WEB SEARCH

Alexandros Ntoulas
Microsoft Research
1065 La Avenida
Mountain View CA 94043, USA
antoulas@microsoft.com

Michalis Vazirgiannis
Dept. of Informatics, Athens Univ. of Economics & Business
Patision 76, 10434, Athens
Greece
mvazirg@aueb.gr

1. INTRODUCTION

The Web today plays a central part in the cultural, educational and commercial life of millions of users. Due to the astonishing amount of information available on the Web, users typically rely on the Web search engines in order to locate relevant and useful information. A Web search engine's task is to find the most relevant content on the Web, given a user's query. To achieve this, different search engines may follow different approaches, but, in general, major search engines such as Google [14], Live Search [19] and Yahoo! [37] follow the architecture outlined in Figure 1.

The *crawlers* are programs that "browse" the Web by following links in a manner similar to the way human users visit different pages by following links. Their job is to download the pages from the Web and store them locally in a *page repository* database. From there, every page is processed and indexed. During the *analysis of its content* the search engine records various useful metadata for each Web page. Such metadata may include the set of outgoing links from a given Web page (which are also fed back to the crawler to download more content), the geo-location of every page, the graph structure around a given page etc. The metadata recorded for every page may also include information from external sources (e.g. the search engine query logs), such as the number of times a page has been clicked or viewed. The *indexer module* extracts all the words from each page and generates a data structure capable of answering very quickly which Web pages contain a set of given words. The *query processing module* is responsible for receiving a user's query and identifying the relevant pages by consulting the metadata, the index and the page repository. Finally, once all relevant results are identified, the *ranking module* sorts them so that the results that the user is most likely interested in appear on top, and presents them to the user.

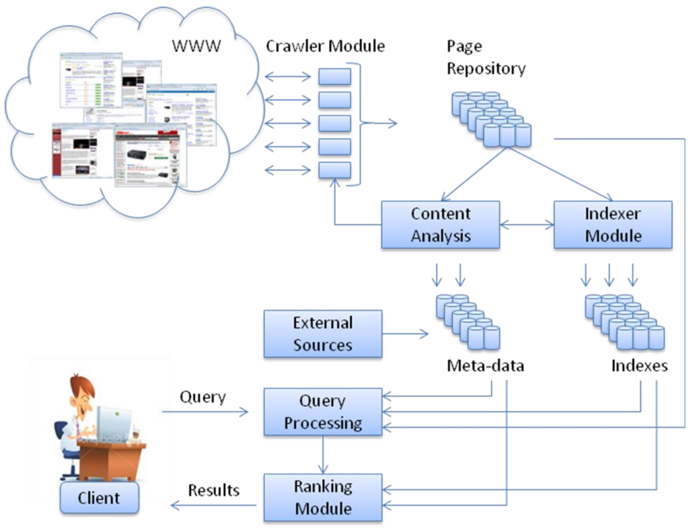

Figure 1. The architecture of a Web search engine

During this process, the most important pieces are the crawler, indexing and ranking modules. If the crawler cannot retrieve the right content from the Web, the search engine will not have an appropriate answer for some of the user's queries. In addition, since the Web is highly dynamic and evolving, the crawlers need to ensure that the search engine always has fresh information in order to serve the users in the best way. The indexing module needs to be capable of answering very quickly and efficiently which pages contain a given set of keywords coming from the user's query. Given the size of the Web, this task can prove to be very costly and challenging. Finally, if the ranking module cannot surface the right pages to the top of the results the users will be dissatisfied and may turn away from the search engine.

In this chapter we present some of the challenges in the area of Web search and an overview of the research performed by Greek researchers to address these challenges. We start by presenting previous work on the crawling, indexing and ranking areas in the next three sections. We discuss approaches on distributed and peer-to-peer search in Section 5, and we present existing Web search systems in Section 6. We conclude in Section 7.

2. CRAWLING WEB PAGES

Typically, the crawler of a search engine operates as follows: given an initial set of URLs, the crawler downloads their content and stores it locally in the page repository for later processing. Next, the crawler proceeds iteratively (usually in a breadth-first manner) following all the outgoing links from the already downloaded set of URLs in order to retrieve more content. This process is repeated until all resources (in terms of, e.g., time, bandwidth or storage space) are exhausted.

