SemSearch 2008, CEUR Workshop Proceedings, ISSN 1613-0073, online at CEUR-WS.org/Vol-334/

Exploring the knowledge in Semi Structured Data Sets with Rich Queries

Jürgen Umbrich and Sebastian Blohm

Institut AIFB, Universität Karlsruhe(TH), D-76128 Karlsruhe, Germany
{juum, blohm}@aifb.uni-karlsruhe.de

Abstract. Semantics can be integrated in to search processing during both document analysis and querying stages. We describe a system that incorporates both, semantic annotations of Wikipedia articles into the search process and allows for rich annotation search, enabling users to formulate queries based on their knowledge about how entities relate to one another while simultaneously retaining the freedom of free text search where appropriate. The outcome of this work is an application consisting of semantic annotators, an extended search engine and an interactive user interface.

1 Introduction

Currently, there is a vast amount of data available on the Web, mostly encoded in unstructured formats, such as plain text or HTML pages. Users are investing a substantial amount of effort in an attempt to organise and structure the unstructured information within their respective knowledge bases. A classic way for managing information in a semi structured way is the use of encyclopedias. Encyclopedias are compendiums containing information about branches of knowledge. Depending on the scope of the encyclopedia, each knowledge branches try to capture the information of a particular knowledge field or of a group, like of a community, of a nation or ideally of the whole mankind.

In the online encyclopedia Wikipedia[1], articles are organised as follows:

- **Article titles cover the subject**
  Each subject in the encyclopedia is covered by one article and is identifiable by the article title. Usually, these articles can be accessed by the list of the article titles, which are ordered in alphabetical manner.
- **Articles belong to categories**
  Articles can also belong to one or more categories, which pre-existing or created by the author manually. Encyclopedia users can access the knowledge base by exploring the articles within a category.
- **Article can link to other articles**
  If articles refer to other articles or subjects the author can express this relationship via a link. By reading articles contained within Wikipedia, the users can navigate to other articles following the links.

[1] http://www.wikipedia.org/

While this provides a good structure for manual browsing, it does not directly facilitate search nor does it support machine-understandable information about document content. At this stage, we would like to introduce a specific user scenario to motivate our study.

A user wants to find famous scientists born in Germany, specifically scientists which have received a degree at the University of Karlsruhe.

In our example user scenario, the user knows that he is looking only for information about persons, especially about scientists, beyond this, the user knows that the entities in his query are related to each other; the wanted scientists are *born in* in Germany, and have *received a* degree at a University in the city Karlsruhe. When entering a keyword-based query, most of this information is not conveyed and thus cannot be used to exploit Wikipedia's rich structure to increase the retrieval quality.

Depending on the structure and on the search possibilities of the encyclopedia the users are not able to use their background information to explore the knowledge base. Usually, the users have two ways to get access to the knowledge: 1) They can use a keyword -based search interface and 2) browse and navigate through the data set by articles, links or by categories. The problem, with the keyword-based search interfaces is, that people can neither express the meaning of words nor the relation between words, nor can they specify the category of the search results. For data sets of a manageable size, browsing might be a good way to explore the knowledge, but for huge knowledge repositories browsing can result in a very time consuming task, one which is not guaranteed to find the required results. Another disadvantage of browsing through categories and articles is that users have to inspect each article to decide if the subject matter presented is a suitable answer to the query.

Our approach is based on the idea to extract the implicit knowledge encoded in the category system and furthermore make the knowledge explicitly searchable via the annotation of articles, thus enables structured query functionalities over the knowledge base. With new query options, users can express the meaning of words with annotations. Moreover, they can describe and model relationships between entities through the combination of both annotation- and free text search. Thus, users are able to apply their background knowledge about the search term and the expected results to ask more specific queries and receive higher quality results.

We will show how to improve search functionalities for semi structured information sources by using annotation search combined with rich query functionalities. Therefore, we use the online encyclopedia Wikipedia and annotate the articles with meta information encoded in Wikipedia's category structure and with information from a external knowledge base. More specifically, we exploit Wikipedia's category and link structure for capturing semantics in keyword-base search. Pages are annotated with Wikipedia's category information, which is semantically grounded by using the Yago ontology[1]. References to other en-

tities within a document for which Wikipedia holds further information are also annotated with categories and Yago concepts.

