Structured Data Meets the Web: A Few Observations

Jayant Madhavan, Alon Halevy, Shirley Cohen, Xin (Luna) Dong,
Shawn R. Jeffery, David Ko, Cong Yu
Google, Inc.
jayant@google.com, halevy@google.com, shirleyc@cis.upenn.edu, lunadong@cs.washington.edu,
jeffery@cs.berkeley.edu, dko@google.com, congy@eecs.umich.edu

Abstract

The World Wide Web is witnessing an increase in the amount of structured content – vast heterogeneous collections of structured data are on the rise due to the Deep Web, annotation schemes like Flickr, and sites like Google Base. While this phenomenon is creating an opportunity for structured data management, dealing with heterogeneity on the web-scale presents many new challenges. In this paper we articulate challenges based on our experience with addressing them at Google, and offer some principles for addressing them in a general fashion.

1 Introduction

Since its inception, the World Wide Web has been dominated by unstructured content, and searching the web has primarily been based on techniques from Information Retrieval. Recently, however, we are witnessing an increase both in the amount of structured data on the web and in the diversity of the structures in which these data are stored. The prime example of such data is the *deep web*, referring to content on the web that is stored in databases and served by querying HTML forms. More recent examples of structure are a variety of annotation schemes (e.g., Flickr [4], the ESP game [14], Google Co-op [6]) that enable people to add labels to content (pages and images) on the web, and Google Base [7], a service that allows users to load structured data from any domain they desire into a central repository.

Google is conducting multiple efforts whose common goal is to leverage structured data on the web for better search and to help people create structure that aids search. This paper describes some of our initial observations regarding the problem and some of the challenges entailed by them.

To begin, we look at what kinds of structure can be associated with data on the web. As is typical when managing structured data, there is a trade-off between the investment in creating the structure and the benefit obtained from having the structure. On the Web, the trade-off is a trickier one, because we are trying to appeal to a much larger audience, both to create the structure and to benefit from it. We discuss how the different efforts fall w.r.t. this trade-off.

Next, we consider how structured data is integrated into today's web-search paradigm that is dominated by keyword search. We argue that a critical aspect of any use of structured data on the Web is that it be seamlessly integrated with standard web search. We discuss how our different efforts address this challenge.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

Finally, we pinpoint (what we believe to be) the key reason that makes querying structured data on the web so challenging. Specifically, on the web we model *all* domains of possible interest to users. We cannot make any assumptions about the scope of the domains for which we have structured data, and because of that, creating a *schema* is out of the question. We describe some of the challenges involved in providing access to *data about anything*, and some of our initial steps addressing these challenges.

2 Structured Data On the Web

In this section we describe three kinds of structured data that exist on the Web today. We then discuss the level of structure for each kind of data and how it is currently integrated into Web search.

2.1 The Deep Web

The deep (or invisible) web refers to content that lies hidden behind queryable HTML forms. These are pages that are dynamically created in response to HTML-form submissions, using structured data that lies in backend databases. This content is considered invisible because search-engine crawlers rely on hyperlinks to discover new content. There are very few links that point to deep web pages and crawlers do not have the ability to fill out arbitrary HTML forms. The deep web represents a major gap in the coverage of search engines: the content on the deep web is believed to be possibly more than the current WWW, and typically is of very high-quality [1].

A simple count of the number of forms on the Web would be very misleading if our goal is to understand how much deep web content exists. For instance, there are numerous forms that appear on a large number of web-pages, but direct the user to the same underlying data source. A significant number of forms on the web are simply redirects of site-specific searches to a major search engine. Another large source of forms involves Web 2.0 applications, where users contribute data to online communities. Further, forms are also used for applications such as logins, subscriptions, feedback, etc., none of which yield useful structured data. In [12] we offer some evidence from our explorations at Google that predict the number of deep web sources (that do not fall into the above categories) to be in the tens of millions.

The content on the deep web varies wildly. Many of the sources have information that is geographically specific, such as locators for chain stores, businesses, and local services (e.g., doctors, lawyers, architects, schools, tax offices). There are many sources that provide access to reports with statistics and analysis generated by governmental and non-governmental organizations. Of course, many sources offer product search. However, there is a long tail of sources that offer access to a variety of data, such as art collections, public records, photo galleries, bus schedules, etc. In fact, deep web sites can be found under most categories of the ODP directory [13]. Further, sources vary from those offering only keyword queries through a single text box, to those with detailed forms with many select-menus and other refinement options.

