A Case for Semantic Full-Text Search

(position paper)

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

ABSTRACT

We discuss the advantages and shortcomings of full-text search on the one hand and search in ontologies / triple stores on the other hand. We argue that both techniques have an important quality missing from the other. We advocate a deep integration of the two, and describe the associated requirements and challenges.

1. FULL-TEXT SEARCH

The basic principle of full-text search is that the user enters a (typically small) set of keywords, and the search engine returns a list of documents, in which some or all of these keywords (or variations of them like spelling variants or synonyms) occur. The results are ranked by how *prominent* these occurrences are (term frequency, occurrence in title, relative proximity, absolute importance of the document, etc.)

1.1 Document-oriented queries

This works well as long as (i) the given keywords or variants of them occur in enough of the relevant documents, and (ii) the mentioned prominence of these occurrences is highest for the most relevant documents. For example, a Google query for *broccoli* will return the Wikipedia page as the first hit, because it's a popular page containing the query word in the URL. A query for *broccoli gardening* will also work, because relevant documents will contain both of the words, most likely in a title / heading and in close proximity.

For large document collections (as in web search), the number of matching documents is usually beyond what a human can read. Then *precision* is of primary concern for such queries, not recall. The informational Wikipedia page (or a similar page) should come first, not second or fourth. And it is not important that we find *all* broccoli gardening tips on the internet.

Bottom line: *Full-text queries work well when relevant documents contain the keywords or simple variations of them in a prominent way. The primary concern is precision, not recall.*

JIWES '12 August 12 2012, Portland, OR, USA

1.2 Entity-oriented queries

Consider the query *plants with edible leaves*. As we explain now, this kind of query is inherently problematic for full-text search engines.[1]

The first problem is as follows. Relevant documents are likely to contain the words *edible leaves* or variations of them (see above). But there is no reason why they should contain the word *plants*, or variations of it like *plant* or *botany*. Rather, they will contain the name of a particular plant, for example, *broccoli*. This is exactly the kind of knowledge contained in *ontologies*, discussed in Section 2.

The second problem is that the sought-for results are not documents and also not passages in documents, but rather a list of entities, plants with a certain property. This is not only an issue of convenience for the user, but also one of *result diversity*. Even if the search engine would manage to match instances of plants (like *broccoli*) to the keyword *plant*, the problem remains that the result list contains many hits referring to one and the same (well-known) plant, while many (lesser known) plants will be missing.

Worse than that, the information for a single hit could be spread over several documents. For example, for the query *plants with edible leaves and native to Europe*, the information that a particular plant has edible leaves may be contained in one document, while the information that it is native to Europe may be contained in another document. This is beyond the capabilities of full-text search engines.

Unlike for the queries from the previous subsection, *recall* is much more important now. Precision must not be ignored, but becomes a seconday concern. For example, consider the query *apollo astronauts who walked on the moon* from the 2011 Yahoo Semsearch challenge[2]. A user would certainly like to find all 12 entities matching this query. Regarding precision, it would be acceptable if a number of irrelevant results of the *same* order of magnitude were interspersed.

Bottom line: *Entity-oriented queries typically require ontological knowledge. High recall is of primary concern. Precision must not be ignored, but becomes secondary.*

2. ONTOLOGIES

For the purpose of this short paper, we view ontologies as collections of subject-predicate-object triples (often called *facts*), where each element of each triple has an identifier that is consistent among triples. For example, *Broccoli is-*

[1]Let us ignore the untypical case that a precompiled document containing those words and the result list exist.
[2]`semsearch.yahoo.com`

a Vegetable or *Vegetable is-subclass-of Plant* or *Broccoli is-native-to Europe*.

Given an ontology with sufficient information, it is easy to ask even complex queries, which require the connection of many facts, with a precisely defined semantics. For example, the query for all *plants native to europe* would require the connection of all three facts from the previous paragraph.

2.1 Obtaining the facts

The first obvious problem of ontologies is how to obtain the facts. This is easy, if the facts are already organized in a database. Then all that is needed is a conversion to the proper format. A large part of *linked open data (LOD)* [5] and hence of the BTC datasets [7] is of this kind.

Another source of ontology data are users creating machine-readable fact triples explicitly. Only a relatively small part of the BTC datasets is of this kind.

However, much (if not most) information is stored in the form of natural language text, without semantic markup. For the obvious reason that this is the primary form of communication between human beings. For example, there are thousands of documents, including several Wikipedia articles, on the web stating somewhere in a sentence that the leaves of broccoli are edible. But this information is neither contained in DBpedia, nor in current LOD, nor in BTC.

