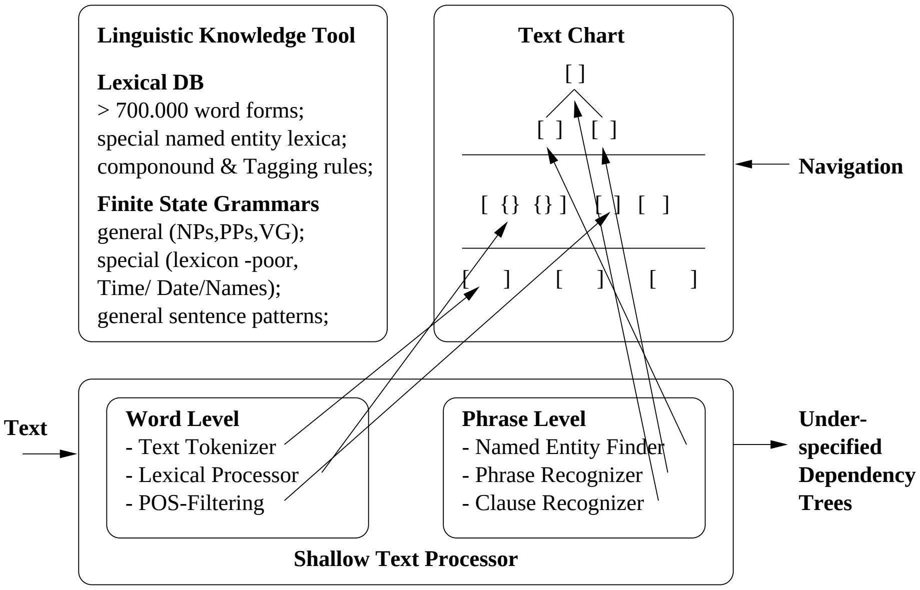

Fig. 2. The blue print of the SMES system architecture

disambiguation. Morphological analysis follows text scanning and performs inflection, and processes compounds. The capability of efficiently processing compounds is crucial because compounding is a very productive process of the German language. The output after morphology 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 and 2800 words/sec with compound processing, where all lexically possible decompositions are computed for each compound.

Chunk parser. The chunk parser is subdivided into three components.In the 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 sentence fragments 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 using a large subcategorization lexicon.

SMES 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, special sub-grammar for recognizing unknown proper names on the fly and general grammar for recognizing nominal phrases, prepositional phrases, and verb groups. Complete processing of 400-word text takes about 1second. Most important as it relates to this paper, SMES has a high degree of modularity: each component can be used in isolation. Thus, it is possible to run just a subset of the components, e.g., using only the specialized subgrammars and/or the phrasal grammars for peforming term extraction.

4.2 Creating domain-related abstracts

A document's content is represented in an *abstract*. An *abstract* is composed of tuples whose elements are the results of the term extraction. Heuristics determine which terms are paired on the basis of text and document structures. The employed keys for pairing two terms may be either linguistic or extra-linguistic.

Heuristics are applied in two stages. First, a set of positive heuristics suggests possible tuples. The following positive heuristics are currently implemented:

- The *title heuristic* 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 allows every suggested pairing to appear only once in the set of tuples, however, it tracks every positive heuristic that suggested a tuple. Therefore, if tuple was suggested 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 heuristic* rejects all tuples that bear the same instance name. This would be the case for pairings of identical place names.
- The *coordination heuristic* 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, and we would reject <Swimmingpool, Dachterasse>.

5 Integrating SMES into Harvest

The aim of the GETESS Gatherer is to construct a linguistic search index in combination with a keyword index [2]. Because Harvest is used for constructing the keyword index, we have to analyze the possibilities of integrating the SMES information extraction.

