Chapter 10: Web Data

Introduced by Peter Bailis

Selected Readings:

Sergey Brin and Larry Page. The Anatomy of a Large-scale Hypertextual Web Search Engine. *WWW*, 1998.

Eric A. Brewer. Combining Systems and Databases: A Search Engine Retrospective. *Readings in Database Systems, Fourth Edition*, 2005.

Michael J. Cafarella, Alon Halevy, Daisy Zhe Wang, Eugene Wu, Yang Zhang. WebTables: Exploring the Power of Tables on the Web. *VLDB*, 2008.

Since the previous edition of this collection, the World Wide Web has unequivocally laid any lingering questions regarding its longevity and global impact to rest. Several multi-Billion-user services including Google and Facebook have become central to modern life in the first world, while Internet- and Web-related technology has permeated both business and personal interactions. The Web is undoubtedly here to stayat least for the foreseeable future.

Web data systems bring a new set of challenges, including high scale, data heterogeneity, and a complex and evolving set of user interaction modes. Classical relational database system designs did not have the Web workload in mind, and are not the technology of choice in this context. Rather, Web data management requires a melange of techniques spanning information retrieval, database internals, data integration, and distributed systems. In this section, we include three papers that highlight technical solutions to problems inherent in Web data management.

Our first two papers describe the internals of search engine and indexing technology. Our first paper, from Larry Page and Sergey Brin, Google co-founders, describes the internals of an early prototype of Google. The paper is interesting both from a historical perspective as well as a technical one. The first Web indices, such as Yahoo!, consisted of human-curated ”directories”. While directory curation proved useful, directories were difficult to scale and required considerable human power to maintain. As a result, a number of search engines, including Google but also Inktomi, co-created by Eric Brewer, author of the second paper, sought automated approaches. The design of these engines is conceptually straightforward: a set of crawlers downloads copies of web data and builds (and maintains) read-only indices that are used to compute a relevance scoring function. Queries, in turn, are serviced by a front-end web service that reads from the indices and presents an ordered set of results, ranked by scoring function.

The implementation and realization of these engines is complex. For example, scoring algorithms are highly tuned, and their implementation is considered a trade secret even within search engines today: Web authors have a large incentive to manipulate the scoring function to their advantage. The PageRank algorithm described in the Google paper (and detailed in [5]) is an famous example of a scoring function, and measures the ”influence” of each page measured according to the hyperlink graph. Both papers describe how a combination of mostly unspecified attributes is used for scoring in practice, including ”anchor text” (providing context on the source of a link) and other forms of metadata. The algorithmic foundations of these techniques, such as keyword indexing date, date to the 1950s [4], while others, such as TFxIDF ranking and inverted indices, date to the 1960s [6]. Many of the key systems innovations in building Internet search engines came in scaling them and in handling dirty, heterogenous data sources.

While the high-level details of these papers are helpful in understanding how modern search engines operate, these papers are also interesting for their commentary on the process of building a production Web search engine. A central message in each is that Web services must account for variety; the Google authors describe how assumptions made in typical information retrieval techniques may no longer hold in a Web context (e.g., the ”Bill Clinton sucks” web page). Web sources change at varying rates, necessitating prioritized crawling for maintaining fresh indices. Brewer also highlights the importance of fault tolerance and availability of operation, echoing his experience in the field building Inktomi (which also led to the development of concepts including *harvest* and *yield* [2] and the CAP Theorem; see Chapter 7). Brewer outlines the difficulty in building a search engine using commodity database engines (e.g., Informix was 10x slower than Inktomi’s custom solution). However, he notes that the principles of database system design, including ”top-down” design, data independence, and a declarative query engine, are valuable in this context—if appropriately adapted.

Today, Web search engines are considered mature technology. However, competing services continually improve search experience by adding additional functionality. Today's search engines are much more than information retrieval engines for textual data web pages; the content of the first two papers is a small subset of the internals of a service like Google or Baidu. These services provide a range of functionality, including targeted advertisement, image search, navigation, shopping, and mobile search. There is undoubtedly bleed-over in retrieval, entity resolution, and indexing techniques between these domains, but each requires domain-specific adaptation.

As an example of a new type of search enabled by massive Web data, we include a paper from the WebTables project led by Alon Halevy at Google. WebTables allows users to query and understand relationships between data stored in HTML tables. HTML tables are inherently varied in structure due to a lack of fixed schema. However, aggregating enough of them at Web scale and performing some lightweight automated data integration enables some interesting queries (e.g., a table of influenza outbreak locations can be combined with a table containing data about city populations). Mining the schema of these tables, determining their structure and veracity (e.g., only 1% of the tables in the paper corpus were, in fact, relations), and efficiently inferring their relationships is difficult. The paper we have included describes techniques for building an attribute correlation statistics database (AcsDB) to answer queries about the table metadata, enabling novel functionality including schema auto-complete. The WebTables project continues today in various forms, including Google Table Search and integration with Google's core search technology; an update on the project can be found in [1]. The ability to produce structured search results is desirable in several non-traditional domains, including mobile, contextual, and audio-based search.

The WebTables paper in particular highlights the power of working with Web data at scale. In a 2009 article, Halevy and colleagues describe the "Unreasonable Effectiveness of Data," effectively arguing that, with sufficient amount of data, enough latent structure is captured to make modeling simpler: relatively simple data mining techniques often beat more mathematically sophisticated statistical models [3]. This argument stresses the potential for unlocking hidden structure by sheer volume of data and computation, whether mining schema correlations or performing machine translation between languages. With a big enough haystack, needles become large. Even examining 1% of the tables in the web corpus, the VLDB 2009 paper studies 154M distinct relations, a corpus that was "five orders of magnitude larger than the largest one [previously] considered."

The barrier for performing analysis of massive datasets and system architectures outside of these companies is decreasing, due to cheap commodity storage and cloud computing resources. However, it is difficult to replicate the feedback loop between users (e.g., spammers) and algorithms (e.g., search ranking algorithms). Internet companies are uniquely positioned to pioneer systems designs that account for this feedback loop. As database technologies power additional interactive domains, we believe this paradigm will become even more important. That is, the database market and interesting database workloads may benefit from similar analyses. For example, it would be interesting to perform a similar analysis on hosted database platforms such as Amazon Redshift and Microsoft SQL Azure, enabling a variety of functionality including index auto-tuning, adaptive query optimization, schema discovery from unstructured data, query autocomplete, and visualization recommendations.

References

[1] S. Balakrishnan, A. Halevy, B. Harb, H. Lee, J. Madhavan, A. Rostamizadeh, W. Shen, K. Wilder, F. Wu, and C. Yu. Applying webtables in practice. In *CIDR*, 2015.

[2] E. Brewer et al. Lessons from giant-scale services. *Internet Computing, IEEE*, 5(4):46–55, 2001.

[3] A. Halevy, P. Norvig, and F. Pereira. The unreasonable effectiveness of data. *IEEE Intelligent Systems*, 24(2):8–12, Mar. 2009.

[4] H. P. Luhn. Auto-encoding of documents for information retrieval systems. *Modern Trends in Documentation*, pages 45–58, 1959.

[5] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: bringing order to the web. Technical report, Stanford InfoLab, 1999. SIDL-WP-1999-0120.

[6] G. Salton and M. E. Lesk. Computer evaluation of indexing and text processing. *Journal of the ACM (JACM)*, 15(1):8–36, 1968.