W

W3C

Serguei Mankovskii
CA Labs, CA, Inc., Thornhill, ON, Canada

Synonyms
World Wide Web consortium

Definition
W3C is an international consortium for development of World Wide Web protocols and guidelines to ensure long-term growth of the Web.

Key Points
W3C was founded in 1994 by the inventor of the World Wide Web Tim Berners-Lee as a vendor-neutral forum for building consensus around Web technologies. The consortium consists of member organization and dedicated staff of technical experts. Membership is open to any organization or individual whose application is reviewed and approved by the W3C. Usually W3C members invest significant resources into the Web technologies.

W3C fulfils its mission by creation of recommendations enjoying status of international standards. In the first 10 years of existence, it produced over eighty W3C recommendations. W3C is responsible for such technologies as HTML, XHTML, XML, XML Schema, CSS, SOAP, WSDL and others. W3C members play a leading role in the development of the recommendations.

W3C initiatives involve international, national, and regional organizations on global scale. Global participation reflects broad adoption of the web technologies. Along with broad adoption comes growth in diversity of software and hardware used on the Web. W3C aims to facilitate hardware and software interoperability to avoid fragmentation of the Web.

W3C operations are jointly administered by the MIT Computer Science and Artificial Intelligence Laboratory, European Research Consortium for Informatics and Mathematics, and Keio University.

Cross-references
▶ Web 2.0/3.0
▶ Web Services
▶ XML
▶ XML Schema

Recommended Reading
1. W3C. Available at: http://www.w3.org/

W3C XML Path Language

▶ XPath/XQuery

W3C XML Query Language

▶ XPath/XQuery

W3C XML Schema

▶ XML Schema

WAN Data Replication

Maarten van Steen
VU University, Amsterdam, The Netherlands

Synonyms
Wide-area data replication

Definition
The field of WAN data replication covers the problems and solutions for distributing and replicating data across wide-area networks. Concentrating on databases alone, a wide-area database is defined as a collection of multiple, logically interrelated databases distributed and

possibly replicated across sites that are connected through a wide-area network.

The characteristic feature is that data are spread across sites that are separated through wide-area links. Unlike links in local-area networks, the quality of communication through wide-area links is relatively poor. Links are subject to latencies of tens to thousands of milliseconds, there are often severe bandwidth restrictions, and connections between sites are much less reliable.

In principle, WAN data replication also covers the distribution and replication of plain files. These issues are traditionally handled by wide-area distributed file systems such as AFS [11] and NFS [3,12], which are both widely used. These distributed file systems aim at shielding data distribution from applications, i.e., they aim at providing a high degree of distribution transparency. As such, they tackle the same problems that wide-area databases need to solve. However, matters are complicated for databases, because relations *between* and *within* files (i.e., tables) also need to be taken into account.

Historical Background

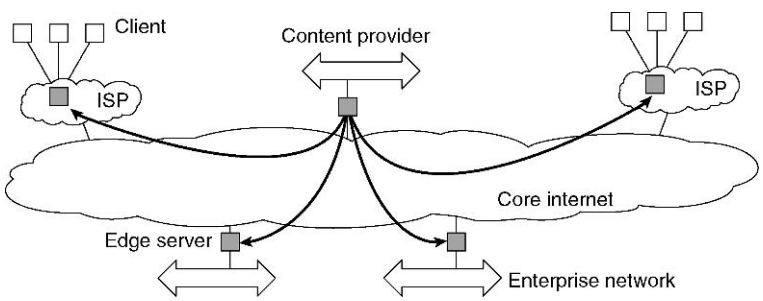
WAN data replication is driven by the need for improving application performance across wide-area networks. Performance is generally expressed in terms of client-perceived quality of service: response times (latency), data transfer rates (bandwidth), and availability. Replication may also be driven by the requirement for reducing monetary costs, as the infrastructure over which data distribution and replication takes place is generally owned by a separate provider who may be charging per transferred byte.

Replication of data has been explored in early wide-area systems such as Grapevine [2], Clearinghouse [4], and Lotus Notes [6]. All these systems concentrated on achieving a balance between data consistency, performance, and availability, recognizing that acceptable quality of service can be achieved only when weak consistency can be tolerated. In general, this required exploring application semantics making solutions more or less specific for an application.

In the mid-1990s, researchers from Xerox explored client-centric consistency models by which data were distributed and replicated in a wide-area database such that a notion of strong data consistency could be presented to users individually [8]. For example, data that had been modified by a user when accessing the database at location *A* would be propagated to location *B* before the user would access the system again at *B*, but not to other locations.

The need for WAN data replication became widely recognized with the explosion of the Web, which instantly revealed the shortcomings of its traditional client-server architecture. Up to date, virtually all Web sites are centrally organized, with end users sending requests to a single server. However, this organization does not suffice for large commercial sites, in which high performance and availability is crucial. To address these needs, so-called *Content Delivery Networks* (CDNs) came into play. A CDN is essentially a Web hosting service with many servers placed across the Internet (see Fig. 1). Its main goal is to ensure that a single Web site is automatically distributed and replicated across these servers in such a way that negotiated performance and availability requirements are met (see also [9]).

CDNs surfaced when most Web sites were still organized as a (possibly very large) collection of files that could be accessed through a single server. Modern sites, however, are no longer statically organized, but deploy full-fledged databases from which Web content



WAN Data Replication. Figure 1. Web-based data replication deploying servers that are placed at the edge of the Internet. Adapted from Tanenbaum, van Steen: Distributed Systems, 2nd edn. Prentice-Hall, Englewood, Cliffs, NJ, 2007.

is dynamically generated by application servers. As a consequence, the so-called edge servers to which client requests are initially directed are gradually turning into servers hosting partially or fully replicated databases, or database caches. These issues are discussed below.

