GETESS — Searching the Web Exploiting German Texts[‡]

Steffen Staab[1], Christian Braun[2], Ilvio Bruder[3], Antje Düsterhöft[4], Andreas Heuer[4], Meike Klettke[4], Günter Neumann[2], Bernd Prager[3], Jan Pretzel[3], Hans-Peter Schnurr[1], Rudi Studer[1], Hans Uszkoreit[2], and Burkhard Wrenger[3]

[1] AIFB, Univ. Karlsruhe, D-76128 Karlsruhe, Germany
[2] DFKI, Stuhlsatzenhausweg 3, D-66123 Saarbrücken, Germany
[3] GECKO mbH, Koch-Gotha-Str. 1, D-18055 Rostock, Germany
[4] Universität Rostock, Fachbereich Informatik, D-18051 Rostock, Germany
`http://www.getess.de`

Abstract. We present an intelligent information agent that uses semantic methods and natural language processing capabilites in order to gather tourist information from the WWW and present it to the human user in an intuitive, user-friendly way. Thereby, the information agent is designed such that as background knowledge and linguistic coverage increase, its benefits improve, while it guarantees state-of-the-art information and database retrieval capabilities as its bottom line.

1 Introduction

Due to the vast amounts of information in the WWW, its users have more and more difficulties finding the information they are looking for among the many heterogeneous information resources. Hence, intelligent information agents that support the gathering and exploitation of web site contents are in the primary focus of a number of research communities these days. Currently, syntactic methods of information retrieval prevail in realistic scenarios (cf., e.g., [3]), such as in general search engines like AltaVista, but the limits inherent in these approaches often make finding the proper information a nuisance. On the other end of methodologies, semantic methods could provide just the right level for finding information, but they rely on explicitly annotated sources (cf., e.g., [7]) or on complete and correct natural language understanding systems, both of which cannot be expected in the near future.

Therefore our information assistant, GETESS[1], uses the semantics of documents in the WWW — as far as it is provided explicitly or as it can be inferred by an incomplete natural language understanding system, but relies on syntactic retrieval methods, once the methods at the semantic level fail to fulfill their task. In particular, we consider an information finding approach that, *(i)*, has semantic knowledge for supporting the retrieval task, *(ii)*, partially, but robustly, understands natural language, *(iii)*, allows for advanced ways of interaction that appear natural to the human user, *(iv)*, integrates

[‡] Corresponding author: S. Staab, e-mail: `staab@aifb.uni-karlsruhe.de`, tel.: +49-721-6087363, fax: +49-721-693717

[1] GErman Text Exploitation and Search System

knowledge from unstructured and semi-structured documents with knowledge from relational database systems, and, *(v)*, reasons about this knowledge in order to support the user dialogue and the retrieval of implicit knowledge.

In our project, we decided to aim at an information agent that provides information finding and filtering methods for a restricted domain, *viz.* for prospective tourists that may travel in a certain region and are looking for all kinds of information, such as housing, leisure activities, seesights, etc. The information about all this cannot be found within a narrowly restricted format — neither in a single database nor in a single web site. Rather, the information agent must gather information that is stored in an open and dynamic environment on many different web servers, often in unstructured text, and even in some databases, such as a booking database of a hotel chain. In order to improve on common information retrieval systems, at least part of what is stated in the (HTML) texts must be made available semantically. However, since automatic text understanding is still far from perfect, we pursue a *fail-soft* approach that is based on extracting knowledge from text with a robust parser, but also integrates and falls back onto common information retrieval mechanisms when the more elaborate understanding component fails.

In the following, we first give an example of how GETESS may assist an user in finding tourist information on the web that will be used for the purpose of illustration throughout the rest of this paper. Then, we draft the architecture of GETESS with its overall sharing of the work load. From this outline we will then motivate and describe some key issues of the major subsystems of GETESS.