The remainder of the paper is organised as follows: In Section 3 we give an overview about Semantic Search across annotated text. Section 4 describes our architecture and the functionality of each individual component. In Section 5 we introduce the semantic query syntax used by the components of the architecture. In Section 6 we present a user-interface, that hides the syntactic complexity of the extended notion of queries from the end-user. Finally, in Section 7 we conclude with an outlook of an approach for supporting end-users in creating structured queries.

2 Related Work

A simple but appealing definition of Semantic Search has been given by Soumen Chakrabarti [2] which states that queries "must enable schema-free searches but reward schema knowledge". Schema knowledge thereby can be integrated in various positions in the Semantic Search process. Generally speaking, information retrieval processing consists of an *indexing time (offline)* phase and a *query time* phase. At indexing time, documents are collected and pre-processed. This includes data normalisation, identification of relevant content (e.g. text tokens) and may include higher level processing like the extraction of relevant metadata and information as well as deriving a semantic document representation. At query time, the user query is taken to construct a query that is interpretable by the query processor. The output may be a (weighted) Boolean or bag-of-words query, a SPARQL expression or a request otherwise formalised according to the requirements of the query processor. After triggering query processing, the results are ranked and presented to the user. A feedback processing component may then allow a user to refine the request and re-trigger the query time process.

The vision of Semantic Search has inspired work in various directions. Different parts of the information retrieval process have been augmented with semantic information. We discuss briefly several Semantic Search systems which have in common that users may be unaware of (parts of) the ontology and that query language enables schema-free searches but allows improving retrieval when further knowledge is incorporated.

Guha et al. [3] introduce Semantic Search as the idea of using information from the Semantic Web for search. Applications are presented that add to classical search results with search results from RDF knowledge bases traversed using graph search, into classical search results. Thus, the semantics are captured in an additional query processor and then integrated during result presentation. A mechanism to capture rich ontological structure during query construction for parallel query processing has been presented by Tran et al. [4].

In the field of XML retrieval, methods from text search are being integrated into the structured XML retrieval paradigm. The XXL-engine [5] allows for the retrieval of objects, which have similar semantic names as the search term

given. The similarity operator is defined using semantic distance measures in an ontology graph. This work thus approaches Semantic Search from the side of structured retrieval.

Bonino et al. [6] transfer the standard term-index-based retrieval to an ontology-based paradigm by mapping the term by term to concepts in an ontology and then applying tf-idf like similarity search on "conceptual vectors". The query is also mapped to such conceptual vectors with the help of query-refinement techniques.

Chakrabarti [7] presents a search system that operates on both, the plain corpus and annotations. While annotations are defined as (probabilistic) connections to one or more ontologies and queries may involve ontology elements as well as uninterpreted strings. Among those discussed here, this work is closest to ours as it works on a shallowly annotated corpus relying on an extended standard information retrieval index.

The RelSE system[2] uses the category information in Wikipedia to allow precise search. Keyword search is made possible on a set of pages restricted by selecting Wikipedia categories. Our work extends this principle by indexing not only articles with their categories but also providing category informations for words mentioned within the pages.

Semantic content for Wikipedia has been derived in various ways and for various purposes. We employ the Yago ontology[1] which connects the Wikipedia category system to the WordNet lexical taxonomy and thereby creates an ontology with a large coverage within Wikipedia. The DBPedia project [8] provides relational information as captured in the Wikipedia Infoboxes. Information Extraction techniques can further extend Wikipedia annotations [9, 10].

3 Semantic Search on Annotated Text

Assuming that added value of Semantic Search is "rewarding schema knowledge" and based on our review of related work, one can observe that different approaches to Semantic Search differ in the amount and type of knowledge that is integrated as well as where and how the knowledge comes into play. Semantic technologies in an Information Retrieval context can be applied in two ways:

- **Interpretation of the query:** Allow the system/user to relate the contents of the query to formalised concepts and relations.
- **Interpretation of the content:** Allow the system/user to relate the contents of the documents to formalised concepts and relations.

A semantically enabled retrieval system can employ either of them or both. These aspects therefore constitute two key dimensions in which Semantic Search systems can differ. Classical text search systems (no interpretation of query no interpretation of content) and fully formalised knowledge bases with formal query language (full semantic access to both content and query) form the corners of the

[2] http://relse.apexlab.org/

space spanned by these dimensions. As opposed to mere KB lookups semantic search does not operate on any kind of formalised knowledge but on structured information derived from text. Due to the imperfection and incompleteness of this derivation, the text itself cannot be discarded during search.

