Poster Paper

Type-aware Web-search

Michael Gubanov
University of Texas at San Antonio
mikhail.gubanov@utsa.edu

Anna Pyayt
University of South Florida
pyayt@usf.edu

ABSTRACT

Keyword-search engines (e.g. Web-search) usually can be outperformed by a specialized system optimized for a specific domain, type of data, or queries [8, 2, 12, 5, 11, 9]. For example, Halevy et. al. in [13] demonstrate how a specialized Google Fusion Tables spatial search can outperform the general-purpose Google Web-search on *bike trails search* in San Francisco Bay Area. At the same time, Web content providers usually exhibit a specific focus for their postings. For example, information at `http://www.csail.mit.edu` is devoted to Computer Science research and education, *Hannah Montana* is mostly tweeting about music, and the same is true for most sources.

This paper describes the work in progress on a new Type-aware Web-search system that uses topical focus of information sources to process a large class of queries better than a regular Web search-engine. It leverages semantic profiles similar to [10, 6, 7] and a new Type-aware Locality-Sensitive Hashing (TLSH) scheme to accomplish it.

1. INTRODUCTION

Figure 1 illustrates search-results from one of the Web search engines for query *Frozen in Phoenix* where a user is trying to find a theater to watch a movie. You can see the search-results are not the best (about ice cream and frozen yogurt). It happened, because the generic Web-search engine employs simple term matching of the query with the Web pages, and did not take into account type information, which can be done to get more relevant results. Table 1 illustrates the Web-search results of a *type-aware* search-engine described here for queries *Careers of People with Ph.D.* You can see, it returns precisely what the user has been asking for in these queries. A regular Web-search engine would return career-pages of companies and recruiting agencies, resulting from term matching to *careers*.

Categories and Subject Descriptors

H.2 [**Database Management**]: Heterogeneous Databases

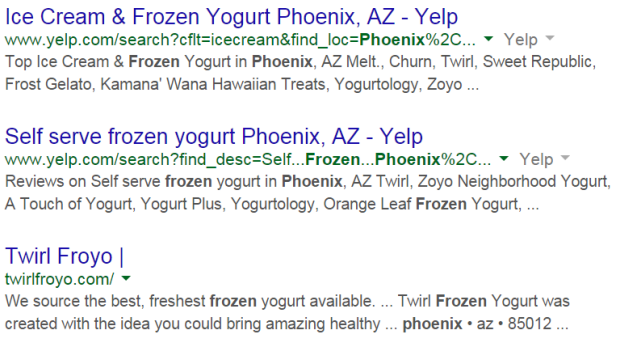

Figure 1: Search-results for *Frozen in Phoenix*

```
Query: Careers of People with Ph.D.
Results: - Romania News Watch:
...Ponta obtained his PhD from the University of
Bucharest while acting as Secretary of State
in the government of an earlier prime minister..
```

Table 1: Type-aware Search Results

2. ARCHITECTURE

The crawled Web pages are processed by a Natural Language Processing domain-dependent parser, which emits the entity data along with the text fragments they came from and saves the result into a large-scale storage (see Figure 2 for a schematic). Both a large-scale semi-structured sharded storage engine as well as a parallel relational engine are used.

The earlier work in [10] introduces *semantic profiles* intended to capture the semantics of an information source and store it in a compact and reusable manner. It summarizes and accumulates all *types* of entities from the source. For example, the newspaper *New York Times* often publishes about `companies`, `products`, and `organizations`; *The Finance* usually tweets about `dividends` and `products`; *The Oregonian* publishes about `sports`, `holidays`, `music`, and hence their profiles are comprised of these types. These profiles are calculated and saved for each source. Due to space limitations, interested readers are referred to [10] for more details on profile construction.

Next, the hashing routines treat each profile as a vector and assign it to one of the hash tables. Similarly, the incoming query is represented a vector, the query processing module computes the set of relevant hash tables for a query, the relevance score of the documents from these hash tables

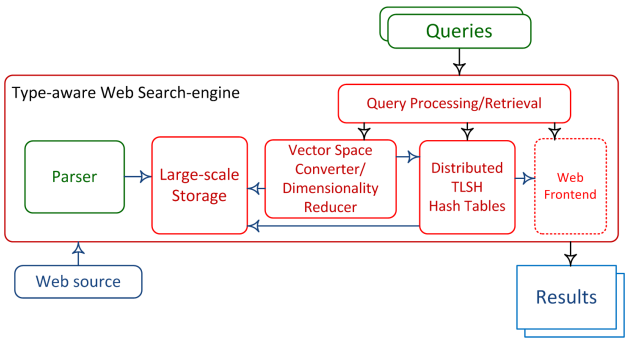

Figure 2: Architecture

and the query is computed, and finally the documents are ranked by this relevance score and output to the user.