2 Example Scenario

The GETESS intelligent assistant knows about a multitude of sites offering tourist information on the web. This first contact may be established by different means, such as by manual registration of sites through the information provider or by automatic classification methods that simply have to determine whether a web site offers tourist information. Our current testbed consists of a few hundred text documents which include different types of information that might be of interest to a prospective tourist. Examples are information about the regional administration or about activities in particular places. A typical example scenario that illustrates the range of services we want to offer to an information seeker is given in the following.

First, we have documents like (1a) and (1b) that contain the italicized propositions.

(1) a. … *The island Usedom belongs to Mecklenburg-Vorpommern.* …
    b. … *The Usedom music festival is a touristic highlight every summer.* …

Then, a tourist who is planning to travel the region might look for places and activities and she may pose queries like (2a) to (2c).

(2) a. *cultural events, Mecklenburg-Vorpommern*
    b. *Which cultural events take place in Mecklenburg-Vorpommern?*
    c. *I am searching for cultural events in Mecklenburg-Vorpommern.*

These example queries may differ in their types, but ultimately they are geared toward the same goal, *viz.* the retrieval of cultural events in Mecklenburg-Vorpommern, such as the one in Usedom. The GETESS intelligent assistant must understand this information request, integrate informations from different documents (e.g., it must integrate the propositions from (1a) and (1b) in order to retrieve the Usedom music festival as a correct answer) and present a list of references to the information seeker. The first three of this long list may be given in (3a) to (3c), together with some hints, (4), that may help the user to narrow down the choices she is looking for.

(3) a. *Rostock, Concert series of the Hochschule*[2] *für Musik und Theater, http://...*
b. *Usedom music festival, http://...*
c. *Ralswiek, Störtebecker theater festival, http://...*

(4) You might want to reduce your hit rate by refining your search to MUSIC EVENT, THEATER EVENT, SHOW EVENT, or EDUCATIONAL EVENT.

Then, the dialogue may continue with query (5):

(5) *What type of music is played on Usedom?*

Resulting in answer (6) by GETESS:

(6) The Usedom music festival features classical music.

And the user may proceed with

(7) *Show me folklore music events.*

This may result in

(8) There are no folklore music events known in Mecklenburg-Vorpommern. You might want to check out related categories like
a. *Folklore dance event: Ribnitz-Damgarten International Folklore dance festival, http://...*

This result here may happily suffice the information seekers request and serve as an ongoing illustration in the further course of this paper.

3 Architecture

The front end of the GETESS agent (cf. a depiction of its main modules in Figure 1) provides a user interface that is embedded in a *dialogue system* controlling the history of interactions (cf. Section 4). Single interactions are handed to the query processor that selects the corresponding analysis methods, *viz.* the natural language processing module (NLP system; also cf. Section 6) or the information retrieval and database query mechanisms (cf. Section 5). 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 it sends this formal query to the search system.

In order to process queries and search for results, three kinds of resources are provided by the back end of the GETESS assistant. First, archived 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

[2] College for music and theatre.

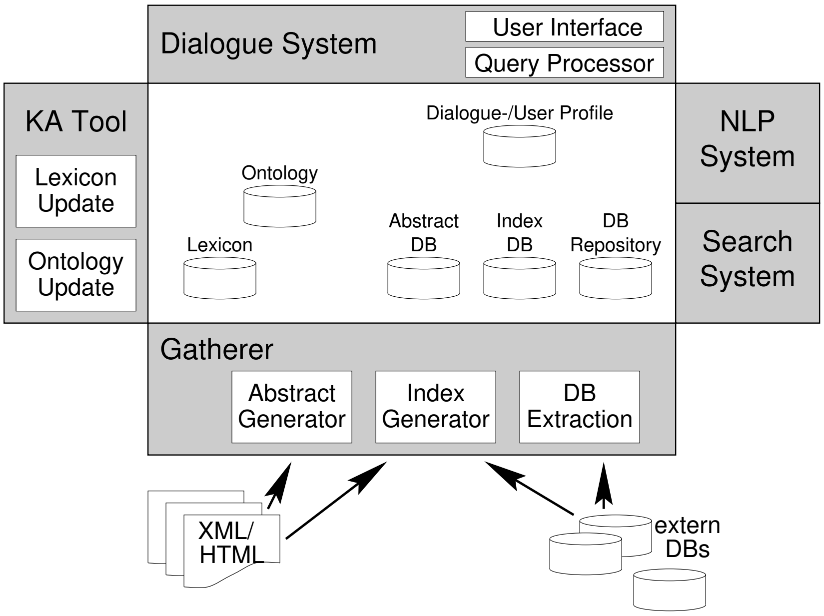

Fig. 1. The architecture of GETESS

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 interactions.

