Entity Search Track submission by Yahoo! Research Barcelona

Roi Blanco, Peter Mika, Hugo Zaragoza

May 20, 2010

1 Introduction

This document describes the indexing and ranking we have performed for the Entity Search Track 2010.

2 Indexing

The data has been indexed using the distributed indexing method described in [3], i.e. implementing distributed indexing for MG4J [2] using Hadoop. We refer the reader to our paper for the details on the indexing process, the various alternative index structures we have implemented, the cost of creating these indices and the size of the resulting indices.

As a first step of processing, we grouped triples about the same subject into virtual documents using Hadoop. We indexed the data primarily using the vertical alternative, but for the Entity Search Track we also added the URI of the object as an index field as shown in Figure 2. We have preselected 300 datatype properties to be indexed based on the frequency of the properties. (We were required to cap the number of indexed fields due to memory requirements during index building.) Object-properties and their values have not been indexed. We have segmented literal values into tokens using MG4J's *FastBufferedReader*. We have blacklisted popular terms and also ignored terms longer than four characters that contained only numbers. As the data set contains text in Asian languages that would not be queried for and would have required special tokenization, we also ignored terms with non-ASCII characters. Lastly, we ignored documents longer than 10,000 triples.

In addition to the vertical index, we have also used the token field of a horizontal index (see Figure 1). Note that the two indices do not necessarily have the same contents, because the horizontal index contains the values for all datatype-properties. We used the horizontal index solely for acquiring global term frequency information (see section 3) and document sizes.

We have also built a document collection for our own testing using MG4J's *SimpleCompressedDocumentCollection*. The building of this collection took con-

Field	p1	p2	p3	p4
token	peter	mika	32	barcelona

Table 1: Horizontal indexing of RDF data

Field	p1	p2	p3	p4
foaf:name	peter	mika		
foaf:age	32			
vcard:location	barcelona			
uri	http	research	yahoo	com

Table 2: Vertical indexing of RDF data

siderable time (close to one week), and the resulting collection is close 80GB in size, but it could be efficiently served by a single machine. It is among the future work for us to implement distributed collection building. Having the collection allowed us to display the query results for testing and fine-tuning the parameters.

3 Ranking

Using the schema just presented we ended up with a document-like representation of the triples, where each final entity is indexed by a number of fields (< 300). We ranked the entities using a variation of BM25F [4].

In Web search, BM25F is employed to aggregate information from the different fields of web pages in such a way that some fields have a bigger impact into the final ranking than others (such as title, body, anchor text, etc.). Potentially, each field could have a different weight; however, given the lack of training data and the huge number of fields, we classified manually some properties into important, unimportant and neutral (default). Then, we assigned a weight *per-class* of property, and we allowed for a different weight to the URI field.

We also integrated per-site document priors, by multiplying the value of this prior to the final BM25F score (like in [1]). Due to the big number of sites we followed the same approach as we did to weight properties (three categories of sites and a weight per category).

Having 300 fields makes it difficult to store and retrieve a file size per document, so we employed the horizontal index's document sizes to normalize every field. Due to the heavy pruning and BM25F promotion of short documents, we included a document threshold that acted as a lower bound for entity length. Finally, we integrated another multiplicative prior that boosted documents that matched all the query terms. In summary, the final model had information about:

- Global term frequency (inverse document frequency) computed using the *horizontal* index

- Document length information coming only from the *horizontal* index
- Local term frequency per property
- Property classification and per-property class weighting.
- Site classification and per-site class weighting
- Boost based on the number of query terms matched

We processed all the queries with Yahoo!'s spell corrector and removed stopwords from the queries.

For tuning the model, we eyeballed some queries. The difference between our three runs was on the selection of parameters (which turned out to be quite blind, due to the lack of training data).

References

[1] R. Blanco and A. Barreiro. Probabilistic document length priors for language models. In *ECIR'08: Proceedings of the IR research, 30th European conference on Advances in information retrieval*, pages 394–405, Berlin, Heidelberg, 2008. Springer-Verlag.

[2] P. Boldi and S. Vigna. MG4J at TREC 2005. In E. M. Voorhees and L. P. Buckland, editors, *The Fourteenth Text REtrieval Conference (TREC 2005) Proceedings*, number SP 500-266 in Special Publications. NIST, 2005. `http://mg4j.dsi.unimi.it/`.

[3] P. Mika. Distributed Indexing for Semantic Search, 2010. http://km.aifb.kit.edu/ws/semsearch10/Files/indexing.pdf.

[4] S. Robertson and H. Zaragoza. The probabilistic relevance framework: BM25 and beyond, foundations and trends in information retrieval. volume 3, pages 333–389, 2009.