Searching the Deep Web Using Proactive Phrase Queries

Wensheng Wu, Tingting Zhong
University of North Carolina at Charlotte
{w.wu, tzhong}@uncc.edu

ABSTRACT

This paper proposes Deep2Q, a *novel* search engine that *proactively* transforms query forms of Deep Web sources into phrase queries, constructs query evaluation plans, and caches results for popular queries offline. Then at query time, keyword queries are simply matched with phrase queries to retrieve results. Deep2Q embodies a novel *dual-ranking* framework for query answering and novel solutions for discovering frequent attributes and queries. Preliminary experiments show the great potentials of Deep2Q.

Categories and Subject Descriptors

H.2.5 [**Database Management**]: Heterogeneous Databases

Keywords

Deep Web; proactive search engine; natural language queries

1. INTRODUCTION

The *Deep Web* contains over 25 millions of online data sources whose contents are typically only accessible through their form-based query interfaces [4]. These data sources are becoming indispensible resources for our daily life, from online shopping, flight reservation, to searching for jobs. However, a *serious* problem with the form-based interfaces is that users need to locate right sources, understand often complex query forms, and pose separate form queries—an extremely labor-intensive and painstaking process.

To address this challenge, one solution is to allow users to pose keyword queries on search engines [4, 5]. The engines will then reformulate the queries into form queries and pose them to sources to obtain answers. However, this solution often suffers from slow query response. One reason is that query reformulation may take time since it involves many challenging tasks: (1) *query parsing*, e.g., it needs to recognize "books bill clinton 2012" contains an entity name, an author, and a year; (2) *query translation*, e.g., it needs to find appropriate sources (e.g., bookstores that accept queries on author and year) and translate keyword queries into form queries. Another reason is that data sources may be slow in responding to queries and often multiple sources need to be accessed to obtain the answers. To make matters worse, keyword queries are known for their *ambiguities*. For example, "books bill clinton" could mean "books written by bill clinton" or "books about bill clinton" or both.

WWW 2013 Companion, May 13–17, 2013, Rio de Janeiro, Brazil.
ACM 978-1-4503-2038-2/13/05.

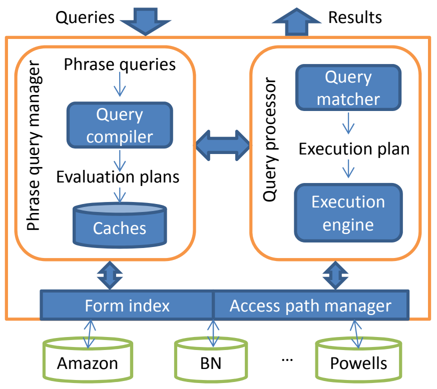


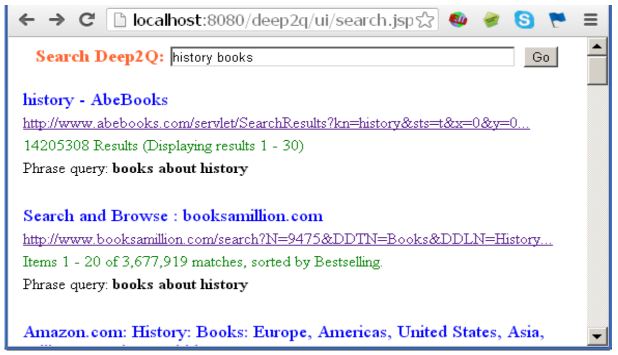
Figure 1: The architecture of Deep2Q

2. OUR PROPOSAL

We propose to tackle the problem from an *opposite* direction: instead of transforming keyword queries into form queries at query time, we propose a Deep Web search engine that *proactively* transforms web forms into *phrase queries* and constructs query evaluation plans (e.g., searching and combining bill clinton books from Amazon and Barnes & Noble) offline. For popular queries, the engine may also pre-execute the queries and cache results to achieve fast query response. Then at query time, the engine only needs to find phrase queries similar to user keyword queries and retrieve results by either executing the precompiled evaluation plans or looking up in the caches.

We start by considering *noun phrase* queries of the form "$M\ N\ O$" (e.g., "history books for kids") where N is a head noun (e.g., books) and M (e.g., "history") and O (e.g., "for kids") are its pre-modifiers and post-modifiers respectively. Each phrase query corresponds to one or more form queries which, when executed, will access sources to obtain query answers. Note that noun phrase queries are very common on search engines [3]. However, our solution is generally applicable to other query forms.

Advantages: First, phrase queries *abstract away* the tedious details of formulating form queries and accessing data sources. They are in a sense mediators between user queries and the data on the Deep Web. Second, it avoids the expensive parsing at query time: user queries can be simply matched with phrase queries using standard IR techniques. Third, phrase queries are *more precise* than keyword queries and may be used to disambiguating keyword queries, e.g., refining "bill clinton books" into "books written by bill clinton". Lastly, the engine may cache results of *popular* queries to enable instant response for these queries.



localhost:8080/deep2q/ui/search.jsp

Search Deep2Q: history books [Go]

history - AbeBooks
http://www.abebooks.com/servlet/SearchResults?kn=history&sts=t&x=0&y=0...
14205308 Results (Displaying results 1 - 30)
Phrase query: **books about history**

Search and Browse : booksamillion.com
http://www.booksamillion.com/search?N=9475&DDTN=Books&DDLN=History...
Items 1 - 20 of 3,677,919 matches, sorted by Bestselling.
Phrase query: **books about history**

Amazon.com: History: Books: Europe, Americas, United States, Asia,

Figure 2: The search interface of Deep2Q

Challenges: First, we may not have access to logs of form queries users posed on the Deep Web sources. Second, some attributes (e.g., book author) can have many possible values. Third, there may be a large number of combinations of values from different attributes. Thus a *naive* approach of generating one phrase query for every attribute value and combination of values may result in an excessive number of queries that are rarely asked. Fourth, some attributes might not even have values available on query forms.