3. TYPE-AWARE WEB-SEARCH

Type-aware Locality Sensitive Hashing: Locality-Sensitive Hashing (LSH) [14] is an algorithm that enables searching for near neighbors in a high-dimensional vector space S^n with dimensionality n. Formally, given a query $q \in S^n$, return the nearest neighbors of q within certain radius R. LSH performance crucially depends on a family of hash functions F that it uses to map the input vectors to its internal data structures. In order for the algorithm to perform well, F usually has to reduce the dimensionality of the original vector space still satisfying the *locality-sensitive* requirements on the reduced vector space. F is considered to be *locality-sensitive* if collision of two vectors v_1 and v_2 under a random choice of a hash function from F depends only on the distance between v_1 and v_2. Refer to [3] for an overview of locality-sensitive hash-function families.

Here, a new two-tier family of hash functions Ψ is described and used. First, it maps the original vector space V of *terms* into a vector space of *types* - T, hence reduces dimensionality (there are much less types than terms). Second, k random unit vectors $u \in T$ are generated, which defines a family of hash-functions $h \in \Psi$ as follows $h(v) = sign(u \cdot v/\|v\|) : u, v \in T$. Refer to [4] for a proof of its locality-sensitivity. Angular distance measure is used here for this vector space.

Query Processing: The queries and Web documents are represented as vectors in a high-dimensional vector space S^n with dimensionality n (number of types). To return vectors (Web documents) within raduis R of the query q the algorithm concatenates k hash functions $h_i \in \Psi$ described above into a composite hash function $h_c(v) = h_1(v), .., h_k(v)$, hence creating a family of hash functions $hc \in \Phi$.

Next, for query q it computes all functions from h_c and considers the documents only from the corresponding hash tables. It returns all vectors v from those hash tables that are within angular distance R from q. The evaluation below justifies that using this semantic hashing/retrieval algorithm outperforms a generic Web-search engine by relevance of search-results.

Relevance Evaluation: Here, relevance gain of TLSH hashing/retrieval scheme compared to a general purpose Web-search engine for "type-containing" queries (i.e. containing a Named-entity) is quantitatively evaluated. An experiment was conducted to calculate NDCG (Normalized Discounted Cumulitive Gain) [1] on a static set of queries with respect to a general purpose Web-search engine, which provides quantitative insight into their performance difference. NDCG is one of the standard widely used search relevance measures, which is employed by major search engines and, similarly to F-measure, measures both precision and recall of retrieval. NDCG is well suited for search evaluation, because it rewards relevant results in the top positions more than those ranked lower. Due to space limitations, interested readers are referred to [1] for details about NDCG computation. Total NDCG gain over all queries turned out to be very large $> 6\%$. Usually for two industrial Web-search engines NDCG difference more than 4% is considered to be significant.

4. REFERENCES

[1] E. Agichtein, E. Brill, and S. Dumais. Improving web search ranking by incorporating user behavior information. In *SIGIR*, 2006.

[2] B. Alexe, M. Gubanov, M. Hernandez, H. Ho, J.-W. Huang, Y. Katsis, L. Popa, B. Saha, and I. Stanoi. Simplifying information integration: Object-based flow-of-mappings framework for integration. In *BIRTE*, volume 27 of *Lecture Notes in Business Information Processing*, pages 108–121. Springer, 2009.

[3] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Commun. ACM*, 51(1):117–122, Jan. 2008.

[4] M. S. Charikar. Similarity estimation techniques from rounding algorithms. In *STOC*. ACM, 2002.

[5] S. Cheemalapati, M. Gubanov, M. Del Vale, and A. Pyayt. A real-time classification algorithm for emotion detection using portable eeg. In *IRI*. IEEE, 2013.

[6] M. Gubanov and A. Pyayt. Readfast: High-relevance search-engine for big text. In *ACM CIKM*, 2013.

[7] M. Gubanov and A. Pyayt. Readfast: Optimizing structural search relevance for big biomedical text. In *IRI*, 2013.

[8] M. Gubanov and L. Shapiro. Using unified famous objects (ufo) to automate alzheimer's disease diagnostics. In *BIBM*, 2012.

[9] M. Gubanov, L. Shapiro, and A. Pyayt. Readfast: Structural information retrieval from biomedical big text by natural language processing. In *Information Reuse and Integration in Academia and Industry*, pages 187–200. Springer, 2013.

[10] M. Gubanov and M. Stonebraker. Large-scale semantic profile extraction. In *EDBT*, 2014.

[11] M. Gubanov, M. Stonebraker, and D. Bruckner. Text and structured data fusion in data tamer at scale. In *ICDE*, 2014.

[12] M. N. Gubanov, P. A. Bernstein, and A. Moshchuk. Model management engine for data integration with reverse-engineering support. In *ICDE*, 2008.

[13] A. Halevy. Data publishing and sharing using fusion tables. In *CIDR*, 2013.

[14] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. STOC. ACM, 1998.