While dialogue sequences and user profiles are acquired during 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 contents of typical web sites change almost on a daily basis. For this task the *gatherer* searches regularly through relevant XML/HTML-pages and databases of specified sites 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, though incomplete natural language understanding module that parses documents and extracts semantic information building a so called "*abstract*" for each document. These abstracts are sets of facts, *i.e.* tupels, like `hostsEvent(Usedom,music-festival-1)`, that could be extracted from natural language text, like "The Usedom music festival ..." in example (1b). The index generator builds access information for full text search with information retrieval methods, while the DB repository offers relevant views onto extern databases.

Subsequently, we will first introduce the front end, the dialogue system. The key issues here are concerned with *facilitating* user interaction at different levels of expertise (Section 4). At the back end of the system, the tools for gathering, database management and information retrieval provide the technical platform for efficiently updating and communicating with the agent's information repositories (Section 5). The natural language processing component in GETESS is employed by the dialogue system as well as by the back end in order to understand natural language queries and extract information from natural language texts, respectively, and, thus, *enhance* the quality of the web search (Section 6). Finally, we outline the function of the ontology that constitutes the "glue" of the system at the semantic level (Section 7).

4 Dialogue System

The dialogue system constitutes the interface of the GETESS intelligent assistant. In order to facilitate the user's task of finding the information he is looking for, users should be able to communicate conveniently at their level of expertise. This means the agent should allow for intuitive interaction by natural language queries as well as for formal queries that may be the preferred mode of interaction by a human expert user. Indepent of the concrete mode of interaction, the GETESS assistant should react quickly and accurately while using the capabilities of the different modal actions.

In particular, GETESS provides four types of interaction, *viz.* natural language, graphical interface, keyword search and formal database query. Since the methods for natural language processing as well as for keyword search and formal database queries form major components, their description has been delegated to subsequent sections (6 as well as 5, respectively).

Thus, this chapter serves the following three goals: First, it is described how reasoning about user interactions may support the user's goal in quickly finding the appropriate information. Second, it is sketched how single interactions are treated as elements of a complex dialogue. Hence, the user does not have to start from scratch every time he inititiates a new query, but can instead refer to his previous queries, e.g. by requests like "Show me folklore music events." (7) that implicitly relates to *folklore music events in Mecklenburg-Vorpommern* — the location need not be restated explicitly. Finally, we give a glimpse on the use of the graphical interface.

The Knowledge Base of the Dialogue System. Knowledge is crucial for all modes of interaction, because we want the system to give appropriate responses to the user when problems arise. For instance, when a query results in an abundance of hits, the system must reason about why this problem might have occured and how it might be solved. Knowledge that allows for this type of reasoning is encoded in the *knowledge base of the dialogue system* (KBD).

The KBD includes all the definitions available in the ontology (cf. Section 7). These definitions help with explaining to the user why a query might have been too unspecific or with giving him hints how he might try to rephrase the query such that he gets the information he is looking for. For example, if the user seeks for information on the local offers of "`cultural events`", the hit rate for a database query may be reduced by the choice of one of the refined search terms "`music events`", "`theater events`", "`show events`" and "`educational events`" (cf. (4)). Vice versa, a too specific choice like "`folk music event`" might result in no hits, but the hint towards more general search terms might bring up similar events such as "`folklore dance event`" in (8a) that may be of interest. Further help is also provided through important terminological links such as synonyms, homonyms, antonyms and terms that may be parts of other terms, e.g. the show of a magician may be part of a circus show and, hence, the circus show might be a viable entertainment alternative to a magician's show.