3. Deep2Q SOLUTION

We present Deep2Q, a novel proactive search engine for the Deep Web that addresses the above challenges. Figure 1 shows the architecture of Deep2Q which consists of *two* major modules: phrase query manager and query processor. The *phrase query manager* works offline and is responsible for discovering phrase queries, constructing query evaluation plans, and caching query results. It utilizes a form index to speed up the search for form attributes that match phrase queries and an access path manager to communicate with sources for form submissions and result fetching. Then at query time, the *query processor* takes user queries, finds matching phrase queries, and obtains results from caches or data sources. Figure 2 shows Deep2Q's search interface.

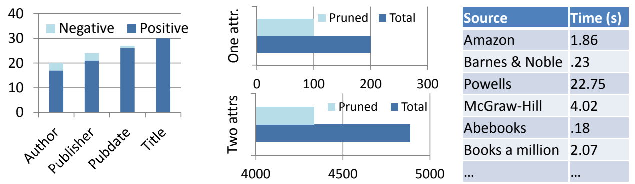
Constructing phrase queries: Deep2Q employs a set of rules for rewriting form queries into phrase queries. For example, a rule may be: if a form attribute A is labeled as "author", create a phrase query "books written by [author]" for every known author. The rules may be discovered from the Web and added to the engine in a *pay-as-you-go* fashion [6]. Since similar attributes (e.g., author) may appear over many forms, this approach allows us to quickly bootstrap the engine with a few popular queries to address the need of a large number of users.

Discovering popular attribute values & form queries: Deep2Q leverages query autocompletion features [2] of web search engines to discover values of form attributes. For example, it may pose an extraction query "books written by" to the engines and extract authors (e.g., "el james") from completions (e.g., "books written by el james").

Deep2Q uses the number of results from sources to gauge the popularity of form queries. It employs an *Apriori*-like algorithm [1] to prune overly general (e.g., returning almost all books from Amazon) and rare queries. The algorithm starts with one-attribute queries and progressively finds frequent queries with multiple attributes.

Dual-ranking: Deep2Q measures the similarity of user keyword queries with phrase queries in the engine using the Cosine function (with $tf * idf$ term weights) by treating phrase queries as short documents. Deep2Q measures the quality of sources with respect to a phrase query based on several *key* factors including the authoritativeness of sources, the number of results, and communication costs.

More details on Deep2Q can be found in [7, 6]. Deep2Q is an important first step toward realizing our vision in [6].



Source	Time (s)
Amazon	1.86
Barnes & Noble	.23
Powells	22.75
McGraw-Hill	4.02
Abebooks	.18
Books a million	2.07
...	...

(a) Discovering attribute values (b) Pruning infrequent form queries (c) Timing result

Figure 3: Results of evaluating Deep2Q

4. EXPERIMENTS

We now present several *key* experimental results [7].

(1) *Discover attribute values from search engines*: Figure 3.a shows the number of positive (e.g., "obama") and negative (e.g., "kids") values discovered for several book attributes (e.g., author) using Yahoo!, Google, and Bing. We observe that the majority of values discovered are positive.

(2) *Discover frequent form queries*: Figure 3.b shows that 50% (80%) of one-attribute (two-attribute) queries on the Amazon query interface are pruned, with thresholds for general and rare queries set to .9 and .001 respectively. Example pruned queries are "romance books in 2011" and "biographies for baby-3 years". Overall, 87% of queries were pruned.

(3) *Speed up query processing*: Results using 10 book sources (Figure 3.c) show that it took about .9s to process a query using cached results in Deep2Q, while existing solution would need up to 22s to retrieve results from sources.

5. CONCLUSIONS & FUTURE WORK

To the best of our knowledge, Deep2Q is the *first* proactive engine for searching the Deep Web using phrase queries. It combines the advantages of existing surfacing (caching for fast response time) and virtual (retrieving fresh data when needed) approaches to integrating the Deep Web [6].

We are currently conducting a large scale of experiments to further evaluate our solution and investigate several *key* issues. (1) To what extent can Deep2Q provide answers to web queries? (2) Can the rewrite rules for form queries easily be adapted to different sources? (3) How to properly manage the caches in Deep2Q? (4) Can we combine Deep2Q with existing search engines? We will also make Deep2Q accessible online to solicit feedback from users.

6. REFERENCES

[1] R. Agrawal et al. Fast algorithms for mining association rules in large databases. In *VLDB*, 1994.

[2] Z. Bar-Yossef and M. Gurevich. Mining search engine query logs via suggestion sampling. *PVLDB*, 2008.

[3] X. Li. Understanding the semantic structure of noun phrase queries. In *ACL*, 2010.

[4] J. Madhavan et al. Web-scale data integration: You can afford to pay as you go. In *CIDR*, 2007.

[5] J. Madhavan et al. Google's deep web crawl. *PVLDB*, 2008.

[6] W. Wu. Proactive natural language search engine: Tapping into structured data on the web. In *EDBT*, 2013.

[7] W. Wu and T. Zhong. A proactive phrase query-based deep web search engine. Technical report, UNCC, 2013.