2.2 Google Base

The second source of structured data on the web that we discuss, Google Base, is an attempt to enable content owners to upload structured data into Google so it can be searched. The intention of Google Base is that data can be about *anything*. In addition to the mundane (but popular) product data, it also contains data about clinical trials, event announcements, exotic car parts, people profiles, etc. Google indexes this data and supports simple but structured queries over it.

The data model for Google Base data is purposely kept very simple. Users describe the data they upload into Google Base using an *item type* and attribute/value pairs. For example, a classified advertisement for a used Honda Civic has the item type vehicle and attributes such as make = Honda, model = Civic, location = San Francisco, CA, etc. While Google Base recommends popular item types and attribute names, users are free to invent their own. Users, in fact, do often invent their own item types and labels, leading to a very heterogeneous collection of data. The result is that Google Base is a *very large*, *self-describing*, *semi-structured*, *heterogeneous database*. It is self-describing because each item has a corresponding schema (item type and attribute names). It is semi-structured and heterogeneous because of the lack of restrictions on names and values.

Heterogeneity in Google Base occurs at the level of item types, attributes, and attribute values. Less than a year since its launch, Google Base already contains well over 10,000 item types that together contribute almost 100,000 unique schemata (unique set of attributes names associated with some item). There are over 1000 item types that have more than 10 items. In addition to the sheer number of item types, we observe two other important aspects of Google Base data. First, the large number of item types form a specialization hierarchy. For example, an item can alternately be described as a product, a car part, or a high performance car part. Users choose such different item types naturally and this leads to a proliferation of item types. Second, in addition to naturally occurring heterogeneity, we find that heterogeneity occurs in a different, almost malicious way, that is typically not considered in discussions of data heterogeneity – users provide different item-type names or attribute names deliberately in an attempt to improve the ranking of their items. We are thus witnessing a kind of database design by the masses.

To get a feel for the diversity of data at the attribute-level, we looked at the items of type vehicle, and found that there are over 50 attribute names that occur in 10 or more items. While this might seem to be a large number, there is a smaller core set of attributes that occur in a large number of items. This includes common attributes like make, model, location, color, and price. A commonly occurring attribute-level heterogeneity is one of complex attributes. For example, color for some items includes both the internal color and the external color of a vehicle, while for others there are separate attributes.

Finally, we note that Google Base data also displays significant heterogeneity at the level of attribute values. For example, considering the different values for the attribute color of vehicles, we found that there are over 250 different colors with 5 or more items. In addition to more frequent colors like silver, white, and black, there are cars with the colors polished pewter and light almond pearl metallic. An interesting heterogeneity challenge arises in the extrapolation of domain specific similarities for attribute values, e.g., polished pewter is similar to metallic silver, and incorporating these similarities into query processing. We note that the presence of structure (colors of cars in this case) makes it possible to perform such reconciliation which would not be possible for purely unstructured data.

2.3 Annotation Schemes

There is a third class of structured data on the web which is the result of a variety of *annotation schemes*. Annotation schemes enable users to add tags describing underlying content (e.g., photos) to enable better search over the content. The Flickr Service by Yahoo! is a prime example of an annotation service for photo collections. von Ahn [14] took this idea to the next level by showing how mass collaboration can be used to create high-quality annotations of photos.

Recently, Google created the Google Co-op service that enables users to create customized search engines (see http://data.cs.washington.edu/coop/dbresearch/index.html for a search engine developed for the database research community). To create a custom search engine, a user identifies a set of labels or categories (known as *facets* in Google Co-op) that are of interest to the community at hand. For example, in the case of database research, these facets include professor, project, publication, jobs. Web-pages that fit the various categories are manually (or otherwise) annotated, e.g., we would annotate the homepage of a database faculty with facet professor and pages that appear in DBLP with the facet publication. The important point to note is that the annotations are very light-weight – we only annotate which facet the page belongs to and nothing else. These annotations are uploaded in an XML format to Google Co-op. The annotations can be used to either query for web-pages that specify a particular facet, or to refine the results returned for an earlier query. Facets can also be automatically triggered, e.g., if we were to search for data integration publications, the publication facet will be triggered, and the answers would include only the pages annotated as publications.[1] Google Co-op uses a collaboration model to manage annotations contributed by different users – the creator or a customized search engine approves a list of *collaborators*. Only the annotations contributed by collaborators get included in the customized search engine.

2.4 A Spectrum of Structure Levels

Data on the web, and in particular the kinds of data we described above, vary significantly in its level of structure. The trade-offs involved in structuring data are quite obvious: it takes an effort to create the structure (i.e., create schemata, fit the data into the schema, run a database system), but the benefit is more powerful querying capabilities.