In addition to the definitions of the ontology, the KBD features definitions and rules about dialogue concepts. At the moment, this part is tuned to map different interactions onto common requests to the database.[3] For example, the user input "Which cultural events take place in Mecklenburg-Vorpommern" (2b), which may also be supplemented by restrictions from the graphical interface, has the same meaning, i.e. it constitutes the same speech act, as "I am searching for cultural events in Mecklenburg-Vorpommern" (2c). Therefore, both inputs must be mapped onto the same query to the database.

Both types of knowledge provide user support that reduces the number of inquiries the user has to pose to the system and, hence, accelerates the dialogue compared to common keyword retrieval interactions.

Complex Dialogues. As indicated by examples (3) and (4), information finding rarely produces an instantaneous hit — after the user has formulated just a single query. This is true for syntactic methods and it will improve only to a limited amount with semantic methods either. However, when a user's sequence of interactions is perceived as being executed in order to achieve a goal, such as illustrated by the user's requests in Section 2, then this task of finding the proper information can be substantially facilitated. For this purpose, we provide a query processor that analyses not only the single interactions, but also views them as being embedded into a more global structure.

The methodology we use is based on work done by Ahrenberg et al. [1], who structure the dialog hierarchically into segments that are opened by a request and closed by the appropriate answer. Our assumption here is that users typically have a request for a certain piece of information and give related information in order to succeed. For example, they give a *topic*[4], which here boils down to a type restriction, like "cultural event", and spatial information about where these events should occur. The task of the dialogue system lies in zooming in or out on relevant information according to the interaction initiated by the user. For example, two user interactions[5] like, *(i)*, "Which cultural events take place in Mecklenburg-Vorpommern?" (2b), and, *(ii)*,"Show me folklore music events" (7) return a large set of documents first (with feedback such as in (4)), but a much smaller set of data after the second interaction has narrowed down the focus.

Hence, we here identify *dialogue segments*, *interactions* and *topics* as the major parameters (though not the only ones) that the dialogue system keeps track of. By this way the user's single interactions may all convey towards the common information finding goal and, thus, facilitate the human-computer interaction.

The Graphical Interface. Besides of the natural language query capabilities and the possibility of directly composing a formal query, the GETESS system features a graphical interface. This interface constitutes an intermediate level of access to the system

---

[3] In the linguistic literature this mapping is defined by the way natural language propositions, requests or questions can be considered as so called *speech acts* [2].

[4] The "topic" in a dialogue corresponds to what the dialogue is about. Usually, it is given only implicitly in natural language statements. In our setting it may also be given explicitly, *e.g.* through the graphical interface.

[5] Each interaction corresponds loosely to a speech act as introduced above, but might also be an act in the graphical user interface. Examples are *update* (users provide information to the system), *question*, *answer*, *assertion* or *directive*.

between the most professional (and fastest) one, *viz.* the formal query, and the most intuitive one (that requires a somewhat more elaborate interaction), *viz.* the natural language access. The graphical interface does not require from the user to learn the syntax of a particular query language or the concepts that are available in the ontology, but expects some basic understanding of formal systems from the user. This interface visualizes the ontology in a manner suited for selecting appropriate classes and attributes and, thus, allows the assembly of a formal query through simple mouse clicks. For this purpose, the ontology is visualized by a technology based on hyperbolic geometry [14, 9]: classes in the center of the visualization are represented with a big circle, surrounding classes are represented with a smaller circle. This technique allows fast navigation to distant classes and a clear illustration of each class and its neighboring concepts.

5 Gathering, Database Management and Information Retrieval

In this section, we outline the back end of GETESS that gathers data from the web and stores it in a way that allows for efficient retrieval mechanisms as far as keyword search and formal queries are concerned.