Bottom line: *Much information is available only in the form of natural language text. This is unlikely to change also in the long run. In particular, for recent and specific information.*

2.2 Information extraction

Extracting facts of the form above from natural language text is a hard problem due to the diversity and ill-definedness of natural language. This task, known as *information extraction*, is an offline process. Whatever information was failed to be extracted will not be contained in the ontology, although it should be. Whatever wrong information was extracted will be in the ontology, although it should not be.

It is exactly one of the secrets of success of full-text search that it avoids this problem in the first place. Full-text search engines simply index (almost) every word in all documents. When the document contains your keywords, you have a chance to find it. Also note how, for the sake of precision, a search engine company like Google has introduced new features (like error-tolerance or synonyms or returning) only at a point where they were virtually error-free.

In contrast, state of the art information extraction from natural language text is far from being error-free. For example, the best system on the ACE 2004 dataset for the extraction of 7 predefined relations reported a precision of 83% and a recall of 72% [11], [1]. In [2], a state of the art system that automatically identifies and extracts arbitrary relationships in a text gives an average precision of 88% and a recall of only 45%. Both systems only extract binary relationships and the extraction of multiway relationships is a significantly more complicated task [13].

Bottom line: *Fact extraction from natural language text is an offline problem with a high error rate. The typical recall is 70% or less even for popular relations.*

2.3 Consistent names

The other big problem with ontology data is consistent naming of entities and relations. LOD solves this by unifying different names meaning the same thing via user-created links (*owl:sameAs*). This works well for popular relations and entities, but for more specific and less popular relations, such user-created links are less likely to exist.

Bottom line: *Unified names for entities and relations are feasible for a core of popular facts, but unreasonable to expect for other facts.*

3. INTEGRATION OF FULL TEXT AND ONTOLOGY DATA

In the previous two sections we have argued how a large part of the world's information is (and will be for a long time) available only as full text, while for a certain core of popular knowledge an ontology is the storage medium of choice. We therefore advocate an integration of the two types of search, which we will refer to as *semantic full-text search*.

We see *four* major research challenges associated with such a semantic full-text search. We will describe each of them in one of the following four subsections. We will also comment how we addressed them in our own prototype for an integrated such search, called *Broccoli* [3]. We encourage the reader to try our online demo available under `broccoli.informatik.uni-freiburg.de`

We remark that we are not claiming that our own prototype is the only way to address these research challenges.

3.1 Entity recognition in the full text

An essential ingredient of a system for semantic full-text search is the recognition of references (including anaphora) to entities from the given ontology[3] in the given full text. For example, consider the following sentence: *The stalks of rhubarb are edible, but its leaves are toxic.* Both of the underlined words should map these words to the corresponding entity or entities from the given ontology, for example, `dbpedia.org/resource/Rhubarb`.

For reasonable query times, this kind of entity recognition has to be done *offline*. However, unlike the fact extraction described in Section 2.2, state-of-the-art methods for entity recognition achieve relatively high values for both precision and recall of around 90% [12].

Bottom line: *Offline entity recognition is an essential ingredient of semantic full-text search. The task is much simpler than full information extraction, with precision and recall values of around 90%.*

3.2 Combined Index

Typical entity-oriented queries like our *plants with edible leaves native to Europe* require three things: (1) finding entities matching the ontology part of the query (*plants native to Europe*), (2) finding text passages matching the full-text part of the query (*edible leaves*), and (3) finding occurrences of the entities from (1) that co-occur with the matches from (2).

For both (1) and (2), efficient index structures with fast query times exist. To solve (3), the solutions from (1) and (2) could be combined at query time. However, this is a major obstacle for fast query times, for two reasons. First, the entity recognition problem described in the previous section would have to be solved at query time. Second, even if the

[3]In particular, this could be from LOD or BTC.

final result is small, the separate result sets for (1) and (2) will often be huge, and fully materializing them is expensive.

In our own prototype *Broccoli*, we therefore propose a joint index with hybrid inverted lists that refer to both word and entity occurrences; for details, see [3].

Bottom line: *Semantic full-text search with fast query times seems to require a joint index over both the word and the entity occurrences.*

3.3 Semantic Context

For queries with a large number of hits, prominence of keyword occurrence has turned out to be a very reliable indicator of relevance. However, entity-oriented queries tend to have a long tail of hits with relatively little evidence in the document collection. Then natural language processing becomes indispensable [8].