Annotating semantic information within text is a challenging and error-prone task. Search systems must be prepared to handle a large amount of queries from various domains from users with completely different information needs. Information Extraction tools are either focused on a limited domain or can handle only a generic set of semantic concepts or properties Furthermore, the cost of engineering Language Resources either a Machine Learning or Knowledge Based Approach is time-consuming and expensive, requiring either specialist knowledge or large volumes of quality training data, which may be difficult to obtain. Our approach attempts to leverage pre-existing metadata into the IE process to aid retrieval thus viewing Wikipedia as a preannotated semantic corpus to be exploited. We thus build our present work on what we call the *Annotation/Query trade-off* hypothesis: A lack of fine-grained annotation can be compensated by incorporating more knowledge in the querying process and conversely, richer annotations allow semantic retrieval with less effort on the side of the user.

In this study, we produce annotations based on knowledge that is present in the Wikipedia. This knowledge consist of conceptual annotations for articles and words mentioned in the articles. Yet, we allow the user to query for relational information by providing a query mechanism that allows him to formulate his knowledge of the relations (e.g. domain and range) into the query.

4 Setup

In this section we describe our architecture and present each individual component. Figure 1 gives an overview of our system, consisting of the following five components (the dotted components are future work):

- A **semi structured data set** serves documents as input for the annotation engine.
- Various text analsys engines within the **annotation engine** parse the documents and extract the implicit knowledge and simultaneously anchoring it to the document.
- The **search engine** stores and index the document and the anchored annotation set and enables access to the indexed data via keyword, annotation and structured queries.
- The **pattern extraction** module will use the information from the annotation set and the document content to extract relation patterns between entities, like *born_in*, *studied_at*, *capital_of*. It also will supports the query interface with these extracted patterns. The extracted patterns are used to support the query creation process and can give recommendation for the most frequent relation patterns between certain entities. (*The implementation of this component is future work*)

– With the **query interface** users can create and query the index of the search engine in a user friendly way.

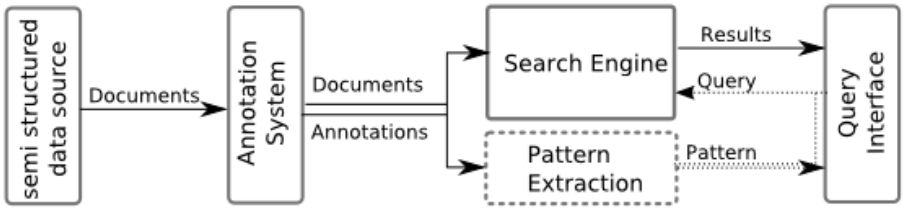

Fig. 1. Overview of the architecture

For the five components of the architecture we choose the following setup:

4.1 Data Source: English Wikipedia

We use a dump of the English Wikipdia from December 17th 2006 containing 1.6 million articles. Wikipedia is a multilingual, web-based, free content encyclopedia project and the biggest collaboratively edited knowledge source on the internet. More than 75.000 contributors have published over nine million articles in around 250 languages. Furthermore the knowledge is not restricted to a particular domain, the Wikipedia data set contains articles about a various of different domains and topics. Our motivation to use this data set was, that Wikipedia articles provide a lot of meta-information like the Wikipedia-categories or links to other articles.

4.2 Annotation Engine : Apache UIMA Framework and Text Analytic engines

The annotation engine processes documents and annotates new discovered knowledge to the documents. These information can be obtained directly from the content of the document or can be added from external meta data sources. Various text analysis engines, like word and sentence tokenizer, named entity recogniser or part of speech taggers, can be developed and plugged in the processing pipeline of a annotation engine. We use the Unstructured Information Management Architecture (UIMA)[3] to drive the annotation engine. UIMA provides from scratch some useful text analysis engines like a tokeniser, sentence and paragraph splitter, moreover, the API allows annotator developers to focus on writing the annotation logic (in Java) while a common data structure and a workflow engine are provided. Hence, we developed some text analysis engines expose the implicit knowledge in Wikipedia articles and annotate the articles

[3] http://incubator.apache.org/uima/

with additional information derived from the structure. We extract the following knowledge from the Wikipedia articles and pass them to the search engine together with meta information from the Yago knowledge base.