The back end of GETESS employs a typical gatherer-broker structure, *viz.* a Harvest search service [4] with a database interface. Though we use the tools provided by another project, SWING [13], the setting of GETESS puts additional demands on the gatherer-broker system: *(i)*, the GETESS search engine has to work with facts contained in the abstracts, *(ii)*, ontology knowledge must be integrated into the process of analysing internet information as well as answering user queries, *(iii)*, internet information can be of different types (*e.g.*, HTML, XML texts), and, *(iv)*, data collections such as information stored in databases must also be accessible via the GETESS search engine. These different requirements must be met both during the main process of gathering data and during the querying process (broker).

The gatherer process. Periodically, internet information is analysed via internet agents in order to build a search index for the GETESS search engine. Information (*e.g.* HTML-texts, Postscript-files, ...) is checked to find keywords. Additionally, the GETESS gatherer initiates the abstract generation on these information (cf. Sections 6, 7). The two kinds of index data ('simple' keywords and abstracts) are stored in databases.

The broker process. As indicated above, the dialogue system maps the user's queries (with the help of the natural language processing module and the definition in the ontology) onto formal or keyword queries in IRQL, the Information Retrieval Query Language [12], which is based on SQL99. The IRQL language combines different kinds of queries — both database and information retrieval queries — thus providing access to the index data. The query result set may be ranked with a function the shape of which will have to be determined by future experiments. The ranked result set is then presented to the user via the dialogue system.

The integration of different types of information (full text, abstracts, relational database facts) during gathering and querying has put forth and still requires demand for research, however at the same time it raises new possibilities of posing queries, because:

1. Conventional search engines support an efficient search for keywords or combinations of keywords over a whole document. This is still possible, but the GETESS

abstracts also provide relational information for the documents. Exploiting this type of information, we can realise attributed queries. That means users have the possibility to search for terms in special attributes, such as for instances of a particular concept.

2. Searching for particular integer values, for instance prices or distances is nearly impossible with a conventional information retrieval approach. The GETESS assistant will allow the comparison of integer and real values, *e.g.* to search for all prices that fall below a threshold. In addition, one may also determine minimum, maximum and average values as well as sort and group results by particular values.
3. Database functionality brings up answers from the abstract databases that are composed of different abstracts. That means, for answering a query of an user we may refer and exploit facts derived from different websites. Thereby, it is not even necessary that these websites are connected by links, but all the algebraic operations given through database functionality can be employed in order to deduce information. For instance, a music festival may be announced on one web site (1a), the corresponding reviews are found in another one. Database technology allows for retrieving all the events that take place in a particular location like Usedom and that also received a good note in the corresponding reviews.

This functionality is implemented in an object relational database system. It will be employed in a distributed database solution, which provides for data storage at different local servers. Having made available different languages for accessing this repository of information, we are now researching a common language level, the IRQL described above, for accessing structured information (abstracts) and unstructured information (for instance HTML or XML) with the same interface. This language will then reduce the burden on the dialogue system, because the dialogue system will not have to distinguish between formal and keyword search anymore. Thus, IRQL will enhance the overall robustness of the system.

6 Natural Language Processing

The GETESS intelligent assistant uses the natural language processing (NLP) component in order to, *(i)*, linguistically analyse user queries specified in the dialogue system, *(ii)*, to generate the linguistic basis for the extraction of facts from NL-documents, and, *(iii)*, generate natural language responses from facts in the abstract database.

The design of the NLP component is based on two major design criteria: First, the GETESS system requests a high degree of robustness and efficiency in order that the agent may be applied in a real-world setting. The reason is that we must be able to process arbitrary sequences of strings efficiently, because "broken" documents appear on real web sites, and that the number of documents that must be processed is too large in order to allow for response times of several minutes per sentence. Second, we employ the same shallow NL core components and linguistic data managing tools for processing texts and extracting information as well as for analysing a user's query. Thus, we can reduce the amount of redundancies as far as possible and keep the system in a consistent state with regard to its language capabilities. For instance, when the internal linguistic representation of an NL query and the abstracts use the same data sources, and if we also use the same knowledge sources for NL-based generation, inconsistencies as a result of unshared data can be reduced.