For example, consider again the query *plants with edible leaves* and again the sentence *The stalks of rhubarb are edible, but its leaves are toxic.* This sentence is one of only few in the whole Wikipedia matching that query. But it should not count as a hit, since in it *edible* refers only to the *stalks* and not to the *leaves.* For such queries, we need an instrument for determining which words semantically "belong together".

In our own prototype Broccoli, we solve this problem by splitting sentences into subsentences of words that belong together in this way. For the sentence above, after anaphora resolution this would be *The stalks of rhubarb are edible* and *Rhubarb leaves are toxic.* Again, see [3] for details.

Bottom line: *Entity-oriented queries often have hits with little evidence in the document collection. To identify those, a natural language processing is required that tells which words semantically "belong together".*

3.4 User interface

We discuss two challenges which are particularly hard and important for semantic full-text search, especially in combination: *ease of use* and *transparency.*

A standard query language for ontology search is SPARQL [10]. The big advantage is its precise query semantics. The big disadvantage is that most users are either not willing or not able (or both) to learn / use such a complex query language. Languages like SPARQL are useful for the work behind the scenes, but not for the front-end.

On the other extreme of the spectrum is keyword search, the simplicity of which is one the secrets of success of full-text search. For full-text search, the query semantics of keyword search is reasonably transparent: the user gets documents which contain some or all of the keywords. For semantic full-text search this is no longer the case. Which part of the query was considered as an entity, which as a word, and which as a class of entities? How were these parts put in relation to each other?

The other major ingredient of transparency, besides a precise query semantics, are *results snippets.* Result snippets serve two main purposes. First, clarifying why the respective hit was returned. Second, allowing the user a quick check whether the hit is relevant.

Systems for what has become known as *ad-hoc object retrieval* [9] try to infer the query semantics from a simple keyword query. Result snippets are treated as a separate problem [6]. In existing semantic search engines on the web, they are often of low quality and little use, e.g. Falcons or SWSE [4].

In our own prototype Broccoli, we use a hybrid approach. Like in keyword search, there is only a single search field. However, using it, the user can build a query, where part of the semantic structure is made explicit. This process is guided by extensive search-as-you-type query suggestions. Due to lack of space here, we refer the reader to our online demo under `broccoli.informatik.uni-freiburg.de`.

Bottom line: *Particular challenges for a user interface for semantic full-text search are ease of use and transparency. Of the currently existing semantic search engines, most neglect one or even both.*

4. REFERENCES

[1] C. C. Aggarwal and C. Zhai, editors. *Mining Text Data*. Springer, 2012.

[2] M. Banko and O. Etzioni. The tradeoffs between open and traditional relation extraction. In *ACL*, pages 28–36, 2008.

[3] H. Bast, F. Bäurle, B. Buchhold, and E. Haussmann. Broccoli: Semantic full-text search at your fingertips. *CoRR*, ad.informatik.uni-freiburg.de/papers, 2012.

[4] C. Bizer, T. Heath, and T. Berners-Lee. Linked data - the story so far. *Int. J. Semantic Web Inf. Syst.*, 5(3):1–22, 2009.

[5] C. Bizer, T. Heath, K. Idehen, and T. Berners-Lee. Linked data on the web. In *WWW*, pages 1265–1266, 2008.

[6] R. Blanco and H. Zaragoza. Finding support sentences for entities. In *SIGIR*, pages 339–346, 2010.

[7] Billion triple challenge dataset 2012. `http://km.aifb.kit.edu/projects/btc-2012/`.

[8] S. T. Dumais, M. Banko, E. Brill, J. J. Lin, and A. Y. Ng. Web question answering: is more always better? In *SIGIR*, pages 291–298, 2002.

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

[10] E. Prud'hommeaux and A. Seaborne. SPARQL query language for RDF. W3C recommendation, W3C, Jan. 2008. http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/.

[11] L. Qian, G. Zhou, F. Kong, Q. Zhu, and P. Qian. Exploiting constituent dependencies for tree kernel-based semantic relation extraction. In *COLING*, pages 697–704, 2008.

[12] E. F. T. K. Sang and F. D. Meulder. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. *CoRR*, cs.CL/0306050, 2003.

[13] S. Sarawagi. Information extraction. *Foundations and Trends in Databases*, 1(3):261–377, 2008.