Demands. We defined the following demands for using the SMES system:

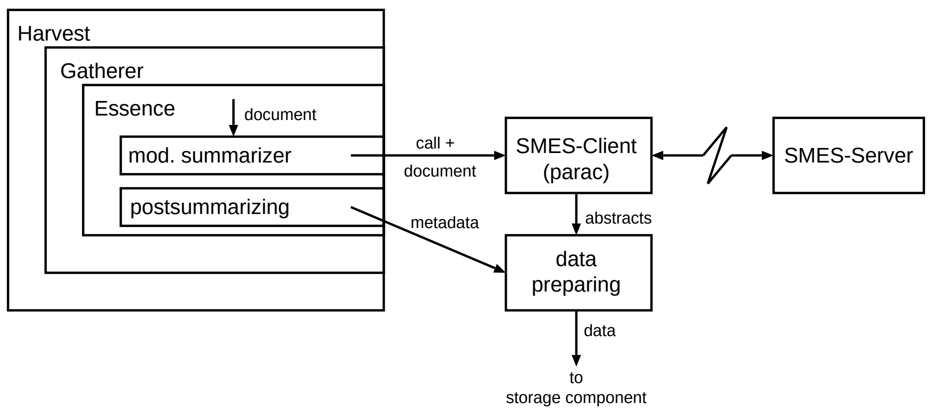

Fig. 3. Integration of SMES into Harvest

- modularity/ flexibility: Because of potential changes in the SMES system during project development, a high degree of modularity and flexibility are required. Internal changes in the SMES system should not have direct consequences for the GETESS Gatherer. Similarly, changes in the GETESS Gatherer should not have direct consequences for the SMES system.
- parallelism: The gathering process (especially the work of the Internet agent) and the SMES analysis should work parallel. Because the process of gathering keywords is very time consuming, the SMES analysis should take place parallel to the other GETESS Gatherer processes.
- database-use: The results of the SMES analysis should be stored into a database. The type of database can differ. So it is possible to have a full-text database system (like the database system Fullcrum), an object-relational database system (like DB2), or only a file system. The resulting abstracts of the SMES analysis should be stored in one of these systems and should not be connected to different database systems.

Connecting the GETESS Gatherer and SMES. Figure 3 depicts the approach for connecting the GETESS Gatherer with the SMES system. We have two general parts: the SMES client and the integration as Harvest summarizer.

- The SMES client. The GETESS gatherer interacts with the SMES system via a client-server architecture.
- The GETESS Gatherer and SMES are connected using the Essence system, which is an important part of the Harvest Gatherer (cf. [5]). The aim of the Essence system is to summarize information. Starting from a known information type (e.g., HTML, PS, ...), special summarizers analyze the formatted text in order to find keywords. We decided to integrate the SMES analysis as a special summarizer. Therfore, an HTML text is 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 thrn converted into a database entry.

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

Let us consider the following example, the source of http://www.all-in-all.com/1165.htm.

Hotel Mecklenburger Hof
D-19288 Ludwigslust
Lindenstrae 40 - 44
Tel. 03874/ 410-0
Fax: 03874/ 410-100

Das Hotel Mecklenburger Hof liegt verkehrsgnstig in der Innenstadt nahe dem Ludwigsluster Schlo mit seinem Schlopark und der Schlokirche.

Hausbeschreibung:
Komfort auf hohem Niveau in 37 Zimmern mit 72 Betten. Das Haus wurde mit viel Liebe zum Detail restauriert und bietet seinen Gsten stilvolle gemtliche Gastlichkeit, verbunden mit einem sehr aufmerksamen Service, einer ausgezeichneten Kche und exzellenten Frhstcksbufett. Ein gut sortierter Weinkeller gehrt genauso dazu, wie der zweimal im Jahr stattfindende Operettenball und weitere interessante Veranstaltungen. Kostenfreie Parkpltze stehen Ihnen in der Tiefgarage zur Verfgung.

Preise:
Einzelzimmer: von DM 88,00 bis DM 120,00
Doppelzimmer: von DM 130,00 bis DM 185,00