For the purpose of a short presentation here, we abstract from two major components of the natural language processing component in GETESS. We do not elaborate on the natural language generation part that also includes features for summarizing facts from the abstract database. Moreover, we are well aware that our project serves an international tourist community and, therefore we will have to add multi-lingual access as well as multi-lingual presentations of the query results. However, at the current state of the project, we focus on parts of Germany as the touristic goal that we want to provide information about and, hence, focus on the analysis of German documents only.

Shallow text processing. The shallow text processor (STP) of GETESS is based on and an extension of SMES, an IE-core system developed at DFKI (see [16, 17]). One of the major advantages of STP is that it makes a clean separation between domain independent and dependent knowledge sources. Its core components include: *(i)*, a text scanner, which recognizes, *e.g.*, number, date, time and word expressions, as well as text structure, like sentence markers and paragraphs; *(ii)*, a very fast component for morphological processing which performs inflectional analyses including processing of compounds and robust lexical access (*e.g.* analysing "events" as the plural of "event"); *(iii)*, a chunk parser based on a cascade of weighted finite state transducers. The chunk parser performs recognition of phrases (specific phrases like complex date expressions and proper names, and general phrases, like nominal phrases and verb groups), collection of phrases into sentences, and determination of the grammatical functions (like the deep cases, which determine the direction in which a particular relation holds; *e.g.* in example (1a) Mecklenburg-Vorpommern encloses Usedom and not vice versa).

STP has large linguistic knowledge sources (e.g., 120.000 general stem entries, more than 20.000 verb–frame entries). The system is fast, and can process 400 words in about one second running all components. In order for adapting STP for the GETESS assistant, we have evaluated STP's coverage on a corpus provided by our industrial partner. Though evaluation is blind, because the current knowledge sources have not been specified using any part of this corpus, we could analyse over 90% of all word forms and found that a majority of the remaining forms can be covered by domain specific lexica.

Extraction of facts. Finally, a word on the extraction of facts — and thus the generation of abstracts: The STP generates a linguistic analysis, *i.e.* it determines syntactic relations between words, *e.g.* between a verb and its subject. How these linguistic cues are exploited in order to go from natural language to a formal description is explained in the following section that elaborates on the semantic level of GETESS.

7 Ontology

As already mentioned, the scope of syntactic methods for gathering, using and querying of information is very limited and often unsatisfactory. A semantic reference model, an ontology, which structures the content and describes relationships between parts of the content helps to overcome these limitations. With the ontology in GETESS, we aim at two major purposes: First, it offers inference facilities that are exploited by the other modules, as, *e.g.*, described in Section 4, the dialogue module may ask for the types a particular instance belongs to in order to present alternative query options to the user. Second, the ontology acts as a mediator between the different modules. This latter role is explained here in more detail, since it illustrates how ontological design influences the working of the GETESS agent and, in particular, the extraction of facts from natural language texts.

The text processing (cf. Section 6) of natural language documents and queries delivers syntactic relations between words and phrases. Whether and how this syntactic relation can be translated into a meaningful semantic relation, depends on how the tourism domain is conceptualized in the ontology. For example (cf. Fig. 2), the natural language processing system finds syntactic relations between the words "music festival" and "Usedom" in the phrase "Usedom music festival". The word "Usedom" refers to `Usedom` which is known as an instance of the class `Island` in the database. "music festival" refers to a — so far — unknown instance, `music-festival-1` of the class `MusicFestival`. The database refers to the ontology for the description of the classes `Island` and `MusicFestival`. Querying the ontology for semantic relations between `Island` and `MusicFestival` results in `hostsEvent`, which is inherited from the class `Region` to the class `Island`. Then, a corresponding entry between `Usedom` and `music-festival-1` is added to the abstract, *i.e.* the set of extracted facts, of the currently processed document in the abstract database.