Foundations

A popular model used to understand the various issues involved in wide-area data(base) is the one in which every data item has a single associated server through which all its updates are propagated. Such a primary or origin server as it is called, simplifies the handling of conflicts and maintenance of consistency. In practice, an origin server maintains a complete database that is partially or completely replicated to other servers. In contrast, in an *update anywhere* approach, updates may be initiated and handled at any replica server. The main problem with this approach is that it requires global consensus among the replica servers on the ordering of updates if strong consistency is to be preserved. Achieving such consensus introduces serious scalability problems, for which reason various optimistic approaches have been proposed (optimistic in the sense that corrective actions may later be necessary).

Queries are initially forwarded to edge servers, which then handle further processing. This could mean that subqueries are issued to different origin servers, but it is also possible that the edge server can compute the answer locally and send the response directly to the requesting client without further contacting the origin. In this context, key issues that need to be addressed for wide-area data replication are replica placement and consistency.

Replica Placement

Somewhat surprisingly, many researchers do not make a clear distinction between placement of server machines and placement of data on servers. Nevertheless, this distinction is important: while data placement can often be decided at runtime, this is obviously not the case for placement of server machines. Moreover, the criteria for placement are different: server placement should be done for many data objects, but deciding on the placement of data can be optimized for individual data objects.

Both problems can be roughly tackled as optimization problems in which the best K out of N possible locations need to be selected. There are a number of variations of this problem, but most important is the fact that heuristics need to be employed due to the exponential complexity of known solutions. For this reason, runtime data placement decisions deploy simpler solutions. An overview is provided in [13].

Relevant in this context is comparing different placements to decide which one is best. To this end, a general cost function can be used in various metrics which are combined:

$$\cos t = w_1\,\mathrm{res}_1 + w_2\mathrm{res}_2 + \cdots + w_n\,\mathrm{res}_n,$$
$$\mathrm{res}_k \geq 0, \quad w_k > 0$$

where res_k is a monotonically increasing variable denoting the cost of resource k and w_k its associated weight. Typical resources include distance (expressed in delay or number of hops) and bandwidth. Note that the actual dimension of *cost* is irrelevant. What matters is that different costs can be compared in order to select the best one.

Using a cost-driven placement strategy also implies that resource usage must be measured or estimated. In many cases, estimating costs may be more difficult than one would initially expect. For example, in order for an origin server to estimate the delay between a client and a given edge server may require mapping Internet locations to coordinates in a high-dimensional geometric space [7].

Data Consistency

Consistency of replicated data has received considerable attention, notably in the context of distributed shared-memory parallel computers. This class of computers attempts to mimic the behavior of traditional shared-memory multiprocessors on clusters or grids of computers. However, traditional distributed systems have often generally assumed that only strong consistency is acceptable, i.e., all processes concurrently accessing shared data see the same ordering of updates everywhere. The problem with strong consistency is that it requires timely global synchronization, which may be prohibitively expensive in wide-area networks. Therefore, weaker consistency models often need to be adopted.

To capture different models, Yu and Vahdat defined *continuous consistency* [15], a framework that captures consistency along three different dimensions: time, content, and ordering of operations. When updates are handled in a centralized manner, then notably the first two are relevant for WAN data replication. Continuous time-based consistency expresses to what

extent copies of the same data are allowed to be stale with respect to each other. Continuous content-based consistency is used to express to what extent the respective *values* of replicated data may differ, a metric that is useful when dealing with, e.g., financial data. The differences in consistency are subsequently expressed as numbers. By exchanging this information between sites, consistency enforcement protocols can be automatically started without further interference from applications.

There are essentially three ways to bring copies in the same state. First, with *state shipping* the complete up-to-date state is transferred to a replica server. This update form can be optimized through *delta shipping* by which only the difference between a replica's current state and that of a fresher replica is computed and transferred. These two forms of update are also referred to as passive replication, or asymmetric update processing. In contrast, with active replication, *function shipping* takes place, meaning that the operations that led to a new state are forwarded to a replica and subsequently executed. This is also known as *symmetric update processing*.

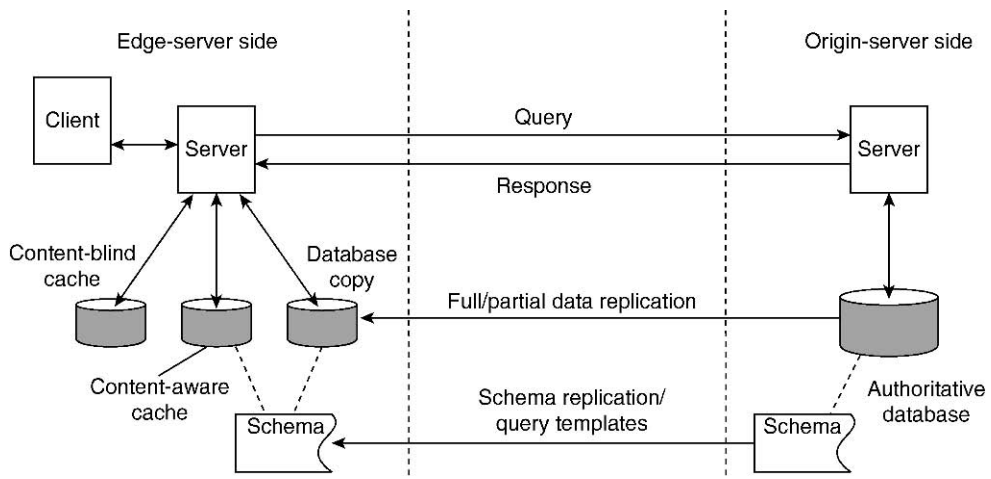
Differences may also exist with respect to the server taking the initiative for being updated. In *pull* approaches a replica server sends a request to a fresher replica to send it updates. In the *push* approach, updates are sent to a replica server at the initiative of a server that has just been updated. Note that for pushing, the server taking the initiative will need to know every other replica server as well as its state. In contrast, pulling in updates requires only that a replica server knows where to fetch fresh state. For these reasons, pull-based consistency enforcement is often used in combination with caches: when a request arrives at a caching server, the latter checks whether its cache is still fresh by contacting an origin server.

