Semantic Full-text Search with Broccoli

Hannah Bast, Florian Bäurle, Björn Buchhold, Elmar Haußmann
Department of Computer Science
University of Freiburg
79110 Freiburg, Germany
{bast, baeurlef, buchhold, haussmann}@informatik.uni-freiburg.de

ABSTRACT

We combine search in triple stores with full-text search into what we call *semantic full-text search*. We provide a fully functional web application that allows the incremental construction of complex queries on the English Wikipedia combined with the facts from Freebase. The user is guided by context-sensitive suggestions of matching words, instances, classes, and relations after each keystroke. We also provide a powerful API, which may be used for research tasks or as a back end, e.g., for a question answering system. Our web application and public API are available under `http://broccoli.cs.uni-freiburg.de`.

1. INTRODUCTION

Knowledge is available in electronic form in two main representations: as natural language text (e.g., Wikipedia), and in structured form (e.g., Freebase). The central motivation behind our system is that both representations have their advantages and should be combined for high-quality semantic search.[1]

For example, consider the query *Plants with edible leaves and rich in vitamin C*. Information about which plant contains how much vitamin C is naturally represented as fact triples. Indeed, this information is found in a knowledge base like Freebase. Information about which plants have edible leaves is more likely to be mentioned in natural language text. It is mentioned many times in Wikipedia, but we don't find it in Freebase (or any other knowledge base that we know of). In principle, the information could be added, but there will always be specific or recent information only described as text.

In the following, we describe how we combine these two information sources in a deep way. Figure 1 shows a screenshot of our demo system in action for our example query.

2. SYSTEM OVERVIEW

Preprocessing In principle, our system works for any given text corpus and ontology. For our demo we use the English Wikipedia (text) + Freebase (ontology). We preprocess this data in three phases. First, we link entities from the ontology to mentions in the full text, utilizing Wikipedia links and a set of heuristics as described in [1]. This provides the basis for our *occurs-with* operator explained below. Second, the full text is split into *contexts* that ”semantically belong together” as described in [3]. This is key for results of high quality. Third, the special-purpose index described in [2] is built. This is key for providing results and suggestions in real time.

Queries The user interface allows to incrementally construct basic tree-like SPARQL queries, extended by an additional relation *occurs-with*. This relation allows to specify the co-occurrence of entities from the ontology with words from the text. For our example query, the back end computes all occurrences of plants that occur in the same context (see above) as the words *edible* and *leaves*. We also provide the special relation *has-occurrence-of*, to search for documents in which words and entities co-occur. Regular full-text search is thus included as a special case.

Query Suggestions Based on the input of a user, our system gives context-sensitive suggestions for words, classes, instances, and relations. The displayed suggestions always lead to hits, and the more / higher-scored hits they lead to, the higher they are ranked. This is an elementary feature for any system that utilizes a very large ontology. Without good suggestions it is very hard to guess how entities and relations are named, even for expert users.

Excerpts For each result hit (an entity or a document), matching evidence for each part of the query is provided. This is invaluable for the user to judge whether a hit indeed matches the query intent. The UI also provides (on click) detailed information about the NLP preprocessing (see above).

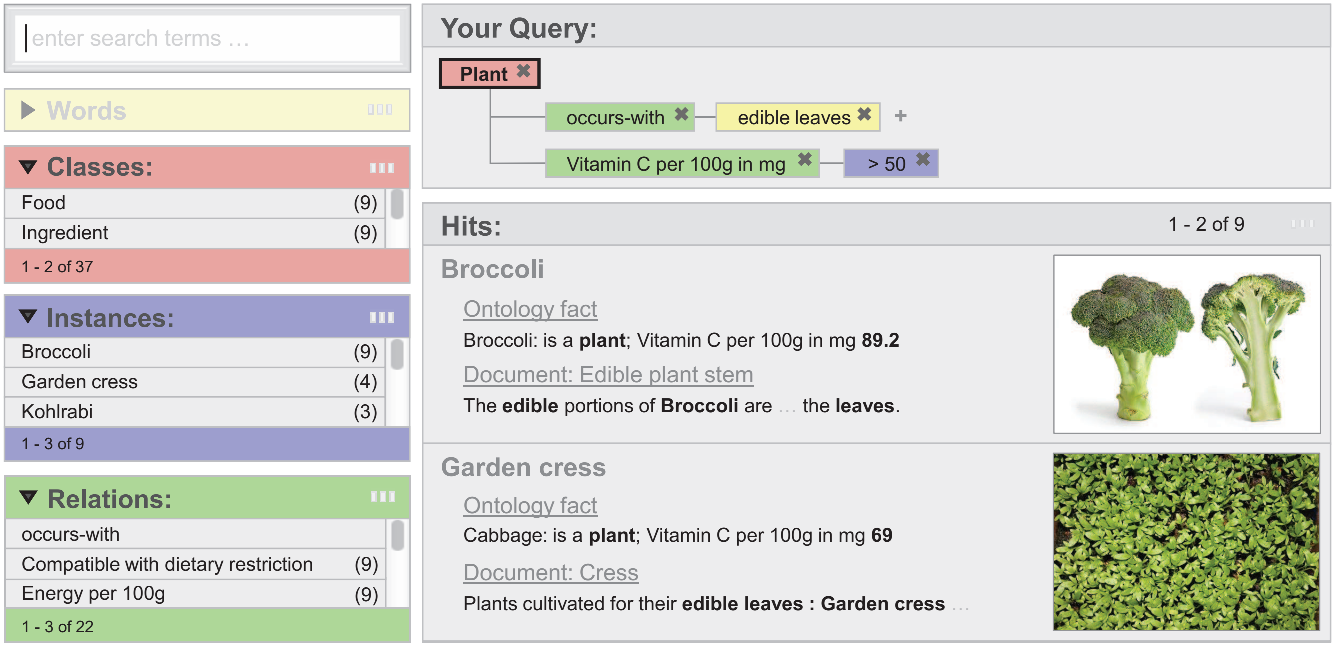
3. TARGET USERS

We see two uses of our system, and hence also two target groups of users.

