QUICK: Expressive and Flexible Search over Knowledge Bases and Text Collections

Jeffrey Pound
David R. Cheriton School of Computer Science
University of Waterloo
Waterloo, Canada
jpound@cs.uwaterloo.ca

Ihab F. Ilyas
David R. Cheriton School of Computer Science
University of Waterloo
Waterloo, Canada
ilyas@cs.uwaterloo.ca

Grant Weddell
David R. Cheriton School of Computer Science
University of Waterloo
Waterloo, Canada
gweddell@cs.uwaterloo.ca

1. INTRODUCTION

Recent work on Web-extracted data sets has produced an interesting new source of structured Web data. These data sets can be viewed as knowledge bases (KB) – large heterogeneous linked entity collections with millions of unique edge and node labels, often encoding rich semantic information over entities. For example, YAGO [5] and ExDB [2] have fact collections numbering in the tens and hundreds of millions respectfully, and WebTables [1] contains over one hundred million extracted relations. In terms of schema information, the ExDB, YAGO, and WebTables data sets all have schema items numbering in the millions.

Due to the sheer size of the schema information in these Web-extracted KBs, there has been limited development of novel tools and technologies to expose these unique data sets to end users. One of the main factors impeding the development of useful applications over these data sets is the *information overload problem* caused by the massive heterogeneous schema information. With KB schema items numbering in the millions, formulating structured queries is a daunting task for an application developer. Most applications are therefore built on keyword search, which does not allow direct access to the expressive structure of the data, or are applications built on structured queries with the assumption that the users are able to digest these massive schemas.

As an example, consider a user searching for all *scientists that have won a nobel award.* Exploiting the structure of a knowledge base, a user may formalize the information need as the following SPARQL query.

$$
\begin{array}{l}
\textbf{SELECT} \ \ ?x \\
\textbf{WHERE} \ \ \{ \\
\quad ?x \ \ \textit{type} \ \ \texttt{SCIENTIST} \ . \\
\quad ?x \ \ \textit{hasWonPrize} \ \ ?y \ . \\
\quad ?y \ \ \textit{type} \ \ \texttt{NOBEL\_PRIZE} \ . \\
\}
\end{array}
\qquad (1)
$$

Proceedings of the VLDB Endowment, Vol. 3, No. 2
Copyright 2010 VLDB Endowment 2150-8097/10/09... $ 10.00.

However, a user formulating this query may have literally *millions* of choices for terms like NOBEL_PRIZE and *hasWonPrize.* In such a scenario, the utility of the structured query language is greatly diminished by the complexity of composing well formed queries. Alternatively, to satisfy the information need, a user may pose the following keyword query.

"scientists have won nobel award" (2)

This query searches for all data items having an occurrence of the given keywords. While this alleviates the information overload problem, it comes at a cost of expressivity. It is not clear in this query what is being searched for (the scientists or the awards?), nor is it explicit what the relationship between keywords should be (which keywords describe entities, types, and/or relations?).

To address these issues, we propose *QUICK* (Queries Using Inferred Concepts from Keywords), a novel information system that allows expressive yet flexible entity retrieval over Web-extracted knowledge bases. Our approach blends keyword search with structured queries, allowing complex information needs to be expressed without intimate knowledge of the underlying schema. Our system also supports reasoning over type hierarchies with an efficient query-time algorithm. To this extent, *QUICK* could accommodate the example information need exploiting structure as in (1) but using keywords as in (2).

"scientists, have won (nobel award)"

Furthermore, we show how knowledge from the Web-extracted knowledge base can be used as an aid in document retrieval, enabling semantic search over a text corpus that exploits structured information with expressive queries.

2. THE QUICK SYSTEM

Figure 1 shows an overview of our complete system architecture. We extract named entities from text documents for indexing, and also index the text to support full text search. We index the content and structure of a Web-extracted knowledge base to allow efficient access to entities based on structured concept queries generated by our disambiguation procedure. We also collect statistics from the KB to aid in the disambiguation process.