This example shows that the design of the ontology determines, *(i)*, the facts that may be extracted from texts, *(ii)*, the database schema that must be used to store these facts and, thus *(iii)*, what information is made available at the semantic level. Hence, the ontology might constitute an engineering bottleneck. However, we try to overcome this problem by a dual approach. On the one hand, we try to ease the burden on the knowledge engineer by a graphical, easy-to-use, knowledge engineering interface in the line of Protege-II [8]. On the other hand, we relieve the decision of which concepts to model by using the linguistic and statistical analyses of the text processing component. These analyses do not only indicate frequent, though yet unmodelled, concepts, but they also help to narrow down the choice of where new concepts should be placed in the ontology. Thus, the composition as well as the adaptation of the ontology is facilitated.

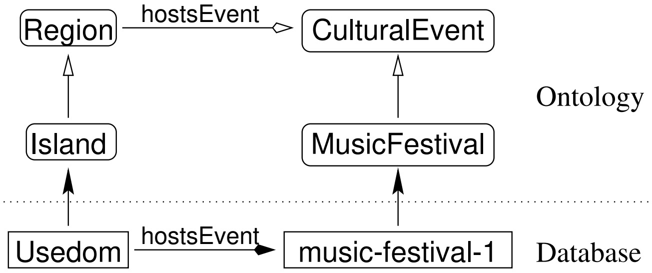

Fig. 2. Interaction of Ontology with NLP system and Database Organization

8 Related Work

The GETESS project builds on and extends a lot of earlier work in various domains. In the natural language community, research like [10] fostered the use of natural language application to databases, though these applications never reached the high precision and generality required in order to access typical databases, e.g. for accounting. Here, our approach seems better suited, since some of the deficits of natural language understanding techniques are counterbalanced by information retrieval facilities and an accompanying graphical interface.

Only few researchers, e.g. Hahn et al. [11], have elaborated on the interaction between natural language understanding and the corresponding use of ontologies. We think this to be an important point since underlying ontologies cannot only be used as submodules of text understanding systems, but can also be employed for a more direct access to the knowledge base and for providing an intermediate layer between text representation and extern databases.

As far as queries of conceptual structures are concerned, we agree with McGuiness & Patel [15] that usability issues play a vital role in determining whether a semantic layer can be made available to the user and, hence, we elaborated on this topic early on [9]. We, thereby, keep in mind that regular users may find lengthy natural language questions too troublesome to deal with and, therefore, prefer an interface that allows fast access, but which is still more comfortable than any formal query language.

Projects that compare directly to GETESS are, *e.g.*, Paradime [16][6], MULINEX [6] and MIETTA [5]. However, none of these projects combines information extraction with similarly rich interactions at the semantic layer. Hence, to the best of our knowledge we are the only one integrating unstructured, semi-structured and highly-structured data with a variety of easy-to-use facilities for human-computer interaction.

9 Conclusion

In the project GETESS (GErman Text Exploitation and Search System) we decided to build an intelligent information finder that relies on current techniques for information retrieval and database querying as its bottom line. The support for finding informations is *enhanced* through an additional semantic layer that is based on ontological engineering and on a partial text understanding component.

In order to *facilitate* web search, intuitive communication with the GETESS agent is considered a crucial point. Here, analyses of complex dialogues allow for refining, rephrasing or refocusing succeeding queries, and thus eliminate the burden of starting from scratch with every single user interaction. Thereby, several modes of interaction are possible, besides keyword search and SQL-queries one can mix natural language queries with clicking in the graphical query interface.

Having built the single modules for our system, the next task is bringing these components together. Given the design methodology of achieving entry level features first and then working towards "the high ceiling" (*viz.* complete text understanding and representation), we expect benefits on the parts of economic and research interests early in the project. The system is general enough in order to be applied to many realistic scenarios, e.g. as an intelligent interface to a company's intranet, even though it is still far from offering a general solution for the most general information finding problems in the WWW. Further research will have to address many of the remaining open issues, such as an evaluation of the overall system, a general methodology for mapping ontological models to database schemata or semi-automatic modeling of ontologies from example texts.

[6] Actually, GETESS uses the same linguistic core machinery as Paradime.

Acknowledgements

The research presented in this paper has been partially funded by the German Bundesministerium für Bildung, Wissenschaft, Forschung und Technologie (BMBF) under grant number 01IN802 (project "GETESS´´).