An important observation for WAN data replication is that it is impossible to simultaneously provide strong consistency, availability, and partition tolerance [5]. In the edge-server model, this means that despite the fact that an origin server is hosting a database offering the traditional ACID properties, the system as a whole cannot provide a solution that guarantees that clients will be offered continuous and correct service as long as at least one server is up and running. This is an important limitation that is often overlooked.

Key Applications

Wide-area data(base) replication is applied in various settings, but is arguably most prevalent in Web applications and services (see also [1]). To illustrate, consider a Web service deployed within an edge-server infrastructure. The question is what kind of replication schemes can be applied for the edge servers. The various solutions are sketched in Fig. 2.

First, full replication of the origin server's database to the edge servers can take place. In that case, queries can be handled completely at the edge server and problems evolve around keeping the database replica's



WAN Data Replication. Figure 2. Caching and replication schemes for Web applications. Adapted from Tanenbaum, van Steen: Distributed Systems, 2nd edn. Prentice-Hall, Englewood, Cliffs, NJ, 2007.

consistent. Full replication is generally a feasible solution when read/write ratios are high and queries are complex (i.e., spanning multiple database tables). The main problem is that if the number of replicas grow, one would need to resort to lazy replication techniques by which an edge server can continue to handle a query in parallel to bringing all replicas up-to-date, but at the risk of having to reconcile conflicting updates later on [10]. If queries and data can be organized such that access to only a single table is required (effectively leading to only simple queries), partial replication by which only the relevant table is copied to the edge server may form a viable optimization.

An alternative solution is to apply what is known as content-aware caching. In this case, an edge server has part of the database stored locally based on caching techniques. A query is assumed to fit a template allowing the edge server to efficiently cache and lookup query results. For example, a query such as "*select * from items where price < 50*" contains the answer to the more specific query "*select * from items where price < 20*." In this case, the edge server is actually building up its own version of a database, but now with a data schema that is strongly related to the structure of queries.

Finally, an edge server can also deploy content-blind caching by which query results are simply cached and uniquely associated with the query that generated them. In other words, unlike content-aware caching and database replication, no relationship with the structure of the query or data is kept track of.

As discussed in [14], each of these replication schemes has its merits and disadvantages, with no clear winner. In other words, it is not possible to simply provide a single solution that will fit all applications. As a consequence, distributed systems that aim to support WAN data replication will need to incorporate a variety of replication schemes if they are to be of general use.

Cross-references

- ▶ Autonomous Replication
- ▶ Consistency Models for Replicated Data
- ▶ Data Replication
- ▶ Distributed Databases
- ▶ Eventual Consistency Optimistic Replication and Resolution
- ▶ One-Copy Serializability
- ▶ Partial Replication
- ▶ Replica Control
- ▶ Replication Based on Group Communication
- ▶ Replication for High Availability
- ▶ Replication for Scalability
- ▶ Traditional Concurrency Control for Replicated Databases

Recommended Reading

1. Alonso G., Casati F., Kuno H., and Machiraju V. Web Services: Concepts, Architectures and Applications. Springer, Berlin Heidelberg New York, 2004.
2. Birrell A., Levin R., Needham R., and Schroeder M. Grapevine: an excercise in distributed computing. Commun. ACM, 25(4): 260–274, 1982.
3. Callaghan B. NFS Illustrated. Addison-Wesley, Reading, MA, 2000.
4. Demers A., Greene D., Hauser C., Irish W., Larson J., Shenker S., Sturgis H., Swinehart D., and Terry D. Epidemic algorithms for replicated database maintenance. In Proc. ACM SIGACT-SIGOPS 6th Symp. on the Principles of Dist. Comp., 1987, pp. 1–12.
5. Gilbert S. and Lynch N. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant Web services. ACM SIGACT News, 33(2):51–59, 2002.
6. Kawell L., Beckhardt S., Halvorsen T., Ozzie R., and Greif I. Replicated document management in a group communication system. In Proc. 2nd Conf. Computer-Supported Cooperative Work. 1988, pp. 226–235.
7. Ng E. and Zhang H. Predicting Internet Network Distance with Coordinates-based Approaches. In Proc. 21st Annual Joint Conf. of the IEEE Computer and Communications Societies, 2002, pp. 170–179.
8. Petersen K., Spreitzer M., Terry D., and Theimer M. Bayou: replicated database services for world-wide applications. In Proc. 7th SIGOPS European Workshop, 1996, pp. 275–280.
9. Rabinovich M. and Spastscheck O. Web Caching and Replication. Addison-Wesley, Reading, MA, 2002.
10. Saito Y. and Shapiro M. Optimistic replication. ACM Comput. Surv., 37(1):42–81, 2005.
11. Satyanarayanan M. Distributed files systems. In Distributed Systems, 2nd edn., S. Mullender (ed.). Addison-Wesley, Wokingham, 1993, pp. 353–383.
12. Shepler S., Callaghan B., Robinson D., Thurlow R., Beame C., Eisler M., and Noveck D. Network File System (NFS) Version 4 Protocol. RFC 3530, 2003.
13. Sivasubramanian S., Szymaniak M., Pierre G., and van Steen M. Replication for Web Hosting Systems. ACM Comput. Surv., 36(3):1–44, 2004.
14. Sivasubramanian S., Pierre G., van Steen M., and Alonso G. Analysis of Caching and Replication Strategies for Web Applications. IEEE Internet Comput., 11(1):60–66, 2007.
15. Yu H. and Vahdat A. Design and Evaluation of a Conit-based Continuous Consistency Model for Replicated Services. ACM Trans. Comput. Syst., 20(3):239–282, 2002.