- **Discovered knowledge from the document corpus**
  Wikipedia articles contain already meta information about the covered subject and about other relevant articles or subjects, encoded in the article categories and in links. We annotate explicit the title and the Wikipedia categories of a document. Also, we parse the document structure for other occurrences of the title string to obtain more information of the page. For outgoing links to other related articles we anchor the link title and the categories of the targeted article to the hyperlinks in the input document. Furthermore, we identify year, month and day information from various date formats and annotate the original document by this date information.
- **Additional added knowledge from the Yago knowledge base**
  Beside extracting knowledge from the document itself, we annotate the articles with additional information derived from the Yago ontology, a huge semantic knowledge base, containing the unification of Wikipdia and WordNet[4] and knows around 14m facts about entities(e.g. person, city, organisation). The Yago data contains for each Wikipedia category a hierarchy of abstract concepts, e.g. the category `american_tennis_player` has the following hierarchy: `american_tennis_player < player < person < causal_agent`. For each Wikipedia category discovered in an article, resulting in link and page categories, we attach the corresponding Yago category and its hierarchical ancestor categories.

Figure 2 shows for parts of the Wikipedia article of Robert Cailliau, one of the inventors of the WWW, and what kind of information we extract and annotate for each article.

4.3 Index-based Search Engine

The search engine stores and indexes the document content and output from the annotation engine, further it offers search, browsing and navigation functionalities over the indexed data. We use the enterprise search platform of IBM, OmniFind, as the search engine in our architecture. OmniFind provides a UIMA compliant processing engine and offers beside the keyword search over document content also search capabilities for the annotate knowledge from the UIMA annotation engine. With the semantic query syntax of OmniFind the users can create structured queries and can efficiently exploit the indexed knowledge.

4.4 Pattern Extraction

With the new discovered knowledge from the annotation engine we can extract the word patterns between two annotated entities. The idea is to obtain information about the relationship between two entities from the tokens of these

[4] http://wordnet.princeton.edu/

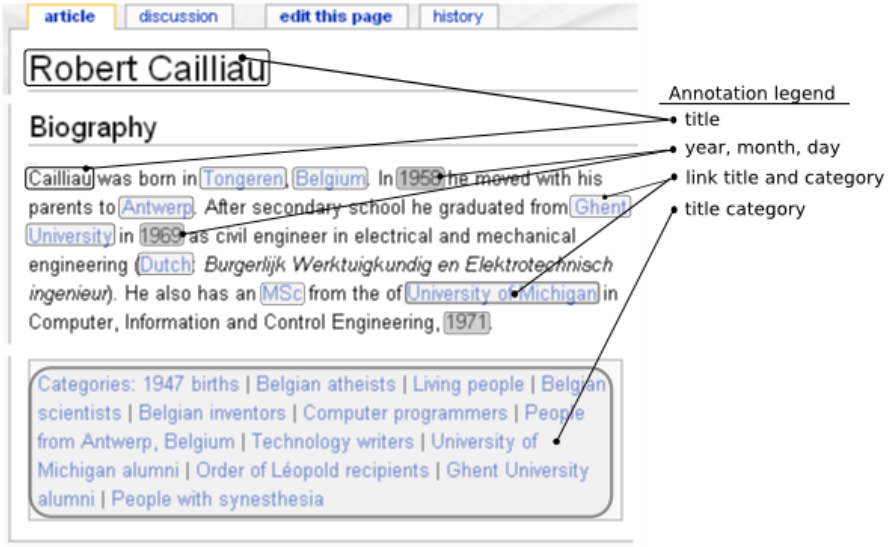

Fig. 2. Annotations for a Wikipedia article.

patterns. But, exposing knowledge about the relation between two entities is not a trivial task and will be addressed in future work.

4.5 Query Interface

The query interface use the search and index API (SIAPI) of the OmniFind search engine to get access to the knowledge base and to execute structured queries. The user friendly query interface, written in Java using Swing components, is described in detail in Section 6.

5 Structured Queries

5.1 OmniFind Query-Interface

OmniFind's search interface supports the same standard query operators as most common search engines, like free text and word phrase search as well as search operators like `AND`, `OR`, `NOT` and `WILDCARDS`. In addition, it provides two functionally equivalent types of query syntax, *XML fragments* and a subset of *XPath*. We use the *XML fragment* syntax in our work.