Since the crawler operates within some resource constraints, it is of paramount importance to select pages to download that are of good quality and of interest to many users. If the crawler downloads pages that are not of good quality the search engine will be unable to return the expected results to the users. To this extend, search engines may choose to crawl a page based on its popularity (captured for example by the number of incoming links), based on the distance of the page from the Web site's root page, based on the expected topic of the page, etc.

A large and important part of the Web, however, is not directly accessible through links but rather is hidden within text databases with content covering a multitude of topics (see [16] for automatically detecting the topic of a text database). This content is oftentimes of great value to the users since it contains well-organized and well-edited information. Unfortunately, the crawler cannot find direct links to such pages (which are collectively called Hidden or Deep Web) because such links are only available after a user types a query in a search interface. Examples of Hidden Web sites are PubMed [28], or the US Patent and Trademark Office [34]. In [25] the authors present algorithms for generating queries in order to download the quality content from the Hidden Web. In certain cases, their algorithm was able to download more than 95% of a Hidden-Web site after issuing about 100 queries. For the cases where a Web site provides access to its content both through links and through a search interface, the authors in [15] present a rigorous framework of estimating the cost of each approach (link-based or query-based) and deciding which one to apply.

Once the crawler has downloaded a set of pages from the Web, it needs to periodically revisit them in order to detect changes and refresh the local repository. Coping with the dynamic nature of the Web is very critical to search engine crawlers. Unless they are able to capture the latest version of the data on the Web the search engines will present the users with obsolete results. In [25] the authors show that 50% of the pages on the Web do not change over the period of one year. Out of the remaining 50% that change the authors found that previous changes are good predictors for future changes [25]. Given these results, the crawlers should mainly focus on refreshing the previously changed pages in order to make good use of their download resources.

In [11] the authors present a sampling-based technique for downloading as many changed pages as possible from the Web. The main idea is to sample a few pages from every Web site and then download more pages based on how many changes where detected in each sample. In a similar line of work, in [17] the authors use survival analysis techniques to model the changes in text databases (often supporting Hidden-Web sites) and determine which ones to update. In [3], the authors compute a set of features that characterize behaviour of a set of Web sites based on their national domain (including the Greek domain). Such features can prove very useful in refreshing Web pages.

3. INDEXING

Once the crawler downloads and creates a local copy of a set of pages from the Web, the search engine then proceeds to analyze and index these pages. During this phase, all the links, words and HTML tags are extracted from every page and a set of indexes is generated. Such indexes may contain, for example, the link structure around a given page which is useful during ranking as we will discuss in Section 4, the set of pages within a given domain, etc. The most important index that a search engine builds during this phase is the one that helps identify which pages contain the user query words. Given the enormous size of the Web, the number of queries served by a search engine every day and the user expectations (users expect search results in less than 1 second), this textual index has to be highly scalable and highly efficient.

The Web search engines, typically utilize a data structure called *inverted index* in order to locate the documents relevant to a user's query very quickly. If we consider a collection of pages $D = \{D_1, ..., D_M\}$ that the crawler has downloaded from the Web and the set of all words $T = \{t_1, ..., t_n\}$ in this collection, the search engine maintains a list $I(t_i)$ of page IDs that contain t_i. Every entry in $I(t_i)$ is called a posting and can be extended to include additional information, such as how many times t_i appears in a Web page, the positions of t_i in the Web pages, whether t_i appears in bold/italic etc. The set of all the lists $I = \{I(t_1), ..., I(t_n)\}$ is our inverted index. Whenever a query comes in, each query word is matched against the corresponding inverted list in order to identify the set of relevant pages.

Search engines are accepting millions of queries every day from eager users searching for information. In order to cope with this huge query load, one way is to replicate fully the inverted index across a large cluster of machines, but given the size of the Web, this approach can prove very costly. To alleviate this problem, search engines sometimes use a small cache where they put frequently accessed pages. The idea is that the cache contains the desired result most of the time, and in this way the fully replicated indexes are accessed less. In this approach however, it is crucial for the search engine to select the right pages to put in the cache.

In [4] the authors explore the tradeoffs between static and dynamic caching as well as whether the search engines should cache based on the query results or the inverted lists of the index. They showed that dynamic caching, in general, has limited effectiveness and they propose new static caching algorithms that demonstrate very good performance. In addition, they showed that caching query results is more preferable in cases where the network communication time between machines is big and they study how the effectiveness of caching changes over time given that the observed query load evolves.