At run-time, our system takes a structured keyword query describing entities of interest. Since the query is based on keywords, there is an inherent ambiguity as to the intent of the query. A disambiguation procedure is run to compute the top ranking query interpretations in the form of a

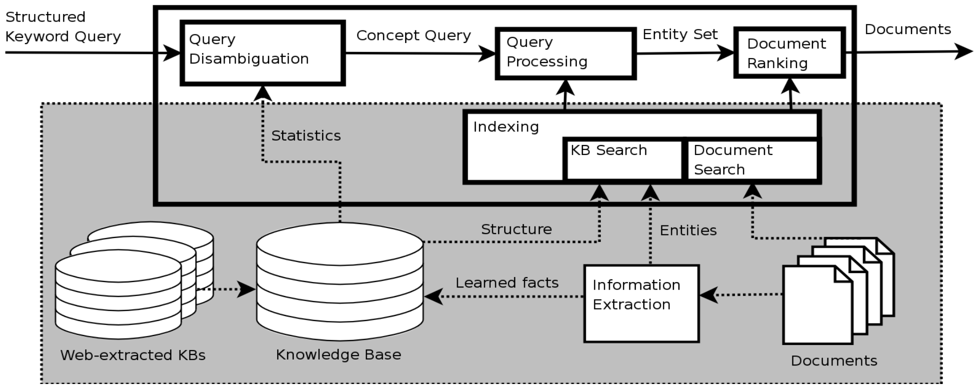


Figure 1: *QUICK* system architecture.

structured concept query over the KB. One or more of these queries is then run against the KB index to find relevant entities, and entity-based document search is used to find documents relevant to the entities. In the case where full disambiguation cannot be achieved, we also integrate residual keywords to aid in document ranking. In the case where no disambiguation can be achieved, our system degrades to a regular keyword search engine.

In the remainder of this section we review the main contributions of the *QUICK* architecture, including the structured keyword query language, query disambiguation, indexing and query evaluation. We use existing solutions to support the other components of the system.

2.1 Structured Keyword Queries

Our system is driven by a blend of structured and keyword queries originally presented in [4]. This query language enables expressive descriptions of information needs while allowing the flexibility of keywords as base level constructs. The language is defined as follows.

Definition 1. **(Structured Keyword Query)** Let k be a keyword phrase (one or more keywords), then a *structured keyword query* Q is defined by the following grammar.

$$\begin{array}{rcl} Q & ::= & k \\ & | & k(Q) \\ & | & Q_1, Q_2, ..., Q_n \end{array}$$

The first construct allows an entity or type in a query to be described by a set of one or more keywords (e.g., "*nobel prize*" or "*scientist*"). The second construct allows the description of an entity in terms of the relationships it has to other entities or types. (e.g., "*born in(Canada)*"). The third construct allows a set of queries to describe a single class of entities in conjunction (e.g., "*harmonica player, songwriter*").

Note that relationship queries allow an arbitrary nesting of subqueries, such as the query "*born in(country, has official language(spanish))*" which could be used to query people born in Spanish speaking countries.

Our system also supports a top-level query language in which multiple structured keyword queries can express that documents of interest should contain multiple types of entities, as described by each subquery.

2.2 Query Disambiguation

Because our queries are based on keywords, there is ambiguity in the meaning of the query. Our system implements a sophisticated disambiguation approach to quickly find top ranking candidate query interpretations expressed in the vocabulary of the given KB. The ranking is based on a combination of syntactic similarity, which captures how closely the candidate mapping matches the users description of their information need, and semantic similarity, which approximates the semantic coherence of the candidate query interpretation with respect to the underlying KB. Intuitively, good query interpretations will reflect the users information need as closely as possible, while producing an expression that is coherent with respect to the knowledge encoded in the KB.