Watermarking

► Storage Security

Wavelets on Streams

Minos Garofalakis
Technical University of Crete, Chania, Greece

Definition

Unlike conventional database query-processing engines that require several passes over a static data image, streaming data-analysis algorithms must often rely on building concise, approximate (but highly accurate) *synopses* of the input stream(s) in real-time (i.e., in one pass over the streaming data). Such synopses typically require space that is significantly sublinear in the size of the data and can be used to provide *approximate query answers*.

The collection of the top (i.e., largest) coefficients in the *wavelet transform* (or, *decomposition*) of an input data vector is one example of such a key feature of the stream. *Wavelets* provide a mathematical tool for the hierarchical decomposition of functions, with a long history of successful applications in signal and image processing [10]. Applying the wavelet transform to a (one- or multi-dimensional) data vector and retaining a select small collection of the largest wavelet coefficient gives a very effective form of lossy data compression. Such *wavelet summaries* provide concise, general-purpose summaries of relational data, and can form the foundation for fast and accurate approximate query processing algorithms.

Historical Background

Haar wavelets have recently emerged as an effective, general-purpose data-reduction technique for approximating an underlying data distribution, and providing a foundation for approximate query processing over traditional (static) relational data. Briefly, the idea is to apply the Haar wavelet decomposition to the input relation to obtain a compact summary that comprises a select small collection of *wavelet coefficients*. The results of [3,9,11] have demonstrated that fast and accurate approximate query processing engines can be designed to operate solely over such compact wavelet summaries. Wavelet summaries can also give accurate *histograms* of the underlying data distribution at multiple levels of resolution, thus providing valuable primitives for effective data visualization.

In the setting of static relational tables, the Haar wavelet transform is well understood, and scalable wavelet decomposition algorithms for constructing wavelet summaries have been known for some time [3,11]. Typically, such algorithms assume at least linear space and several passes over the data. Data streaming models introduce the novel challenge of maintaining the topmost wavelet coefficients over a dynamic data distribution (rendered as a stream of updates), while only utilizing *small space and time* (i.e., significantly sublinear in the size of the data distribution).

Foundations

The Haar Wavelet Decomposition. Consider the one-dimensional data vector $a = [2,2,0,2,3,5,4,4]$ comprising $N = 8$ data values. In general, N denotes the data set size and $[N]$ is defined as the integer index domain $\{0,...,N-1\}$; also, without loss of generality, N is assumed to be a power of 2 (to simplify notation). The Haar Wavelet Transform (HWT) of a is computed as follows. The data values are first averaged together pairwise to get a new "lower-resolution" representation of the data with the pairwise averages $\left[\frac{2+2}{2}, \frac{0+2}{2}, \frac{3+5}{2}, \frac{4+4}{2}\right] = [2,1,4,4]$. This averaging loses some of the information in a. To restore the original a values, *detail coefficients* that capture the missing information are needed. In the HWT, these detail coefficients are the differences of the (second of the) averaged values from the computed pairwise average. Thus, in this simple example, for the first pair of averaged values, the detail coefficient is 0 since $\frac{2-2}{2} = 0$, for the second it is -1 since $\frac{0-2}{2} = -1$. No information is lost in this process – one can reconstruct the eight values of the original data array from the lower-resolution array containing the four averages and the four detail coefficients. This pairwise averaging and differencing process is recursively applied on the lower-resolution array of averages until the overall average is reached, to get the full Haar decomposition. The final HWT of a is given by $w_a = [11/4, -5/4, 1/2, 0, 0, -1, -1, 0]$, that is, the overall average followed by the detail coefficients in order of increasing resolution. Each entry in w_a is called a *wavelet coefficient*. The main advantage of using w_a instead of the original data vector a is that for vectors containing similar values

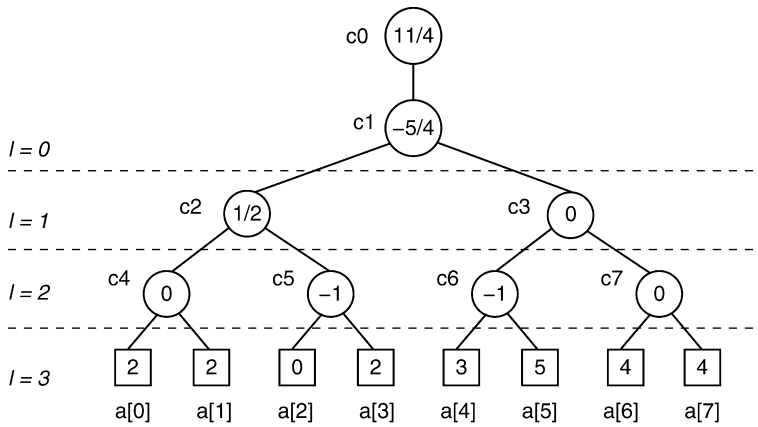
most of the detail coefficients tend to have very small values. Thus, eliminating such small coefficients from the wavelet transform (i.e., treating them as zeros) introduces only small errors when reconstructing the original data, resulting in a very effective form of lossy data compression [10].