In [24], the authors present a replication and caching scheme appropriate for inverted indexes that has the nice property that it can guarantee the results of the scheme to be identical to a scheme that only uses replication, but this is done by using about 50% less space. The paper discusses methods for selecting which postings to put in each tier, one by selecting inverted lists and one by selecting individual postings. It also discusses how we can determine the optimal size of the cache in order to minimize the storage space required without sacrificing performance in terms of speed or result relevance.

For the cases where computing the complete answer from an inverted index is either prohibitively expensive or not required, [2] presents a method for sampling the search results and compute a summary from an inverted index. This method can be useful in identifying the representative topics within the search results without paying the price to calculate the full answer, for providing feedback to the user of what to expect before the full answer is computed, or even for estimating the total number of search results.

4. RANKING OF RESULTS

Once the search engine determines the relevant pages it needs to present them to the user. During this process the results are ranked in terms of relevance to the user's query so that the most relevant appears on top, followed by the second most relevant and so on. In order to determine the relevance of a page to the user's query the search engine, at a high level, depends mainly on two factors: a) how close the content of the page is to the query (textual relevance); b) how "popular" or "important" the page is overall on the Web (page relevance). For the textual relevance, the most widely used method is the Okapi BM25 [30].

The overall "popularity" or "importance" of Web page (page relevance), is usually captured by link-based ranking algorithms, such as PageRank [28] which is currently implemented within Google. The main idea of PageRank is to consider that a page is important if it is linked by other important pages. Since the first appearance of PageRank, there have been several variations of link-based ranking algorithms which have shown their own merits and shortcomings in different settings. In [6, 9, 33] the authors present a theoretical framework that can be applied in the studying and analysis of link-based ranking algorithms. The authors discuss the properties and differences of a number of link-based ranking algorithms and they present comparative studies of their performance. In [1], the authors present extensions to another popular link-based ranking algorithm called HITS [18] by using multiple eigenvectors instead of a single one.

Computing the link-based ranking for a set of pages typically involves extracting the graph structure from these pages and then performing iterative computations over this graph. Given the enormous size of the Web and its dynamic nature, such computations may become prohibitely expensive to be done on a regular basis. To this extend, the authors in [37] study the rank changes within the Web graph. In [36] the authors present a method for predicting the ranking of pages in the future based on their ranking in the past. The method is based on Markov model learning and was shown capable of predicting the ranks with accuracy of 90% on real-Web datasets. The method consists of the following phases: (a) computing the ranking trends from the rank evolution of individual pages (b) computing of the states of a Markov model based on an equi-probable partitioning and (c) using the resulting Markov models for predicting the next ranking of a page whose ranking history is known at the given point in time. The authors of [36] are also working on alternative prediction methods such as regression and spectral pre-processing techniques.

Given the variety of interests and needs that the users around the world have while performing Web searches, several search engines have tried to personalize their search results to the individual users. Personalization normally involves a phase where the profile of a user is first identified before it can be applied during the ranking phase in order to boost results that are likely to be closer to the user's interests. In [31] the authors consider clicked pages to be a good representative of a user's profile and they use a topical taxonomy to categorize the incoming queries and pages and personalize the search results. In [13] the authors discuss an approach of performing personalization of the results through categorization on the client side, i.e. after the search engine has computed the results.

Finally, given the high potential monetary value of the search traffic, some Web site operators pollute the Web with spam pages (i.e. pages meaningless to the users but existing only to trick search engines), in the hope of improving their ranking. In [26], the authors present a number of features that can capture whether a Web page is spam or not. Features such as number of words in the title, fractions of links in the page, and redundancy of the page's content show high promise and when combined they can help a search engine detect about 87% of the spam with 91% accuracy. In [10] the authors use the query logs to identify potential spam pages as well as query words that are very likely targeted by spammers. Their method yields an accuracy of about 78%. The work in [5] discusses spam removal and ranking for the blogosphere.