Kreditkarten: EC, Visa, Diners

The following abstract is the result of analyzing the source:


```
<Getess>
  <HOTEL>
     <HOTEL_1> hotel Mecklenburger Hof   </HOTEL_1>
  </HOTEL>
  <ORT>
     <ORT_1> Ludwigslust   </ORT_1>
     <ORT_2> Ludwigslust (linde-PL-N-strasse) </ORT_2>
  </ORT>
  <GEBIET>
     <GEBIET_1> Mecklenburg   </GEBIET_1>
  </GEBIET>
  <ZIMMERAUSSTATTUNG>
     <ZIMMERAUSSTATTUNG_1> telefon   </ZIMMERAUSSTATTUNG_1>
     <ZIMMERAUSSTATTUNG_2> bett   </ZIMMERAUSSTATTUNG_2>
  </ZIMMERAUSSTATTUNG>
  <ZIMMER>
     <ZIMMER_1> zimmer   </ZIMMER_1>
     <ZIMMER_2> einzelzimmer   </ZIMMER_2>
     <ZIMMER_3> doppelzimmer   </ZIMMER_3>
  </ZIMMER>
  <PREIS>
     <PREIS_1> ez-min 88 </PREIS_1>
     <PREIS_2> ez-max 120 </PREIS_2>
     <PREIS_3> dz-min 130   </PREIS_3>
     <PREIS_4> dz-min 185   </PREIS_4>
  </PREIS>
  <PREIS_WAEHRUNG>
      <PREIS_WAEHRUNG_1> dm </PREIS_WAEHRUNG_1>
  </PREIS_WAEHRUNG>
  <UNTERKUNFT_ZAHLUNGSMOEGLICHKEIT>
     <UNTERKUNFT_ZAHLUNGSMOEGLICHKEIT_1> kredit-karte   </UNTERKUNFT_ZAHLUNGSMOEGLICHKEIT_1>
  </UNTERKUNFT_ZAHLUNGSMOEGLICHKEIT>
  <BETRIEB_ZAHLUNGSMOEGLICHKEIT>
     <BETRIEB_ZAHLUNGSMOEGLICHKEIT_1> ec   </BETRIEB_ZAHLUNGSMOEGLICHKEIT_1>
     <BETRIEB_ZAHLUNGSMOEGLICHKEIT_1> visa   </BETRIEB_ZAHLUNGSMOEGLICHKEIT_1>
     <BETRIEB_ZAHLUNGSMOEGLICHKEIT_1> diners   </BETRIEB_ZAHLUNGSMOEGLICHKEIT_1>
  </BETRIEB_ZAHLUNGSMOEGLICHKEIT>
  <HAUSBESCHREIBUNG>
     <HAUSBESCHREIBUNG_1> haus-beschreibung   </HAUSBESCHREIBUNG_1>
  </HAUSBESCHREIBUNG>
  <FERIENWOHNUNG_RAEUME>
     <FERIENWOHNUNG_RAEUME_1> kueche (auszeichn)   </FERIENWOHNUNG_RAEUME_1>
  </FERIENWOHNUNG_RAEUME>
  <GASTRONOMIE>
     <GASTRONOMIE_1> weinkeller (sortier) </GASTRONOMIE_1>
  </GASTRONOMIE>
</Getess>
```

6 Conclusion

In this paper, we have shown how an Internet search engine can use linguistic and domain-related knowledge. We have illustrated the GETESS search engine and, in particular the GETESS gatherer. The GETESS gatherer collects data from the Internet for the search engine. In contrast to other search engines, GETESS uses not only keywords to answer inquiries, but also focuses on a linguistic and domain-related approach in which so-called *abstracts* are constructed. These *abstracts* are conceptual descriptions of specific Web documents and are the basis for answering user inquiries.
Future work will consider on algorithms that can be exploited for clustering logically related Web pages that should be analyzed together.

References

1. R. Baeza-Yates, B. Rebeiro-Neto: Modern Information Retrieval. ACM Presse, Addison Wesley, New York, 1999
2. I. Bruder: Integration von Harvest und SMES in GETESS. Studienarbeit, Universität Rostock, Fachbereich Informatik, 1999 (in German)
3. I. Bruder: Integration von Kontextwissen in den GETESS-Gatherer. Diplomarbeit, Universität Rostock, Fachbereich Informatik, 2000 (in German)
4. 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
5. Harvest-System: http://harvest.transarc.com
6. 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,
7. 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
8. M. Klettke, H. Meyer: Managing XML documents in object-relational databases. University of Rostock, Rostocker Informatik-Berichte, No. 24, 1999
9. 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
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, "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
11. Search Engine Watch: Tips About Internet Search Engines. http://searchenginewatch.com/