On the Web, this trade-off becomes even more critical. First, our goal is to enable a much broader set of people to create structured data (e.g., in Google Base and with annotation schemes), and therefore the process must be very easy. Second, our user base is incredibly diverse, and therefore querying needs to be easy, structured data needs to be seamlessly integrated with unstructured data, and when structured data appears, it must be easy to explain to the user how to interpret it or query it further. Third, the Web offers us some novel points on the structure spectrum that are valuable to explore. We now examine each of the three cases mentioned above.

The deep web: The deep web represents the most structured data on the web. Although many forms offer search over document collections, the majority of forms offer search into data that is stored in back-end databases and the form queries get translated into queries on the database. Hence, the deep web represents a point in the spectrum where the content creators invested most effort.

Ironically, in many cases, the return on investment for deep web content creators is low. In the early days of the web, high quality collections of structured data were very few, and therefore they became well known and thus were able to attract traffic. Today, however, there are too many high-quality sources, but since web-crawlers do not crawl past forms, the content is invisible to the major web-search engines (and hence to many users looking for the data). As a result, these sites get a fraction of the traffic that is relevant to them. [2]

Google Base: Google Base had to take a much more flexible approach to structuring data. As mentioned before, the idea is that any content provider can contribute structured data to Google Base, without having any expertise in data management. There are a few aspects in which Google Base makes it easier to create structure. First, the data model is kept very simple (objects fall into item types and have attributes). Second, content providers are not forced into any schema. They can choose item types and attribute names from what Google offers, but can invent their own. The same applies to values of attributes in the data. Third, the data can be as semi-structured as the content provider wishes. Specifically, a feed to Google Base can provide as few or as many attributes. Much of the information associated with a Google Base entry can be in unstructured form or even in a URL pointed to by the Google Base entry. As we discuss later, this relaxed approach to structure raises challenges for ranking algorithms.

Annotation schemes: Annotation schemes represent the other extreme of structured data on the web. They require the users to provide very minimal structure (specifically, only the annotations). In a sense, it is as if we attach to every piece of content an attribute isAbout and the annotations provide the values for that attribute. Clearly, annotations can be incorrect and heterogeneous, i.e., use multiple words to say the same thing. However, the minimal structure provided is already enough to offer improvements in search experience.

[1] There is a fallback to general web search if there are no results from the customized search engine.

[2] In some cases, sites have learned to materialize some of their content to enable easier access to web-crawlers.

3 Integrating Structured and Unstructured Data

The reality of web search characteristics dictates the following principle to any project that tries to leverage structured data in web search: querying structured data and presenting answers based on structured data must be *seamlessly* integrated into traditional web search. This principle translates to the following constraints:

- Queries will be posed (at least initially) as keywords. Users will not pose complex queries of any form. At best, users will pick refinements (by filling out a form) that might be presented along with the answers to a keyword query.
- Queries will be posed from one main search destination, e.g., the main page of a search engine. Users will find it hard to memorize the existence of specialized search engines, especially ones that they use occasionally. The onus is hence on the general search engine to detect the correct user intention and automatically activate specialized searches.
- Answers from structured data sources need to (as far as possible) appear along-side regular web-search results. Ideally, they should not be distinguished from the other results. While the research community might care about the distinction between structured and un-structured data, [3] the vast majority of search users do no appreciate the distinction and only care if the results meet their requirement.

Hence, a significant challenge for each of the efforts mentioned above is to fully integrate into web search. We now describe the approach that each of the projects take and the challenges they face in pursuing them.

Deep Web: The typical solution promoted by work on web-data integration is based on creating a virtual schema for a particular domain and mappings from the fields of the forms in that domain to the attributes of the virtual schema. At query time, a user fills out a form in the domain of interest and the query is reformulated as queries over all (or a subset of) the forms in that domain. In fact, in specific domains (e.g., travel, jobs) this approach is used to create vertical search engines. For general web search, however, the approach has several limitations that render it inapplicable in our context.

The first limitation is that the number of domains on the web is large, and even precisely defining the boundaries of a domain is often tricky (as we describe in the next section). Hence, it is infeasible to design virtual schemata to provide broad web search on such content.

The second limitation is the amount of information carried in the source descriptions. Although creating the mappings from web-form fields to the virtual schema attributes can be done at scale (e.g., using techniques described in [3, 10, 11]), source descriptions need to be much more detailed in order to be of use here. Especially, with the numbers of queries on a major search engine, it is absolutely critical that we send *only* relevant queries to the deep web sites; otherwise, the high volume of traffic can potentially crash the sites. For example, for a car site, it is important to know the geographical locations of the cars it is advertising, and the distribution of car makes in its database. Even with this additional knowledge, the engine may impose excessive loads on certain web sites.