5. PEER-TO-PEER (P2P) APPROACHES TO WEB SEARCH

In our discussion so far, we have assumed that the Web search experience of the users is fully controlled by a search engine which builds a centralized index where the information is stored, processed and retrieved at a user's whim. In this centralized approach the search engines can monopolize the information flow, while the content owners on the Web have no control over how their information is processed, stored, ranked and presented to the users. In addition, although the Web content is already stored in an enormous number of servers across the Web, it has to be downloaded by the search engine crawlers before the users can actually access it, thus introducing issues of freshness and coverage of the returned results. In order to take advantage of these observations and the need of the content providers to have more control over their content, researchers have worked on creating peer-to-peer (P2P) search systems.

In a P2P search engine each peer contains parts of the overall content available in the system. Once a user issues a query to her local machine, the query is routed to the appropriate peers, the results are collected and they are presented to the users. There are of course several challenges in P2P Web searching, ranging from how each peer organizes its local index to ensuring that all the information is available at the network at any one time. One of the most important challenges though, is the problem of *query routing*. Given the size of information and the potentially enormous number of peers available, the system cannot possibly contact each and every peer and retrieve a response within a reasonable amount of time, but instead it needs to identify which peers are most likely to contain relevant information and *route* the user's query only to those peers.

A good overview of the different approaches in organizing a peer-to-peer Web search engine is presented in [20]. The authors investigate a number of design challenges, mostly regarding partitioning schemes for the index and data storage technologies. They report on the pros and cons and the cost/performance ratios of each presented approach. Overall, a design strategy (named Anakin) that employs term partitioning for the index and a combined storage of hard disk and flash-ram is described as the most promising.

A system for organizing Web content in a distributed and decentralized way is presented in [12]. The main contribution of this work is the creation of distributed semantic overlay networks that are used for query routing in a semi structured P2P Web content architecture. In order to achieve the desired performance and scalability, Semantic Overlay Networks (SONs) connecting peers storing semantically related information are employed. The lack of global content/topology knowledge in a P2P system is the key challenge in forming SONs. The authors in [12] describe an unsupervised approach for decentralized and distributed generation of SONs (DESENT). Through simulations and analytical cost modeling they verify the claims regarding performance, scalability, and quality.

In [6] the authors discuss two methods of improving query routing in a P2P Web search setting. The first one is based on single-keyword statistics collected in each peer, while the second one is based on multi-keyword statistics. These statistics are represented in terms of hash-sketch synopses and they are used to build routing indices that help a peer determine which of the peers are likely to be the best ones to contact. In particular, the work in [6] aims at exploiting potential correlations among query keywords in order to improve the overall query routing within the Minerva system [7].

6. EXISTING WEB SEARCH SYSTEMS

In this section we briefly discuss some of the existing Web search systems that are based on the work presented in the previous sections.

Blogscope [5] is a system that facilitates the online analysis of contents from the blogosphere. Blogscope currently tracks and indexes some 10 million blogs and is capable of handling several thousands of updates in its collection every day. Blogscope allows for spatio-temporal analysis of blogs, detection of information busts, identification of correlated keywords and a ranking function applied to blogs. The system is currently available at http://www.blogscope.net.

Infocious [23] is a fully-functional Web search engine that is currently indexing more than 2.5 billion Web pages. It implements variations of the technologies presented in [11,20,24,27] and it applies Natural Language Analysis to the downloaded pages in order to understand their content in a better way. It allows the users to perform advanced searches (such as specifying whether a query keyword should be a verb or a noun), and it presents the results topically categorized in the Dmoz hierarchy (similar to [30]). This enables the user to drill-down in a category of interest to retrieve more topically-focused results and it provides the basis for the personalization of results. Infocious is publicly available at http://search.infocious.com.

Minerva [6,20] is a distributed search engine capable of handling a large set of autonomous peers. Within Minerva, each peer maintains a local collection of pages which can be either independently crawled from the Web or imported from external sources. Each peer maintains a local inverted index in order to answer queries quickly and efficiently, with the words stored within the index being stemmed for increased recall. The sets of terms are partitioned among peers with each peer maintaining statistics for every word that it stores. Query routing is performed by exploiting keyword and attribute-value correlations similar to [6]. The project is currently available at http://www.minerva-project.org.