A useful conceptual tool for visualizing and understanding the HWT process is the *error tree* structure [9] (shown in Fig. 1 for the example array a). Each internal tree node c_i corresponds to a wavelet coefficient (with the root node c_0 being the overall average), and leaf nodes $a[i]$ correspond to the original data-array entries. This view allows us to see that the reconstruction of any $a[i]$ depends only on the $\log N + 1$ coefficients in the path between the root and $a[i]$; symmetrically, it means a change in $a[i]$ only impacts its $\log N + 1$ ancestors in an easily computable way. The *support* for a coefficient c_i is defined as the contiguous range of data-array that c_i is used to reconstruct (i.e., the range of data/leaf nodes in the subtree rooted at c_i). Note that the supports of all coefficients at resolution level l of the HWT are exactly the 2^l (disjoint) *dyadic ranges* of size $N/2^l = 2^{\log N - l}$ over $[N]$, defined as $R_{l,k} = [k \cdot 2^{\log N - l}, \ldots, (k+1) \cdot 2^{\log N - l} - 1]$ for $k = 0, \ldots, 2^l - 1$ (for each resolution level $l = 0, \ldots, \log N$). The HWT can also be conceptualized in terms of vector inner-product computations: let $\phi_{l,k}$ denote the vector with $\phi_{l,k}[i] = 2^{l - \log N}$ for $i \in R_{l,k}$ and 0 otherwise, for $l = 0, \ldots, \log N$ and $k = 0, \ldots, 2^l - 1$; then, each of the coefficients in the HWT of a can be expressed as the inner product of a with one of the N distinct Haar *wavelet basis vectors*:

$$\{\frac{1}{2}(\phi_{l+1,2k} - \phi_{l+1,2k+1}) : l = 0, \ldots, \log N - 1; \; k = 0, \ldots, 2^l - 1\} \cup \{\phi_{0,0}\}$$

Intuitively, wavelet coefficients with larger support carry a higher weight in the reconstruction of the original data values. To equalize the importance of all HWT coefficients, a common normalization scheme is to scale the coefficient values at level l (or, equivalently, the basis vectors $\phi_{l,k}$) by a factor of $\sqrt{N/2^l}$. This normalization essentially turns the HWT basis vectors into an *orthonormal basis* – letting c_i^* denote the normalized coefficient values, this fact has two important consequences: (i) The *energy* (squared L_2-norm) of the a vector is preserved in the wavelet domain, that is, $||a||_2^2 = \langle a, a \rangle = \sum_i a[i]^2 = \sum_i (c_i^*)^2$ (by Parseval's theorem); and, (ii) Retaining the B largest coefficients in *absolute normalized value* gives the (provably) best B-term approximation in terms of L_2 (or, sum-squared) error in the data reconstruction (for a given budget of coefficients B) [10].

The HWT and its key properties also naturally extend to the case of *multi-dimensional* data distributions; in that case, the input a is a d-dimensional data array, comprising N^d entries (Without loss of generality, a domain of $[N]^d$ is assumed for the d-dimensional case.), and the HWT of a results in a d-dimensional wavelet-coefficient array w_a with N^d coefficient entries. The supports of d-dimensional Haar coefficients are d-dimensional hyper-rectangles (over dyadic ranges in $[N]^d$), and can be naturally arranged in a generalized error-tree structure [5,6].



Wavelets on Streams. Figure 1. Error-tree structure for the example data array a ($N = 8$).

Wavelet Summaries on Streaming Data. Abstractly, the goal is to continuously track a compact summary of the B topmost wavelet coefficient values for a dynamic data distribution vector a rendered as a continuous stream of updates. Algorithms for this problem should satisfy the key *small space/time* requirements for streaming algorithms; more formally, streaming wavelet algorithms should (ideally) guarantee (i) *sublinear space usage* (for storing a synopsis of the stream), (ii) *sublinear per-item update time* (to maintain the synopsis), and (iii) *sublinear query time* (to produce a, possibly approximate, wavelet summary), where "sublinear" typically means polylogarithmic in the domain size N. The streaming wavelet summary construction problem has been examined under two distinct data streaming models.