The third limitation is our reliance on structured queries. Since queries on the web are typically sets of keywords, the first step in the reformulation will be to identify the relevant domain(s) of a query and then mapping the keywords in the query to the fields of the virtual schema for that domain. This is a hard problem that we refer to as *query routing*.

Finally, the virtual approach makes the search engine reliant on the performance of the deep web sources, which typically do not satisfy the latency requirements of a web-search engine.

The above disadvantages led us to consider a *surfacing* approach to deep web content. In this approach, deep web content is surfaced by simulating form submissions, retrieving answer pages, and putting them into the web index. The main advantage of the surfacing approach is the ability to re-use existing indexing technology; no additional indexing structures are necessary. Further, a search is not dependent on the run-time characteristics

[3] In fact, even as database people investigate the issue deeper, it appears that the distinction is fuzzy to them as well.

of the underlying sources because the form submissions can be simulated off-line and fetched by a crawler over time. A deep web source is accessed only when a user selects a web page that can be crawled from that source. Similarly, the query-routing issue is mostly avoided because web search is performed on HTML pages as before. In terms of schemata and source descriptions, the needs of the surfacing approach are lighter. Instead of creating schemata for individual domains and providing detailed source descriptions, it is enough to identify some way of crawling the data by filling values for a subset of the form fields (in a sense, finding some access path to the data). Hence, this approach can be more easily applied to a much broader collection of domains.

Of course, surfacing has its disadvantages, the most significant one is that we lose the semantics associated with the pages we are surfacing by ultimately putting HTML pages into the web index. Still, at least the pages are in the index, and thus can be retrieved for many searches. Other disadvantages are that it is not always possible to enumerate the data values that make sense for a particular form, and it is easy to create too many form submissions that are not relevant to a particular source. For example, trying all possible car models and zip codes at a used-car site can create about 32 million form submissions – a number significantly larger than the number of cars for sale in the United States. Finally, not all deep web sources can be surfaced: sites with robots.txt as well as forms that use the POST method cannot be surfaced.

Google Base: Google Base faces a different integration challenge. Experience has shown that we cannot expect users to come directly to base.google.com to pose queries targeted solely at Google Base. The vast majority of people are unaware of Google Base and do not understand the distinction between it and the Web index. Therefore, it is crucial to access Google Base in response to keyword queries posed on Google.com. This leads to two main challenges:

- Query routing: given a keyword query, decide if there is data relevant in Google Base and map it to a set of relevant item types in Google Base.
- Result ranking: to fully integrate Google Base results we need to rank results across properties. Specifically, for the query "Honda Civic Mountain View, CA", there might be answers from multiple item types in Google Base, from listings of Honda dealers in the bay area (stored in the Google Local database), and from web documents that may be highly ranked w.r.t. the query.

In some cases, the answer offered by Google Base may be a specific entry (or set of entries) in Google Base. But in most cases, the answer consists of a few example entries from Google Base and a form that enables the user to refine the search results. Hence, an additional challenge Google Base needs to address is that of proposing appropriate refinements. Google Base proposes a set of attributes with candidate values in select-menus. For example, choosing vehicle as an item type leads to possible refinements based on the attributes make, model, color, and location. The user can further continue to restrict search results by choosing specific values in the select-menus for the attributes. With each successive user refinement, Google Base recomputes the set of further candidate refinements.

Google Base proposes query refinements by computing histograms on attributes and their values during query time. It chooses the attributes that best help the user refine the current query. For example, if the user is interested in "Honda Civic" and the location is restricted to "Mountain View, CA", then the select-menus values for color change to reflect only the colors of Honda Civics available in the Mountain View area.

Annotation Schemes: The integration problems faced in the context of annotation schemes are somewhat simpler. Typically, the annotations can be used to improve recall and ranking for resources that otherwise have very little meta-data associated with them (e.g., photos, videos).

In the case of Google Co-op, customized search engines can specify query patterns that trigger specific facets as well as provide hints for re-ranking search results. The annotations that any customized search engine specifies are visible only within the context of that search engine. However, as we start seeing more custom search engines, it would be desirable to point users to different engines that might be relevant.

4 A Database of Everything

The utility of structure-aware search engines has already been demonstrated to some extent by the success of specialized search engines in specific domains (e.g., travel, weather and local services). Handling content in a domain-specific way can potentially lead to better ranking and refinement of search results.

However, the challenges described in the earlier sections lead us to the crux of the problem of querying structured data on the Web: data on the Web is about *everything*. If the domain of structured data on the web were well defined, many of the issues we outlined above would disappear (as in the case of successful specialized search engines).

