BE: A Search Engine for NLP Research

Michael J. Cafarella, Oren Etzioni
Department of Computer Science and Engineering
University of Washington
Seattle, WA 98195-2350
{mjc,etzioni}@cs.washington.edu

Many modern natural language-processing applications utilize search engines to locate large numbers of Web documents or to compute statistics over the Web corpus. Yet Web search engines are designed and optimized for simple human queries—they are not well suited to support such applications. As a result, these applications are forced to issue millions of successive queries resulting in unnecessary search engine load and in slow applications with limited scalability.

In response, we have designed the Bindings Engine (BE), which supports queries containing *typed variables* and *string-processing functions* (Cafarella and Etzioni, 2005). For example, in response to the query *"powerful ⟨noun⟩"* BE will return all the nouns in its index that immediately follow the word "powerful", sorted by frequency. (Figure 1 shows several possible BE queries.) In response to the query *"Cities such as ProperNoun(Head(⟨NounPhrase⟩))"*, BE will return a list of proper nouns likely to be city names.

president Bush <*Verb*>
cities such as *ProperNoun(Head(<NounPhrase>))*
<*NounPhrase*> is the CEO of <*NounPhrase*>

Figure 1: **Examples of queries that can be handled by BE. Queries that include typed variables and string-processing functions allow certain NLP tasks to be done very efficiently.**

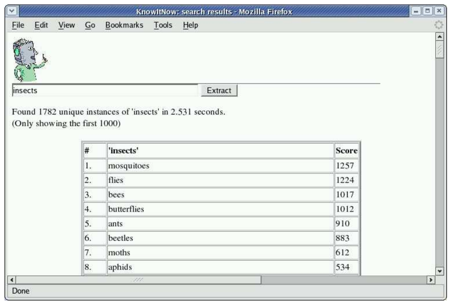

Figure 2: **Most-frequently-seen extractions for query "insects". The score for each extraction is the number of times it was retrieved over several BE extraction phrases.**

BE's novel *neighborhood index* enables it to do so with $O(k)$ random disk seeks and $O(k)$ serial disk reads, where k is the number of non-variable terms in its query. A standard search engine requires $O(k + B)$ random disk seeks, where B is the number of variable "bindings" found in the corpus. Since B is typically very large, BE vastly reduces the number of random disk seeks needed to process a query. Such seeks operate very slowly and make up the bulk of query-processing time. As a result, BE can yield several orders of magnitude speedup for large-scale language-processing applications. The main cost is a modest increase in space to store the index.

To illustrate BE's capabilities, we have built an application to support interactive information extraction in response to simple user queries. For example, in response to the user query "insects", the application returns the results shown in Figure 2. The application generates this list by using the query term to instantiate a set of generic extraction phrase queries such as "insects such as ⟨*NounPhrase*⟩". In effect, the application is doing a kind of query expansion to enable naive users to extract information. In an effort to find high-quality extractions, we sort the list by the hit count for each binding, summed over all the queries.

The key difference between this BE application, called KNOWITNOW, and domain-independent information extraction systems such as KNOWITALL (Etzioni et al., 2005) is that BE enables extraction at interactive speeds — the average time to expand and respond to a user query is between 1 and 45 seconds. With additional optimization, we believe we can reduce that time to 5 seconds or less. A detailed description of KNOWITNOW appears in (Cafarella et al., 2005).

References

M. Cafarella and O. Etzioni. 2005. A Search Engine for Natural Language Applications. In *Procs. of the 14th International World Wide Web Conference (WWW 2005)*.

M. Cafarella, D. Downey, S. Soderland, and O. Etzioni. 2005. Knowitnow: Fast, scalable information extraction from the web. In *Procs. of EMNLP*.

O. Etzioni, M. Cafarella, D. Downey, S. Kok, A. Popescu, T. Shaked, S. Soderland, D. Weld, and A. Yates. 2005. Unsupervised named-entity extraction from the web: An experimental study. *Artificial Intelligence*, 165(1):91–134.