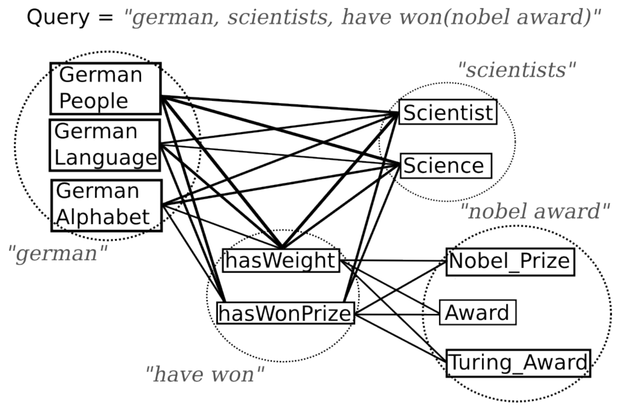
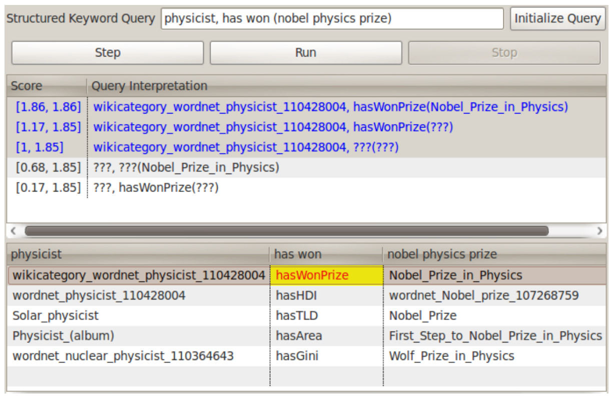


Figure 2: An example query disambiguation graph.

Figure 2 shows an example of a graph representation of the disambiguation procedure. The square nodes represent candidate KB items for a given keyword phrase shown in italicized quotes. The edges denote potential joins on semantic similarities. For each term in the query we generate a *partition*, a candidate set of KB items that the query term may represent using very relaxed syntactic matching. The


Structured Keyword Query: physicist, has won (nobel physics prize)

Score	Query Interpretation
[1.86, 1.86]	wikicategory_wordnet_physicist_110428004, hasWonPrize(Nobel_Prize_in_Physics)
[1.17, 1.85]	wikicategory_wordnet_physicist_110428004, hasWonPrize(???)
[1, 1.85]	wikicategory_wordnet_physicist_110428004, ???(???)
[0.68, 1.85]	???, ???(Nobel_Prize_in_Physics)
[0.17, 1.85]	???, hasWonPrize(???)

physicist	has won	nobel physics prize
wikicategory_wordnet_physicist_110428004	hasWonPrize	Nobel_Prize_in_Physics
wordnet_physicist_110428004	hasHDI	wordnet_Nobel_prize_107268759
Solar_physicist	hasTLD	Nobel_Prize
Physicist_(album)	hasArea	First_Step_to_Nobel_Prize_in_Physics
wordnet_nuclear_physicist_110364643	hasGini	Wolf_Prize_in_Physics

Figure 3: Visualizing the disambiguation algorithm.

goal is then to choose one item for each query term in order to build a query interpretation in the vocabulary of the KB. The number of such interpretations is exponential in the length of the query, thus we must be selective in which query interpretations we consider.

We begin by sorting each partition according to the syntactic similarity of the KB items to the query term. We then consider the coherence of having KB items from each partition in conjunction in order to estimate the semantic coherence of possible combinations of KB items. We traverse partitions in order, terminating when the k highest scoring partitions are found. The algorithm may terminate early by computing upper and lower bounds on the scores of each candidate being considered. This algorithm is a variation of the threshold algorithm known as a rank-join. In this case, we join KB items together that have non-zero semantic similarities, and join across nested queries based on the same binding of the KB relation for keywords denoting relations. Full details of our scoring function and disambiguation model can be found in [4].