One could argue that the problem is simply that there are many domains (perhaps a few hundred) and we should simply model them one by one. However, the issue is much deeper. Data on the Web encompasses much of human knowledge, and therefore it is not even possible to model all the possible domains. Furthermore, it is not even clear what constitutes a single domain. Facets of human knowledge and information about the world are related in very complex ways, making it nearly impossible to decide where one domain ends and another begins. The only broad attempt to model human knowledge, the Cyc Project [2], has been going on for about two decades and has met with only limited success.

Even if it were possible to build a representation of all the domains that appear on the Web, the representation would be inherently brittle and hard to maintain. It is hard enough to manage heterogeneity in a single domain, imagine trying to do it at such scale. And if that was not bad enough, you need to maintain the representation in over a hundred languages.

Currently, several solutions are being deployed for addressing the needs of each of the existing services. In parallel with addressing these needs, we are developing some general concepts that will enable us to deal with such challenges in a unified way.

The key idea we are pursuing is that of a system that integrates data from multiple sources but is designed to handle *uncertainty at its core*. Specifically, uncertainty in such a system can have several sources:

- mapping keyword queries into structured queries on a structured data feed,
- mappings between the schemas of disparate data sources, and
- uncertainty about the actual data in the system, that may have been extracted from unstructured document using automatic techniques.

In the contexts described above, such a system will enable us to let many structures (i.e., schemas) exist but without the need to completely integrate them as required by today's data integration systems. Instead of necessarily creating mappings between data sources and a virtual schema, we will rely much more heavily on schema clustering. Clustering lets us measure how close two schemas are to each other, without actually having to map each of them to a virtual schema in a particular domain. As such, schemas may belong to many clusters, thereby gracefully handling complex relationships between domains. Keyword queries will be mapped to clusters of schemas, and at that point we will try to apply approximate schema mappings in order to leverage data from multiple sources to answer queries. The system we envision is an instance of a dataspace support platform [5, 9], where we structure the data in the system in a pay-as-you-go fashion. We offer some more detail about such a system in [12].

5 Conclusions

There are staggering amounts of structured data on the Web today, and we must finally address the challenge of leveraging such data and combining it with unstructured data. This paper described three main sources of structured data, the deep web, Google Base and a variety of annotation schemes, and outlined the main challenges involved in querying this data.

From the perspective of data management, we are used to thinking in terms of a dichotomy between structured data and unstructured data, and the Structure Chasm that lies between them [8]. Web-scale heterogeneity, i.e., data in millions of disparate schemata, presents a unique point between the two ends of the chasm. Our goal is to build a data management platform that can be applied to these collections of data, but to do so, we need to change some of the assumptions we typically have when dealing with heterogeneous data. We believe that the most significant of these challenges is building a data management system that can model *all data*, rather than data that in a particular domain, described by a single schema.

References

[1] M. K. Bergman. The Deep Web: Surfacing Hidden Value, 2001. http://www.brightplanet.com/technology/deepweb.asp.

[2] Cycorp, Inc. http://www.cyc.com/cyc/technology/whatiscyc.

[3] A. Doan, P. Domingos, and A. Halevy. Reconciling schemas of disparate data sources: a machine learning approach. In *Proc. of SIGMOD*, 2001.

[4] Flickr. http://www.flickr.com.

[5] M. Franklin, A. Halevy, and D. Maier. From databases to dataspaces: A new abstraction for information management. *Sigmod Record*, 34(4):27–33, 2005.

[6] Google Co-op. http://www.google.com/coop.

[7] Google Base. http://base.google.com.

[8] A. Halevy, O. Etzioni, A. Doan, Z. Ives, J. Madhavan, L. McDowell, and I. Tatarinov. Crossing the structure chasm. In *Proc. of CIDR)*, 2003.

[9] A. Halevy, M. Franklin, and D. Maier. Principles of dataspace systems. In *Proc. of PODS*, 2006.

[10] B. He and K. C.-C. Chang. Statistical schema integration across the deep web. In *Proc. of SIGMOD*, 2003.

[11] J. Madhavan, P. A. Bernstein, A. Doan, and A. Y. Halevy. Corpus-based schema matching. In *Proc. of ICDE*, pages 57–68, 2005.

[12] J. Madhavan, S. Cohen, X. L. Dong, A. Y. Halevy, S. R. Jeffery, D. Ko, and C. Yu. Navigating the seas of structured web data. In *Proc. CIDR*, 2007.

[13] ODP – Open Directory Project. http://dmoz.org.

[14] L. von Ahn and L. Dabbish. Labeling Images with a Computer Game. In *Proc. of ACM CHI*, 2004.