5.2 Structured Queries

XML Fragments provides a wide variety of additional query functionalities[11]. An XML Fragments query consists of an underspecified XML structure and thus

combines keyword queries with queries for annotated information. This enables search for more specified concepts, like searching for person's names. With the help of their domain knowledge, users can express relationships between object like "'the person and the city must occur in the same sentence"' or more specific like "'a persons *lives in* the city"' or "'a person *died in* a city"'.

The following query shows the OmniFind XML fragment semantic query syntax and how free text search and annotation search can be combined.

```
@xmlf2::'
  <page category="scientist" /> OR <page category="person" />
 +<sentence>
    </title> * "born in" * <link category="country">Germany</link>
  </sentence>
 +<sentence>
    </title> * "studied at" * <link category="University"/>
             * <link category="city">Karlsruhe</link>
             * "received" * <link category="Degree"/>
  </sentence>
```

The first line restricts the results to articles about persons or scientists. Next, we describe some further restrictions to the results. The sought after person and different entities have to occur in the same sentences and between these annotated entities certain word sequences have to show up. Searching for annotations combined with keywords and several query operators, like wildcards, is one way to express the relationships between entities.

6 End-User Query Interface

The end user query interface allows users to access and search the data indexed by the search engine. The whole query interface is a stand-alone software, written in JAVA 1.5 and adapted especially for the annotations from our annotation engine. The main focus of the end-user query interface is to allow users to create complex queries in a user friendly and understandable way. The queries are converted into the OmniFind query syntax and executed using the search and Index API (SIAPI) of OmniFind. The standard query interface, provided by OmniFind, can only process semantic queries encoded in the OmniFind query syntax. As can be seen from the example query in the previous section, the query syntax can be hard to use and understand for the users. There are two query creation modes available, a very simple version of creating the queries and a advance version, that allows to create complex structured queries.

6.1 Simple Query Interface

The simple query creation interface, enables basic query functionalities. As Figure 3 shows, the simple query interface uses common query concepts like, text

fields or drop-down menus. The same concept can be found in other well-known search interfaces, like by Ebay[5] or Amazon[6]. People can use the *"keyword search"* field for very simple keyword queries, optional with operators like AND, OR, NOT, WILDCARDS and word phrases. In the *"Page Title"* query field people specify the search for page titles or word snippets in page title. Using the *"Page Category"* query field, people can filter the result set for pages of a special page category. Below this is the *"Result Pages Containing"* query field, where people can search for labels or categories of the outgoing links. The search functionalities of this query interface go far beyond the search functionalities offered by the original encyclopedia page.

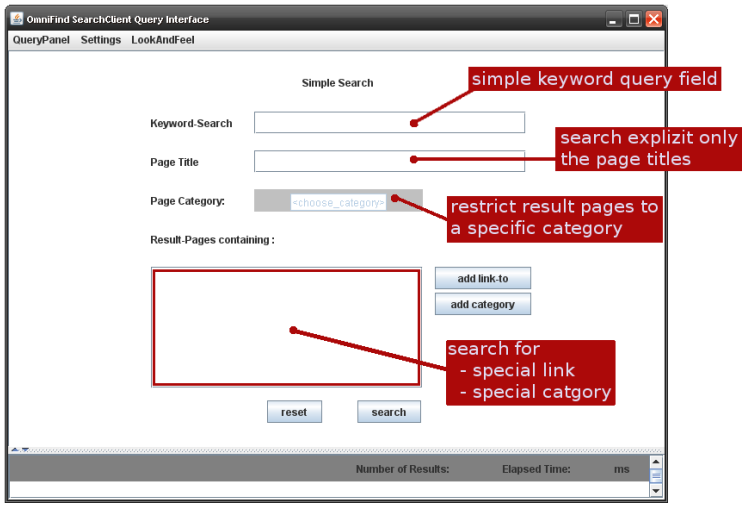

Fig. 3. Simple Search Interface

6.2 Advanced Query Interface

Users, more familiar with the query interface or structured queries, can use the advanced query interface to create complex queries and combine keyword and annotation search. Figure 4 shows the advanced query interface with its core component, the query creation module.

A query is a combination of various query patterns with following query operators:

- keywords, word phrases and keyword query operators.
- queries for links and their Wikipedia or Yago category.