2.3 KB Indexing and Query Evaluation

We have developed a custom tool for indexing and querying semantic graphs while appealing to inference over acyclic type hierarchies. The index builds on standard notions of graph representation, making use of adjacency lists stored in a scalable key-value store. A key observation for our class of concept queries is that we need only index the inverse directions of relations in order to efficiently support query evaluation. This reduces the size of our index by half, as compared to a full graph index. The second decision in our index design is to support inference over the transitive "is-a" relation using a lazy, query-time evaluation technique. This again gives us a very compact representation compared to systems that pre-compute and index the full graph closure. Because of these two space saving properties of our index, we are able to store large KBs (such as YAGO [5]) in memory and achieve very efficient query evaluation. Details of our index and query processing algorithm, along with performance experiments can be found in [3].

2.4 Document Ranking

Ultimately, the function of our system is to retrieve documents of interest. There are two components to document retrieval and ranking in our system. First, our system uses the entities that qualify for the concept queries to retrieve

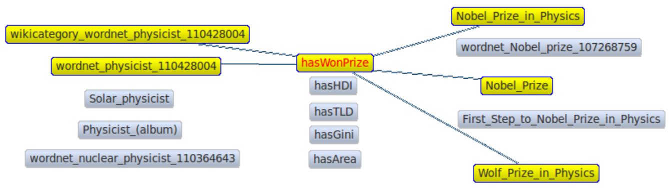

Nobel_Prize_in_Physics
wikicategory_wordnet_physicist_110428004
wordnet_Nobel_prize_107268759
wordnet_physicist_110428004
hasWonPrize
hasHDI
Nobel_Prize
Solar_physicist
hasTLD
First_Step_to_Nobel_Prize_in_Physics
Physicist_(album)
hasGini
wordnet_nuclear_physicist_110364643
hasArea
Wolf_Prize_in_Physics

Figure 4: Visualizing the query disambiguation graph.

documents describing these entities. Because we annotate entities in documents at index time, we are able to perform accurate entity-based retrieval of documents. Second, keywords that were not mapped into the disambiguated concept query, due to a failure in the mapping procedure or incompleteness in the KB, are used to re-rank the candidate document set, boosting the score of documents containing the residual keywords. This can be done efficiently using the full text index. We retrieve all documents containing the residual keywords, and boost the documents in the intersection of this set and the set of documents mentioning the entities of interest. In our prototype, the boosting is done by adding the relevance scores assigned by our document retrieval engine, which is based on $tf \cdot idf$ scoring.

2.5 Implementation Details

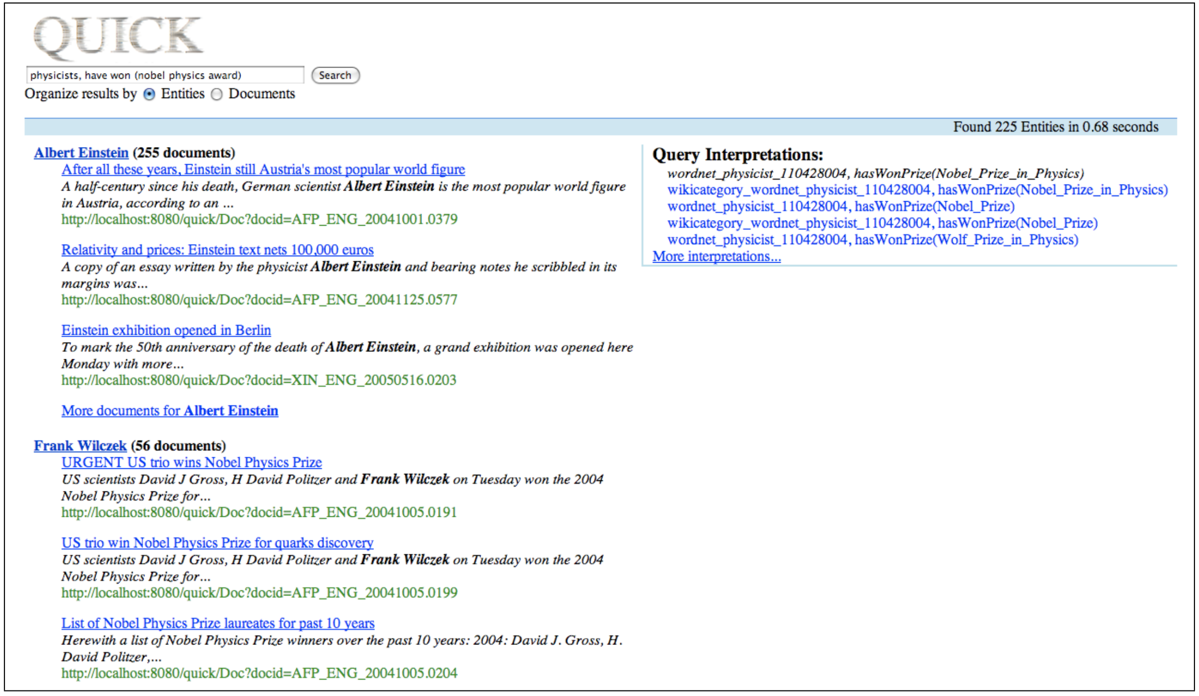
We have built the entire *QUICK* system as depicted in Figure 1 with the exception of the feedback mechanism for learning new facts. We are developing a number of experimental interfaces to apply this technology.