- *Wavelets in the Ordered Aggregate (Time Series) Streaming Model.* Here, the entries of the input data vector a are rendered over time in the increasing (or, decreasing) order of the index domain values. This means, for instance, that $a[1]$ (or, the set of all updates to $a[1]$) is seen first, followed by $a[2]$, then $a[3]$, and so on. In this case, the set of the B topmost HWT values over a can be maintained *exactly* in small space and time, using a simple algorithm based on the error-tree structure (Fig. 1) [7]: Consider reading (an update to) item $a[i+1]$ in the stream; that is, all items $a[j]$ for $j \leq i$ have already streamed through. The algorithm maintains the following two sets of (partial) coefficients:
  1. Highest B HWT coefficient values for the portion of the data vector seen thus far.
  2. $\log N + 1$ *straddling* partial HWT coefficients, one for each level of the error tree. At level l, index i *straddles* the HWT basis vector $\phi_{l,k}$, where $j \in [k \cdot 2^{\log N - l}, (k+1) \cdot 2^{\log N - l} - 1]$. (Note that there is *at most one* such basis vector per level.)

  When the $(i+1)th$ data item is read, the value for each of the affected straddling coefficients is updated. With the arrival of the $(i+1)th$ item, some coefficients may no longer be straddling (i.e., their computation is now complete). In that case, the value of these coefficients is compared against the the current set of B highest coefficients, and only the B largest coefficient values in the combined set are retained. Also, for levels where a straddling coefficient has been completed, a new straddling coefficient is initiated (with an initial value of 0). In this manner, at every position in the time series stream, the set of the B topmost HWT coefficients is retained exactly.

Theorem 1 *[7] The topmost B HWT coefficients can be maintained exactly in the ordered aggregate (time series) streaming model using O(B + logN) space and O(B + logN) processing time per item.*

Similar algorithmic ideas can also be applied to more sophisticated wavelet thresholding schemes (e.g., that optimize for error metrics other than L_2) to obtain space/time efficient wavelet summarization algorithms in the ordered aggregate model [8].

- *Wavelets in the General Turnstile Streaming Model.* Here, updates to the data vector a can appear in any arbitrary order in the stream, and the final vector entries are obtained by implicitly aggregating the updates for a particular index domain value. More formally, in the turnstile model, each streaming update is a pair of the form $(i, \pm v)$, denoting a net change of $\pm v$ in the $a[i]$ entry; that is, the effect of the update is to set $a[i] \leftarrow a[i] \pm v$. (The model naturally generalizes to multi-dimensional data: for d data dimensions, each update $((i_1,...,i_d), \pm v)$ effects a net change of $\pm v$ on entry $a[i_1,...,i_d]$.) The problem of maintaining an accurate wavelet summary becomes significantly more complex when moving to this much more general streaming model. Gilbert et al. [7] prove a strong *lower bound* on the space requirements of the problem: for arbitrary turnstile streaming vectors, *nearly all* of the data must be stored to recover the top B HWT coefficients.

  Existing solutions for wavelet maintenance over turnstile data streams rely on randomized schemes that return only an *approximate* synopsis comprising (at most) B Haar coefficients that is provably near-optimal (in terms of the captured energy of the underlying vector) assuming that the data vector satisfies the "*small-B property*" (i.e., most of its energy is concentrated in a small number of HWT coefficients) – this assumption is typically satisfied for most real-life data distributions [7]. One of the key ideas is to maintain a randomized *AMS sketch* [2], a broadly applicable stream synopsis structure comprising randomized linear projections of the streaming data vector a. Briefly, an *atomic AMS sketch* of a is simply the *inner product*

$\langle a, \xi \rangle = \sum_i a[i]\xi(i)$, where ξ denotes a random vector of four-wise independent $\pm$ 1-valued random variates.

Theorem 2 *[1,2] Consider two (possibly streaming) data vectors a and b, and let Z denote the O(log (1/δ))-wise median of $O(1/\epsilon^2)$-wise means of independent copies of the atomic AMS sketch product* $(\sum_i a[i]\xi(i))(\sum_i b[i]\xi_j(i))$. *Then,* $|Z - \langle a, b\rangle| \leq \epsilon||a||_2||b||_2$ *with probability* $\geq 1 - \delta$.

Thus, using AMS sketches comprising only $O(\frac{\log(1/\delta)}{\epsilon^2})$ atomic counters, the vector inner product $\langle a,b\rangle$ can be approximated to within $\pm\ \epsilon||a||_2||b||_2$ (hence implying an ϵ-relative error estimate for the squared L_2 norm $||a||_2^2$).

Since Haar coefficients of a are inner products with a fixed set of wavelet-basis vectors, the above theorem forms the key to developing efficient, approximate wavelet maintenance algorithms in the turnstile model. Gilbert et al. [7] propose a solution (termed "GKMS" in the remainder of the discussion) that focuses primarily on the one-dimensional case. GKMS maintains an AMS sketch for the streaming data vector a. To produce the approximate B-term representation, GKMS employs the constructed sketch of a to estimate the inner product of a with *all wavelet basis vectors*, essentially performing an exhaustive search over the space of all wavelet coefficients to identify important ones. More formally, assuming that there is a B-coefficient approximate representation of the signal with energy at least $\eta||a||_2^2$ ("small B property"), the GKMS algorithm uses a maintained AMS sketch to exhaustively estimate each Haar coefficient and selects up to B of the largest coefficients (excluding those whose square is less than $\eta\epsilon||a||_2^2/B$, where $\epsilon < 1$ is the desired accuracy guarantee). GKMS also uses techniques based on range-summable random variables constructed using Reed-Muller codes to reduce or amortize the cost of this exhaustive search by allowing the sketches of basis vectors (with potentially large supports) to be computed more quickly.