[5] http://www.ebay.com/
[6] http://www.amazon.com/

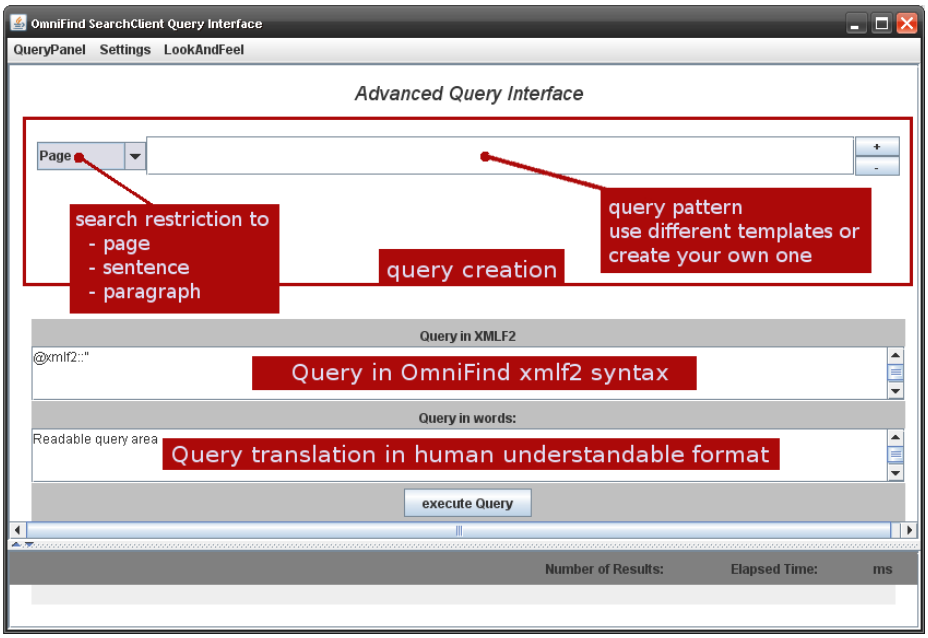

Fig. 4. Advanced Query Interface

– queries for page titles and page categories, as Wikipedia categories and Yago categories.

Furthermore, the user can specified, where the query pattern has to occur in the document, either looking for a match in the whole document, or in a subset of the document content, like a paragraph or a sentence. The query interface offers additionally the translation of the query patterns into the OmniFind query syntax and into a more human readable representation. Figure 5 shows the advanced query interface with out running example query.

6.3 Resultset and Ranking

Generally speaking, the approach allows using the full ranking and result presentation capabilities of the employed search engine. The results returned by the API contain the document URI, title of a summary or short description of the textual content. We show the title and the hyperlink to the Wikipedia article in our result panel of the user interface. Highlighting the semantic annotations is easily possible. Users can then open the documents in a separate browser window. If the indexed collection contains Web documents, like in our case, the ranking of the results also contains link analysis, based on the in-link counts of Omnifind's crawler.