A system called **THESUS**, that supports retrieval of Web content relevant to an ontology is presented in [35]. This line of work puts emphasis on the use of a page's incoming links as means of classification. The authors present the tools needed in order to manage the links, and their semantics. They further process these links using a hierarchy of concepts, akin to an ontology, and a thesaurus. The work results in prototype system, called THESUS that organizes thematic Web documents into semantic clusters. The main contributions of this effort are: (a) a model and language to exploit the information within link semantics (b) the THESUS prototype system, (c) its innovative aspects and algorithms, and in particular the novel similarity measure between Web documents that can be applied to different clustering schemes (DB-Scan and COBWEB).

7. CLOSURE AND REMARKS

Web Search is a cornerstone of today's industry and economy. Millions of users perform searches on the Web in an attempt to locate useful information for their everyday lives. In the last few years, Web search is starting to bear a strong social aspect as well, especially for collaborative tasks such as booking a trip or arranging for vacation. Greek scientists are remarkably active and successful in this competitive field of research covering the aspects of this process ranging from the ranking of results to P2P architectures. Here, we have presented an overview of the challenges in Web search and the work done by Greek researchers. Our goal is to give a high-level summary of the past and ongoing work in this area and not provide an exhaustive list. It is our hope that the covered material will stimulate further intriguing research and fruitful discussions so that work in this area will continue with more exciting and ground-breaking results.

8. REFERENCES

1. D. Achlioptas, A. Fiat, A. Karlin, F. McSherry, *Web search through hub synthesis*, In Proceedings of the 42nd Foundation of Computer Science (FOCS). Las Vegas, NV, 2001.
2. A. Anagnostopoulos, A. Z. Broder, D. Carmel, *Sampling Search-Engine Results,* World Wide Web Journal, Volume 9, Number 4, pp. 397-429, 2006.
3. R. Baeza-Yates, C. Castillo, E. Efthimiadis, *Characterization of National Web Domains. ACM Transactions on Internet Technology*, 7(2), Article 9, 2007.
4. R. Baeza-Yates, A. Gionis, F. Junqueira, V. Murdock, V. Plachouras; F. Silvestri, *The impact of caching on search engines,* In Proceedings of the ACM International Information Retrieval (SIGIR) Conference, 2007, Amsterdam, Netherlands.
5. N. Bansal, N. Koudas, *Searching the Blogosphere,* In Proceedings of WebDB, 2007.
6. M. Bender, S. Michel, Nikos Ntarmos, P. Triantafillou, G. Weikum, C. Zimmer, *Discovering and Exploiting Keyword and Attribute-Value Co-occurrences to Improve P2P routing Indices*, ACM International Conference on Information and Knowledge Management (CIKM), 2006.
7. M. Bender, S. Michel, P. Triantafillou, G. Weikum, C. Zimmer, *MINERVA: Collaborative P2P Web Search*, International Conference on Very Large Data Bases (VLDB) (demo paper), September 2005.
8. A. Borodin, J. S. Rosenthal, G. O. Roberts, P. Tsaparas, *Finding Authorities and Hubs From Link Structures on the World Wide Web*, World Wide Web Conference (WWW), Hong Kong, 2001.
9. A. Borodin, J. S. Rosenthal, G. O. Roberts, P. Tsaparas, *Link Analysis Ranking: Algorithms, Theory and Experimets*, ACM Transactions on Internet Technologies (TOIT), Vol 5, No 1, February 2005.
10. C. Castillo, C. Corsi, D. Donato, P. Ferragina, A. Gionis, *Query-log mining for detecting spam*, Fourth International Workshop on Adversarial Information Retrieval on the Web, 2008.
11. J. Cho, A. Ntoulas, *Effective Change Detection using Sampling*, In Proceedings of the International Conference on Very Large Databases (VLDB), 2002, Hong Kong, China.
12. C. Doulkeridis, K. Norvag, M. Vazirgiannis, *DESENT: Decentralized and distributed semantic overlay generation in P2P networks,* In Special Issue on Peer-to-Peer Communications and Applications, IEEE Journal on Selected Areas in Communications (J-SAC), Vol. 25, Issue 1, pages 25-34, January 2007
13. J. Garofalakis, T. Matsoukas, Y. Panagis, E. Sakkopoulos, A. Tsakalidis, *Personalization Techniques for Web Search Results Categorization*, EEE 2005: 148-151
14. Google Inc. http://www.google.com
15. P. Ipeirotis, E. Agichtein, P. Jain, and L. Gravano. *To Search or to Crawl? Towards a Query Optimizer for Text-Centric Tasks,* In Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data (SIGMOD), 2006.
16. P. Ipeirotis, L. Gravano, *Distributed Search over the Hidden-Web: Hierarchical Database Sampling and Selection*, In Proceedings of the 28th Intenational Conference on Very Large Databases (VLDB), 2002
17. P. Ipeirotis, A. Ntoulas, J. Cho, L. Gravano, *Modeling and Managing Content Changes in Text Databases*, ACM Transactions on Database Systems (TODS), vol. 32, no. 3, September 2007.
18. J. Kleinberg, *Authoritative sources in a hyperlinked environment*, In Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, 1998.
19. Live Search. http://www.live.com
20. S. Michel, M. Bender, P. Triantafillou, G. Weikum, *Design Alternatives for Large-Scale Web Search: Alexander was Great, Aeneas a Pioneer, and Anakin has the Force*, In Workshop on Large scale Distributed Systems for Information Retrieval, Collocated with SIGIR , July 2007.
21. S. Michel, P. Triantafillou, G. Weikum, *MINERVA∞: A Scalable Efficient Peer-to-Peer Search Engine,* ACM/IFIP/USENIX 6th International Middleware Conference, November 2005.
22. A. Ntoulas, *Crawling and Searching the Hidden Web*, Ph.D. thesis, University of California Los Angeles, 2006.
23. A. Ntoulas, G. Chao, J. Cho, *The Infocious Web Search Engine: Improving Web Searching through Linguistic Analysis*, Journal of Digital Information Management (JDIM) vol. 5, no 5, October 2007.
24. A. Ntoulas, J. Cho, *Pruning Policies for Two-Tiered Inverted Index with Correctness Guarantee*, In Proceedings of the ACM International Information Retrieval (SIGIR) Conference, 2007, Amsterdam, Netherlands.
25. A. Ntoulas, J. Cho, C. Olston, *What's New on the Web? The Evolution of the Web from a Search Engine Perspective,* In Proceedings of the World Wide Web (WWW) Conference, 2004, New York, USA.