QUICK integrates a variety of open-source libraries as well as custom built components. We use the Lucene library[1] to build both the full text index and entity-based document index. Named entities are annotated at index time using a text processor built on top of the OpenNLP library[2]. This tool annotates person, location, and organization entities occurring in the text. We also create a Lucene index over all of the schema item labels of the KB, this is used to quickly generate the candidate sets for each keyword expression in the query. Our KB index uses Berkeley DB[3] as the key-value store. Our concept query processor is a custom solution written in Java. The query disambiguation module is also written in Java and uses a separate Berkeley DB store for pre-computed pairwise semantic similarities between all KB items. Our scoring function uses q-gram distance for syntactic similarity and the Jaccard index for semantic similarity as discussed in [4].

3. DEMO SCENARIO

We deploy our system over the YAGO KB [5] and a large news corpus. We use the 2008 version of YAGO as our knowledge base (note that this is approximately three times larger than the initial instance reported in [5]). YAGO consists of about 6 GB of raw ontology and entity data. It contains over 2 million entities, about 250,000 primitive concepts, 100 relations, and over 20 million facts about these entities, concepts, and relations. We use the AQUAINT2

[1]http://lucene.apache.org/
[2]http://opennlp.sourceforge.net/
[3]http://www.oracle.com/technology/products/berkeley-db/je/index.html


QUICK

physicists, have won (nobel physics award) Search

Organize results by [✓ Entities] Documents

Found 225 Entities in 0.68 seconds

Albert Einstein (255 documents)

After all these years, Einstein still Austria's most popular world figure
*A half-century since his death, German scientist **Albert Einstein** is the most popular world figure in Austria, according to an ...*
http://localhost:8080/quick/Doc?docid=AFP_ENG_20041001.0379

Relativity and prices: Einstein text nets 100,000 euros
*A copy of an essay written by the physicist **Albert Einstein** and bearing notes he scribbled in its margins was...*
http://localhost:8080/quick/Doc?docid=AFP_ENG_20041125.0577

Einstein exhibition opened in Berlin
*To mark the 50th anniversary of the death of **Albert Einstein**, a grand exhibition was opened here Monday with more...*
http://localhost:8080/quick/Doc?docid=XIN_ENG_20050516.0203

More documents for **Albert Einstein**

Frank Wilczek (56 documents)

URGENT US trio wins Nobel Physics Prize
*US scientists David J Gross, H David Politzer and **Frank Wilczek** on Tuesday won the 2004 Nobel Physics Prize for...*
http://localhost:8080/quick/Doc?docid=AFP_ENG_20041005.0191