Theorem 3 *[7] Assuming there exists a B-term representation with energy at least* $\eta||a||_2^2$, *then, with probability at least* $1 - \delta$, *the GKMS algorithm finds a representation of at most B coefficients that captures at least* $(1 - \epsilon)\eta$ *of the signal energy* $||a||_2^2$, *using* $O(\log^2 N \log(N/\delta)B^2/(\eta\epsilon)^2)$ *space and per-item processing time.*

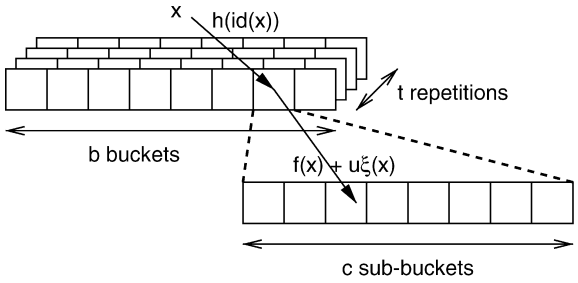
A potential problem lies in the query time requirements of the GKMS algorithm: even with the Reed-Muller code optimizations, the overall query time for discovering the top coefficients remains superlinear in N (i.e., at least $\Omega(\frac{1}{\epsilon^2} N \log N)$), violating the third requirement on streaming schemes. This also renders direct extensions of GKMS to multiple dimensions infeasible since it implies an exponential explosion in query cost (requiring at least $O(N^d)$) time to cycle through all coefficients in d dimensions). In addition, the update cost of the GKMS algorithm is *linear in the size of the sketch* since the whole data structure must be "touched" for each update. This is problematic for high-speed data streams and/or even moderate sized sketch synopses.

To address these issues, Cormode et al. [5] propose a novel solution that relies on two key technical ideas. First, they work *entirely in the wavelet domain*: instead of sketching the original data entries, their algorithms sketch the wavelet-coefficient vector w_a as updates arrive. This avoids any need for complex range-summable hash functions (i.e., Reed-Muller codes). Second, they employ *hash-based grouping* in conjunction with *efficient binary-search-like techniques* to enable very fast updates as well as identification of important coefficients in polylogarithmic time.

- *Sketching in the Wavelet Domain.* The first technical idea in [5] relies on the observation that it is possible efficiently produce sketch synopses of the stream *directly in the wavelet domain*. That is, the impact of each streaming update can be translated on the relevant wavelet coefficients. By the linearity properties of the HWT and the earlier description, an update to the data entries corresponds to only polylogarithmically many coefficients in the wavelet domain. Thus, on receiving an update to a, it can be directly converted to $O(polylog(N))$ updates to the wavelet coefficients, and an approximate (sketch) representation of the wavelet coefficient vector w_a can be maintained.
- *Time-Efficient Updates and Large-Coefficient Searches.* Sketching in the wavelet domain means that, at query time, an approximate representation of the wavelet-coefficient vector w_a is available, and can be employed to identify all those coefficients that are "large," relative to the total energy of the data $||\omega_a||_2^2 = ||a||_2^2$. While AMS sketches can provide such estimates (a point query is just a special case of an inner product), querying remains much

too slow taking at least $\Omega(\frac{1}{\epsilon^2}N)$ time to find which of the N coefficients are the B largest. Instead, the schemes in [5] rely on a *divide-and-conquer* or *binary-search-like* approach for finding the large coefficients. This requires the ability to efficiently estimate sums-of-squares for *groups* of coefficients, corresponding to dyadic subranges of the domain $[N]$. Low-energy regions can then be disregarded, recursing only on high-energy groups – this guarantees no false negatives, as a group that contains a high-energy coefficient will also have high energy as a whole. The algorithms of [5] also employ *randomized, hash-based grouping* of dyadic groups and coefficients to guarantee that each update only touches a small portion of the synopsis, thus guaranteeing very fast update times.

The key to the Cormode et al. solution is a hash-based probabilistic synopsis data structure, termed *Group-Count Sketch (GCS)*, that can estimate the energy of fixed groups of elements from a vector w of size N under the turnstile streaming model [5]. This translates to several streaming L_2-norm estimation problems (one per group). A simple solution would be to keep an AMS sketch of each group separately; however, there can be *many* (e.g., linear in N) groups, implying space requirements that are $O(N)$. Streaming updates should also be processed as quickly as possible. The GCS synopsis requires small, sublinear space and takes sublinear time to process each stream update item; more importantly, a GCS can provide a high-probability estimate of the energy of a group within additive error $\epsilon||\omega||_2^2$ in *sublinear time*. In a nutshell, the GCS synopsis first partitions items of w into their group using an `id()` function (which, in the case of Haar coefficients, is trivial since it corresponds to fixed dyadic ranges over $[N]$), and then randomly maps groups to buckets using a hash function $h()$. Within each bucket, a second stage of hashing of items to sub-buckets is applied (using another hash function $f()$, where each contains an atomic AMS sketch counter in order to estimate the L_2 norm of the elements in the bucket. As with most randomized estimation schemes, a GCS synopsis comprises t independent instantiations of this basic randomized structure, each with independently chosen hash function pairs $(h(), f())$ and ξ families for the AMS estimator; during maintenance, a streaming update (x, u) is used to update each of the t AMS counters corresponding to element x. (A pictorial representation is shown in Fig. 2.) To estimate the energy of a group g, for each independent instantiation $m = 1,...,t$ of the bucketing structure, the squared values of all the AMS counters in the sub-buckets corresponding to bucket $h_m(g)$ are summed, and then the *median* of these t values is returned as the estimate.