26. A. Ntoulas, M. Najork, M. Manasse, D. Fetterly, *Detecting Spam Web Pages through Content Analysis*, In Proceedings of the World Wide Web (WWW) Conference, 2006, Edinburgh, Scotland.
27. A. Ntoulas, P. Zerfos, J. Cho. *Downloading Textual Hidden Web Content through Keyword Queries*, In Proceedings of the Joint Conference on Digital Libraries (JCDL), 2005, Denver, USA.
28. L. Page, S. Brin, R. Motwani, T. Winograd, *The PageRank citation ranking: Bringing order to the Web*, Technical report Stanford Digital Library Technologies Project, 1998.
29. PubMed Medical Library. http://www.pubmed.org
30. K. Sparck Jones, S. Walker, S.E. Robertson, *A probabilistic model of information retrieval: development and comparative experiments*, Information Processing and Management 36, 2000.
31. S. Stamou, A. Ntoulas, *Search Personalization through Query and Page Topical Analysis*, Journal of User Modeling and User-Adapted Interaction (UMAI), 2008.
32. S. Stamou, A. Ntoulas, V. Krikos, P. Kokosis, D. Christodoulakis. *Classifying Web Data in Directory Structures*, In Proceedings of the 8th Asia Pacific Web Conference (APWeb), Harbin, China, January, 2006.
33. P. Tsaparas, *Link analysis ranking*, Ph.D. thesis, University of Toronto, 2004.
34. United States Patent and Trademark Office. http://www.uspto.gov
35. I. Varlamis, M. Vazirgiannis, M. Halkidi, B. Nguyen, *THESUS: Effective Thematic Selection And Organization Of Web Document Collections Based On Link Semantics*, in IEEE Transactions on Knowledge and Data Engineering Journal, June 2004 (Vol. 16, No. 6), pp. 585-600.
36. M. Vazirgiannis, D. Drosos, P. Senellart, A. Vlachou, *Web Page Rank Prediction with Markov Models*, poster, WWW 2008, Beijing, China, April 2008
37. A. Vlachou, M. Vazirgiannis, K. Berberich, *Representing and quantifying rank - change for the Web Graph*, Fourth Workshop on Algorithms and Models for the Web-Graph (WAW2006), Banff, Canada, November 2006
38. Yahoo! Inc. http://search.yahoo.com