A Case for Semantic Full-Text Search

Talk @ SIGIR – JIWES
Workshop on Entity-Oriented and Semantic Search
Portland, August 16\textsuperscript{th}, 2012

Hannah Bast
Chair of Algorithms and Data Structures
University of Freiburg

Joint work with Björn Buchhold,
Elmar Haussmann, and Florian Bäurle
Full-text search ... and its limits

- **Document-oriented search**
  - For example: broccoli or broccoli gardening
  - Relevant documents contain keywords or variations
  - Prominence of keywords is good measure of relevance
  - **Huge result sets** → precision is more important than recall

- **Entity-oriented search**
  - For example: plants with edible leaves native to europe
  - Searching for entities of a class, not the name of the class
  - Combine results from different documents **per hit**
  - **Small result sets** → recall is more important than precision
Ontology search ... and its limits

Perfect for fully structured facts and queries
- Fact 1: Broccoli is-a Vegetable
- Fact 2: Vegetable subclass-of Plant
- Fact 3: Broccoli is-native-to Europe
- Query: $1 is-a Plant AND $1 is-native-to Europe

Problems
- Limited amount of manually entered facts / linked open data
  - in particular for very specific and recent information
- Automatic fact extraction from full text is very error-prone
  - see ACE benchmarks ... ACE = automatic content extraction
- Some information is cumbersome to express as facts
  - for example that the leaves of Broccoli are edible
Combined ontology + full-text search

Aspects and challenges

- Entity recognition
  Recognize the entities from the ontology in the full text
- Semantic Context
  Determine which words in the text "belong together"
- Combined Index
  A separate index for both is a barrier for fast query times
- User interface
  Reconcile ease of use and transparency of results

Our own semantic search engine + research paper

- **Broccoli: Semantic full-text search at your fingertips**
- Online Demo + paper at [broccoli.informatik.uni-freiburg.de](http://broccoli.informatik.uni-freiburg.de)
Recognize entities from ontology in the full text

- Example sentence: The stalks of rhubarb, a plant native to Eastern Asia, are edible, however its leaves are toxic

- Example ontology: DBpedia

- Desired result:
  - rhubarb $\rightarrow$ dbpedia.org/resource/Rhubarb
  - Eastern Asia $\rightarrow$ dbpedia.org/resource/East_Asia
  - its $\rightarrow$ dbpedia.org/resource/Rhubarb

- Offline entity recognition is feasible with high prec / recall
  - in particular, much better than full fact extraction
  - again, see the results from the ACE benchmarks
The **Broccoli** solution

- Currently naive approach tailored to the English Wikipedia
  - Trivially resolve entity occurrences with Wikipedia links
  - For the other entity occurrences in a document, consider only entities once linked to before ... in the following variations:
    - Parts of full entity name ... e.g. document links once to *Albert Einstein*, later in the text only *Einstein* is mentioned
    - Anaphoric reference (*he, she, its*, etc.) ... simply resolve to closest previously recognized entity of matching gender
    - Resolve references of the form *the <class>* to last entity of that class ... e.g. **The plant** is known for its edible leaves
Determine which words "belong together"

- Example sentence 1: The **edible** portion of **Broccoli** are the stem tissue, the flower buds, and some small **leaves**
- Example sentence 2: The stalks of **rhubarb**, a plant native to Eastern Asia, are **edible**, however its **leaves** are toxic
- Query: **plants with edible leaves**
- Sentence 1 should be a hit
- Sentence 2 should **not** be a hit
- How to distinguish between the two?
The "straightforward" solution

- Use **term prominence** / **tf.idf**, just like for full-text search.
- Works reasonably well for hits with large "support":
  - Sentences like the one for *broccoli* will be **frequent**, because it is true that the leaves of Broccoli are edible.
  - Sentences like the one for *rhubarb* will be **infrequent**, because the co-occurrence of the query words is random.
- However, even for large text collections, semantic queries tend to have a long tail of hits with little support.
- Then frequency-based distinction does not work anymore.
- It's good for precision in the upper ranks though!
The **Broccoli** solution

- **Decompose** sentences into "parts" that "belong together"

- Example sentence 1: The **edible** portion of **Broccoli** are the stem tissue, the flower buds, and some small **leaves**
  