References

1. L. Ahrenberg, N. Dahlbäck, A. Jönnson, and A. Thure. Customizing interaction for natural language interfaces. *Computer and Information Science*, 1(1), 1996.
2. J.L. Austin. *How to Do Things with Words*. Oxford University Press, 1962.
3. J.P. Ballerini, M. Büchel, D. Knaus, B. Mateev, M. Mittendorf, P. Schäuble, P. Sheridan, and M. Wechsler. SPIDER retrieval system at TREC 5. In *Proc. of TREC-5*, 1996. http://www-nlpir.nist.gov/pubs.
4. C.M. Bowman, P.B. Danzig, R.D. Hardy, U. Manber, and M.F. Schwartz. The harvest information discovery and access systems. *Networks and ISDN Systems*, 1995. http://ftp.cs.colorado.edu/pub/cs/techreports/schwartz/Harvest.Conf.ps.Z.
5. P. Buitelaar, K. Netter, and F. Xu. Integrating different strategies for cross-language information retrieval in the MIETTA project. In D. Hiemstra, F. de Jong, and K. Netter, editors, *Language Technology in Multimedia Information Retrieval. Proc. of the 14th Twente Workshop on Language Technology*, TWLT 14, pages 9–17. Universiteit Twente, Enschede, 1998.
6. J. Capstick, A.K. Diagne, G. Erbach, H. Uszkoreit, F. Cagno, G. Gadaleta, J.A. Hernandez, R. Korte, A. Leisenberg, M. Leisenberg, and O. Christ. MULINEX: Multilingual web search and navigation. In *Proc. of Natural Language Processing and Industrial Applicatons*, 1998.
7. S. Decker, M. Erdmann, D. Fensel, and R. Studer. Ontobroker: Ontology based access to distributed and semi-structured information. In R. Meersman et al, editor, *Database Semantics, Semantic Issues in Multimedia Systems*, pages 351–369. Kluwer, Boston, MA, 1999.
8. H. Eriksson, Y. Shahar, S. W. Tu, A. R. Puerta, and Mark A. Musen. Task modeling with reusable problem-solving methods. *Artificial Intelligence*, 79(2):293–326, 1995.
9. D. Fensel, S. Decker, M. Erdmann, and R. Studer. Ontobroker: The very high idea. In *Proceedings of the 11^{th} International Flairs Conference*, 1998.
10. B. Grosz, D. Appelt, P. Martin, and F. Pereira. Team: An experiment in the design of transportable natural-language interfaces. *Artificial Intelligence*, 32(2):173–243, 1987.
11. U. Hahn, M. Romacker, and S. Schulz. How knowledge drives understanding: Matching medical ontologies with the needs of medical language processing. *AI in Medicine*, 15(1):25–51, 1999.
12. Andreas Heuer and Denny Priebe. IRQL — Yet another language for querying semistructured data? Preprint CS–01–99, Universität Rostock, Fachbereich Informatik, 1999.
13. A. Heyer, H. Meyer, A. Düsterhöft, and U. Langer. SWING: Der Anfrage- und Suchdienst des Regionalen Informationssystems MV-Info. In *Tagungsband IuK-Tage Mecklenburg-Vorpommern. Schwerin, 27./28. Juni 1997*, 1997.
14. J. Lamping and R. Rao. The hyperbolic browser: A focus + context technique for visualizing large hierarchies. *Journal of Visual Languages & Computing*, 7, 1996.
15. D. McGuinness and P. Patel-Schneider. Usability issues in knowledge representation systems. In *Proc. of AAAI-98*, pages 608–614, 1998.
16. G. Neumann, R. Backofen, J. Baur, M. Becker, and C. Braun. An information extraction core system for real world german text processing. In *5th International Conference of Applied Natural Language Processing*, pages 208–215, Washington, USA, March 1997.
17. G. Neumann and G. Mazzini. Domain-adaptive information extraction. Technical report, DFKI, Saarbrücken, November 1998.