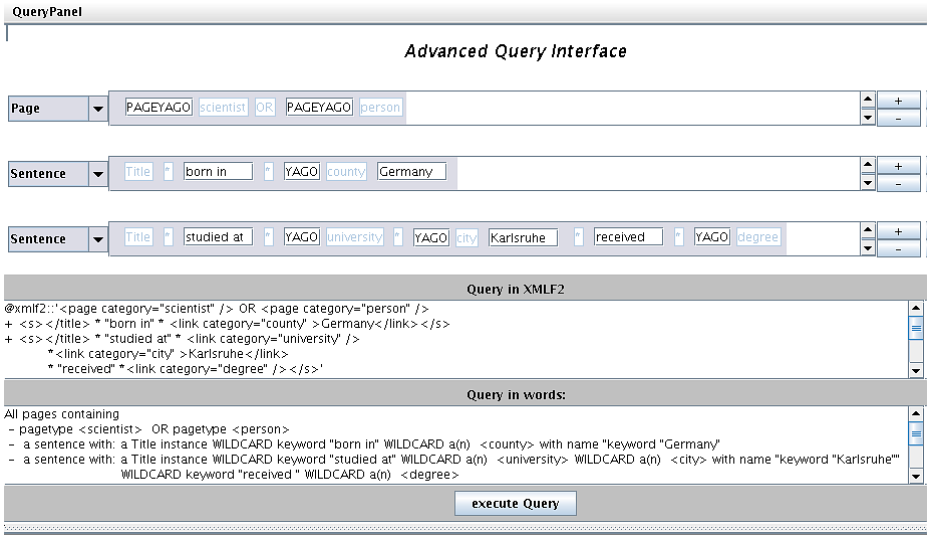

Fig. 5. Advanced Query Interface shows the running example

7 Future Work

Future work includes the implementation of the pattern extraction module and the investigation of discovering information about the relation between two entities. Therefore, we will extract the word tokens between two annotated entities and try to expose relation patterns out of them. With these patterns we can model the semantic relation between entities, beyond this, we can support the query creation task for the end-users by recommending relationships between two entities. The users do not have to know what kind of keyphrases are used in the knowledge base to describe relation, e.g the word tokens *"studied at"*, *"was a student at"* and *"completed a degree at"* are describing the relation between a person and a university. The semantic patterns and their corresponding word tokens can help the end users to model relations between entities in their query, without knowing what keyphrases describe these relations in the knowledge base. For example, in our user scenario, we know that the person we are looking for received a degree at the University in Karlsruhe, but we do not know the exact keyphrases between these entities. A semantic relation pattern for this scenario can be *entity:*`[PERSON]` *relation:*`received` *entity:*`[DEGREE]`.

8 Conclusion

We presented an architecture to exploit Wikipedia's category and link structure for capturing semantics in keyword-base search. Pages are annotated with Wikipedia's category information which is semantically grounded by using the Yago ontology. References to other entities within a document for which Wikipedia

holds further information are also annotated with categories and Yago concepts. This allows extended structured queries which can be posed through a dedicated search interface. Future work contains to use the extracted relation patterns to help users in creating their queries and the corresponding results.

Acknowledgements

This work has been supported by MFG Stiftung, Baden-Würrtemberg and by the X-Media project (www.x-media-project.org) sponsored by the European Commission as part of the Information Society Technologies (IST) program under EC grant number IST-FP6- 026978.

References

1. Suchanek, F.M., Kasneci, G., Weikum, G.: Yago: A core of semantic knowledge. In: Proceedings of the 16th International Conference on World Wide Web (WWW), ACM Press (2007) 697 – 706
2. Chakrabarti, S.: Building blocks for semantic search engines: Ranking and compact indexing in entity-relation graphs. Keynote talk at the International Workshop on Intelligent Information Access (IIIA-2006) (2006)
3. Guha, R., McCool, R., Miller, E.: Semantic search. In: WWW '03: Proceedings of the 12th international conference on World Wide Web, New York, NY, USA, ACM Press (2003) 700–709
4. Tran, T., Cimiano, P., Rudolph, S., Studer, R.: Ontology-based interpretation of keywords for semantic search. In: Proceedings of the 6th 6th International Semantic Web Conference, Busan, Korea (2007) 523–536
5. Schenkel, R., Theobald, A., Weikum, G.: Semantic similarity search on semistructured data with the xxl search engine. Information Retrieval **8**(4) (2005) 521–545
6. Bonino, D., Corno, F., Farinetti, L., Bosca, A.: Ontology driven semantic search. SIGIR Forum **1**(6) (2004) 1597–1605
7. Chakrabarti, S., Puniyani, K., Das, S.: Optimizing scoring functions and indexes for proximity search in type-annotated corpora. In: WWW '06: Proceedings of the 15th international conference on World Wide Web, New York, NY, USA, ACM Press (2006) 717–726
8. Auer, S., Bizer, C., Lehmann, J., Kobilarov, G., Cyganiak, R., Ives, Z.: Dbpedia: A nucleus for a web of open data. In: In: Proceedings of ISWC 2007. (2007)
9. Ruiz-Casado, M., Alfonseca, E., Castells, P.: Automatic extraction of semantic relationships for wordnet by means of pattern learning from wikipedia. In: Natural Language Processing and Information Systems. Springer, Berlin / Heidelberg (May 2005)
10. Blohm, S., Cimiano, P.: Using the web to reduce data sparseness in pattern-based information extraction. In: Proceedings of the 11th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD), Warsaw, Poland, Springer (SEP 2007) 18–29
11. Hampp, T., Lang, A.: Semantic search in websphere information integrator omnifind edition: The case for semantic search. IBM Developer Works (2005)