Our first target group is expert searchers. Our search interface is more complex than ordinary keyword search or asking questions in natural language. The benefit is a powerful query language with precise result semantics. Under these constraints, we have made the query process as easy-to-use as possible. For example, there are tens of thousands of Wikipedia List pages like ”Plants with edible leaves”.

[1] As a matter of fact, Wikipedia also contains some structured data, and Freebase also contains natural language text.

SIGIR'14, July 6–11, 2014, Gold Coast, Queensland, Australia.
ACM 978-1-4503-2257-7/14/07.
http://dx.doi.org/10.1145/2600428.2611186.



Food	(9)
Ingredient	(9)
1 - 2 of 37	

Broccoli	(9)
Garden cress	(4)
Kohlrabi	(3)
1 - 3 of 9	

occurs-with	
Compatible with dietary restriction	(9)
Energy per 100g	(9)
1 - 3 of 22	

Broccoli

Ontology fact

Broccoli: is a **plant**; Vitamin C per 100g in mg **89.2**

Document: Edible plant stem

The **edible** portions of **Broccoli** are … the **leaves**.

Garden cress

Ontology fact

Cabbage: is a **plant**; Vitamin C per 100g in mg **69**

Document: Cress

Plants cultivated for their **edible leaves : Garden cress** …

Figure 1: A screenshot of our demo system. The current query is visualized on the top right as a tree. Below, the result hits are shown, grouped by instance (entity) and ranked by relevance, together with evidence from the ontology and the full text. The query can be extended further with the search field on the top left. The four boxes below provide context-sensitive suggestions that depend on the current focus in the query, here: matching sub and super classes, instances, and relations for plants matching the current query.

Many of these are actually non-trivial semantic queries, which are hard to answer with traditional tools, like Google web search. We expect our tool to be a great asset for contributors to such List pages. We expect a similar benefit for expert searches in other areas, e.g., news (presidential campaign backers) or medicine (symptoms of a disease).

Our second target group is researchers in semantic search or engineers of such systems. They may want to use our system to explore the given data and thus gain insight into which facts are expressed in which ways. Engineers may also use our API as a back end for a more simplistic front end, suited for non-expert users. As a first step towards such a front end, we have integrated the following feature in our demo: when typing a query with three or more words without following any of the suggestions, the system tries to convert these keywords into a matching structured query. For example, try *mafia films directed by francis coppola*.

4. RELATED WORK

We see three lines of research closely related to our system.

First, we already mentioned systems for semantic search with more elaborate front ends. In particular, such allowing natural language queries like IBM's well-known Watson [4], or standard keyword queries like in ad-hoc entity search [5]. When they work, such more intuitive front ends are clearly to be preferred. However, semantic search is complex and hard, and queries often fail. Then simple front ends lack the feedback needed to understand what went wrong and what can be done to ask a better query.

Second, there are various extensions of ontology search by a free-text component. A good example is the MQL language (similar to the more standard SPARQL) provided by Freebase (`http://www.freebase.com/query`). In MQL, objects of triples can also be string literals and these can be matched against regular expressions and keyword queries. For example, find all songs containing the words *love* and *you* in their title. In principle, this could be used to simulate our *occurs-with* operator, but only very inefficiently; see [2, Section 4 and Table 1].

Third, information extraction (IE) aims at extracting factual knowledge from text. If this succeeded perfectly, ontology search would be all we need. There are two caveats, however. Whatever information was not extracted properly is lost. In our system, all the original information is kept and is, in principle, accessible by an appropriate query. Also, IE triples often have string literals as objects. Dealing efficiently with these requires a special index data structure, like the one behind our search.

5. REFERENCES

[1] H. Bast, F. Bäurle, B. Buchhold, and E. Haussmann. Broccoli: Semantic full-text search at your fingertips. *CoRR*, abs/1207.2615, 2012.

[2] H. Bast and B. Buchhold. An index for efficient semantic full-text search. In *CIKM*, pages 369–378, 2013.

[3] H. Bast and E. Haussmann. Open information extraction via contextual sentence decomposition. In *ICSC*, pages 154–159, 2013.

[4] D. A. Ferrucci, E. W. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. Kalyanpur, A. Lally, J. W. Murdock, E. Nyberg, J. M. Prager, N. Schlaefer, and C. A. Welty. Building watson: An overview of the DeepQA project. *AI Magazine*, 31(3):59–79, 2010.

[5] J. Pound, P. Mika, and H. Zaragoza. Ad-hoc object retrieval in the web of data. In *WWW*, pages 771–780, 2010.