US trio win Nobel Physics Prize for quarks discovery
*US scientists David J Gross, H David Politzer and **Frank Wilczek** on Tuesday won the 2004 Nobel Physics Prize for...*
http://localhost:8080/quick/Doc?docid=AFP_ENG_20041005.0199

List of Nobel Physics Prize laureates for past 10 years
Herewith a list of Nobel Physics Prize winners over the past 10 years: 2004: David J. Gross, H. David Politzer,...
http://localhost:8080/quick/Doc?docid=AFP_ENG_20041005.0204

Query Interpretations:

wordnet_physicist_110428004, hasWonPrize(Nobel_Prize_in_Physics)
wikicategory_wordnet_physicist_110428004, hasWonPrize(Nobel_Prize_in_Physics)
wordnet_physicist_110428004, hasWonPrize(Nobel_Prize)
wikicategory_wordnet_physicist_110428004, hasWonPrize(Nobel_Prize)
wordnet_physicist_110428004, hasWonPrize(Wolf_Prize_in_Physics)
More interpretations...

Figure 5: A query result interface grouping relevant documents by entities. The top ranking query interpretations are also shown on the right.

news collection as our corpus, consisting of over 900,000 English news articles from various sources.

We demonstrate entity-based text retrieval using structured keyword queries with our automated disambiguation procedure. In this scenario, arbitrary structured keyword queries are taken as input to the system. The top ranking disambiguation is computed and evaluated over YAGO, appealing to the encoded semantics using a query-time inference procedure over our KB index. Qualifying entities are used to find relevant documents using entity-based document retrieval.

Participants will be able to experiment with the live system, issuing queries over a real-world KB and document collection. Additionally, we will demonstrate a tool that visualizes our disambiguation algorithm, illustrating how candidates are generated and ranked during the disambiguation procedure. Figure 3 shows part of an interface for visualizing the disambiguation algorithm. Each candidate KB item is considered one step at a time to show how concept queries are constructed and ranked.

Figure 4 shows a visualization of the query disambiguation graph. As the disambiguation algorithm is executed, relevant parts of the graph are highlighted to illustrate the candidate connections for joining query terms.

Figure 5 shows an interface for displaying results. A structured keyword query is taken as input and the top ranking disambiguations are computed and displayed in the right-hand column. The top interpretation is executed and the resulting documents are displayed in the main area grouped by entities. An alternate organization for results would allow the document pool to be ranked as a whole.

4. CONCLUSION

We have proposed a demonstration of a new type of search capability, one that integrates the expressivity of structured queries with the flexibility of keyword queries. This technology enables access to large heterogeneous data sets that result from automated Web extractions. We have also proposed an application of this search technology, utilizing a Web-extracted KB for the task of document retrieval.

5. REFERENCES

[1] M. J. Cafarella, A. Halevy, D. Z. Wang, E. Wu, and Y. Zhang. WebTables: exploring the power of tables on the web. *Proc. VLDB Endow.*, 1(1):538–549, 2008.

[2] M. J. Cafarella, C. Re, D. Suciu, and O. Etzioni. Structured Querying of Web Text Data: A Technical Challenge. In *CIDR*, pages 225–234, 2007.

[3] J. Pound, I. F. Ilyas, and G. Weddell. QUICK: Queries Using Inferred Concepts from Keywords. Technical report, Technical Report CS-2009-18, University of Waterloo, 2009.

[4] J. Pound, I. F. Ilyas, and G. Weddell. Expressive and Flexible Access to Web-Extracted Data: A Keyword-based Structured Query Language. In *SIGMOD '10: Proceedings of International Conference on Management of Data*, pages 423–434, 2010.

[5] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: A Core of Semantic Knowledge - Unifying WordNet and Wikipedia. In *16th Intl. World Wide Web Conference (WWW 2007)*, pages 697–706, 2007.