  Part 1: The edible portion of Broccoli are the stem tissue  
  Part 2: The edible portion of Broccoli are the flower buds  
  Part 3: The edible portion of Broccoli are some small leaves

- Example sentence 2: The stalks of **rhubarb**, a plant native to Eastern Asia, are **edible**, however its **leaves** are toxic
  
  Part 1: The stalks of rhubarb are edible  
  Part 2: However rhubarb leaves are toxic  
  Part 3: rhubarb, a plant native to Eastern Asia
Some of our quality results

- **Dataset:** English Wikipedia ... 1.1 billion word occurrences
- **Queries:** 2009 TREC Entity Track benchmark ... 15 queries
- Comparing three kinds of co-occurrence: within same **section**, within same **sentence**, within same semantic **context**

<table>
<thead>
<tr>
<th></th>
<th># false positives</th>
<th># false negatives</th>
<th>prec.</th>
<th>recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>section</td>
<td>6.890</td>
<td>19</td>
<td>5%</td>
<td>81%</td>
<td>8%</td>
</tr>
<tr>
<td>sentence</td>
<td>392</td>
<td>38</td>
<td>39%</td>
<td>65%</td>
<td>37%</td>
</tr>
<tr>
<td><strong>context</strong></td>
<td><strong>297</strong></td>
<td><strong>36</strong></td>
<td><strong>45%</strong></td>
<td><strong>67%</strong></td>
<td><strong>46%</strong></td>
</tr>
</tbody>
</table>

- For more measures + an in-depth query analysis, see the full research paper available at [broccoli.informatik.uni-freiburg.de](http://broccoli.informatik.uni-freiburg.de)
The "straightforward" solution

- **Separate** index for full-text and for ontology search
  - For example: full text search for *edible leaves* and ontology search for $1$ is-a Plant ; $1$ is-native-to Europe

- **Combine** results at query time

- **Problem:** Result lists for the separate searches, in particular the full-text search, can be huge (even if final result is small)
  - Entity recognition and / or other natural processing in those results at query time is (too) slow
  - When considering only the top-\(k\) hits (e.g. \(k = 1000\)), many rare entities (here: plants) will likely be missed
The *Broccoli* solution

- Build a **combined** index tailored for semantic search
- **Hybrid index lists** for occurrences of words and entities in our semantic contexts, for example:

  \[
  \text{WORD:edible} : \ (C17, \text{Pos 5, WORD:edible}), \\
  \quad (C17, \text{Pos 8, ENTITY:Broccoli}), \\
  \quad (C24, \text{Pos 3, ENTITY:Ivy}), \\
  \quad (C24, \text{Pos 5, WORD:edible}), \\
  \quad (C24, \text{Pos 9, ENTITY:Donkey}), \\
  \quad ...
  \]

- To enable fast query suggestions, we actually use lists for **prefixes** instead of whole words ... see Broccoli paper
Some performance results

- **Dataset:** English Wikipedia ... 1.1 billion postings
- **Queries:** 8,000 queries of various kinds and complexity
- Index has $\approx 3$ times as many postings as std full-text index
- Average query time below 100 milliseconds
- Average time for query suggestions below 100 milliseconds
- Future optimizations: compression, fancy caching, ...
- Next big step: run on 10 – 100 times larger corpus

  But note: even for a dataset like **BTC** much if not most of the actually useful information comes from Wikipedia
  And datasets like **ClueWeb09** contain so much trash ...
Particular challenges for combined search:

- **Transparency**
  
  Full-text search: return documents containing query words + display results snippets containing those words
  
  Ontology search: formal query semantics → no problem
  
  **Combined search:** for most existing engines query interpretation unclear and/or lack of comprehensive result snippets

- **Ease of use**
  
  Full-text search: simple keyword queries
  
  Ontology search: languages like SPARQL are unusable for ordinary users, and even for experts they are painful
  
  **Combined search:** keyword queries lack transparency, more complex languages quickly become unusable
The **Broccoli** solution

- **Single search field** like in ordinary full-text search
- **Full-text search** performed as used to, when user types an ordinary keyword query
- **Semantic search** queries can be constructed via proactive query suggestions (after each keystroke)
  - at any point, structure of current query is visualized
- **Result snippets** come for free with our combined index
  - for other approaches (for example: ad-hoc object retrieval) this becomes a non-trivial problem
Thank you for your attention
Questions please!

And do play around with our demo ... just google broccoli semantic