Wavelets on Streams. Figure 2. The GCS data structure: Element x is hashed (t times) to a bucket of groups (using $h(\mathrm{id}(x))$) and then a sub-bucket within the bucket (using $f(x)$), where an AMS counter is updated.

Theorem 4 *[5] The GCS can estimate the energy of item groups of the vector w within additive error* $\epsilon||\omega||_2^2$ *with probability* $\geq 1 - \delta$ *using space of* $O(\frac{1}{\epsilon^3}\log\frac{1}{\delta})$ *counters, per-item update time of* $O(\log\frac{1}{\delta})$, *and query time of* $O(\frac{1}{\epsilon^2}\log\frac{1}{\delta})$.

To recover coefficients with large energy in the w vector, the algorithm employs a *hierarchical search-tree structure* on top of $[N]$: Each level in this tree structure induces a certain partitioning of elements into groups (corresponding to the nodes at that level), and per-level GCS synopses can be used to efficiently recover the high-energy groups at each level (and, thus, quickly zero in on high-energy Haar coefficients). Using these ideas, Cormode et al. [5] demonstrate that the accuracy guarantees of Theorem 3 can be obtained using $O(\frac{B^3\log N}{\epsilon^3\eta^3}\cdot\log\frac{B\log N}{\epsilon\eta\delta})$ space, $O(\log^2 N\cdot\log\frac{B\log N}{\epsilon\eta\delta})$ per item processing time, *and* $O(\frac{B^3}{\epsilon^3\eta^3}\cdot\log N\cdot\log\frac{B\log N}{\epsilon\eta\delta})$ query time. In other words, their GCS-based solution guarantees sublinear space *and* query time, as well as per-item processing times that are sublinear *in the size of the stream synopsis*. Their results also naturally extend to the multi-dimensional case [5].

Key Applications

Wavelet-based summaries are a general-purpose data-reduction tool, and the maintenance of such summaries

over continuous data streams has several important applications, including large-scale IP network monitoring and network-event tracking (e.g., for detecting network traffic anomalies or Denial-of-Service attacks), approximate query processing over warehouse update streams, clickstream and transaction-log monitoring in large web-server farms, and satelite- or sensornet-based environmental monitoring.

Data Sets

Several publicly-available real-life data collections have been used in the experimental study of streaming wavelet summaries; examples include the US Census Bureau data sets (http://www.census.gov/), the UCI KDD Archive (http://kdd.ics.uci.edu/), and the UW Earth Climate and Weather Data Archive (http://www-k12.atmos.washington.edu/k12/grayskies/).

Future Directions

The area of streaming wavelet-based summaries is rich with interesting algorithmic questions. The bulk of the discussion here focuses on L_2-error synopses. The problem of designing efficient streaming methods for maintaining wavelet summaries that optimize for *non-L_2 error metrics* (e.g., general L_p error) under a turnstile streaming model remains open. Optimizing for non-L_2 error implies more sophisticated coefficient thresholding schemes based on dynamic programming over the error-tree structure (e.g., [6,8]); while such methods can be efficiently implemented over ordered aggregate streams [8], no space/time efficient solutions are known for turnstile streams. Dealing with *physically-distributed streams* also raises interesting issues for wavelet maintenance, such as trading off approximation quality with *communication* (in addition to time/space). The distributed AMS sketching algorithms of Cormode and Garofalakis [4] can be applied to maintain an approximate wavelet representation over distributed turnstile streams, but several issues (e.g., appropriate prediction models for local sites) remain open. Finally, from a systems perspective, the problem of incorporating wavelet summaries in industrial-strength data-streaming engines, and testing their viability in real-life scenarios remains open.

Cross-references

- ▶ Approximate Query Processing
- ▶ Data Compression in Sensor Networks
- ▶ Data Reduction
- ▶ Data Sketch/Synopsis
- ▶ Synopsis Structure
- ▶ Wavelets in Database Systems

Recommended Reading

1. Alon N., Gibbons P.B., Matias Y., and Szegedy M. Tracking join and self-join sizes in limited storage. In Proc. 18th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems, 1999, pp. 10–20.
2. Alon N., Matias Y., and Szegedy M. The space complexity of approximating the frequency moments. In Proc. 28th Annual ACM Symp. on Theory of Computing, 1996, pp. 20–29.
3. Chakrabarti K., Garofalakis M., Rastogi R., and Shim K. Approximate query processing using wavelets. In Proc. 26th Int. Conf. on Very Large Data Bases, 2000, pp. 111–122.
4. Cormode G. and Garofalakis M. Approximate continuous querying over distributed streams. ACM Trans. Database Syst., 33(2): 1–39, 2008.
5. Cormode G., Garofalakis M., and Sacharidis D. Fast approximate wavelet tracking on streams. In Advances in Database Technology, Proc. 10th Int. Conf. on Extending Database Technology, 2006, pp. 4–22.
6. Garofalakis M. and Kumar A. Wavelet synopses for general error metrics. ACM Trans. Database Syst., 30(4): 888–928, December 2005.
7. Gilbert A.C., Kotidis Y., Muthukrishnan S., and Strauss M.J. One-pass wavelet decomposition of data streams. IEEE Trans. Knowl. Data Eng., 15(3):541–554, May 2003.
8. Guha S. and Harb B. Wavelet synopsis for data streams: minimizing non-euclidean error. In Proc. 11th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, 2005, pp. 88–97.
9. Matias Y., Vitter J.S., and Wang M. Wavelet-based histograms for selectivity estimation. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1998, pp. 448–459.
10. Stollnitz E.J., DeRose T.D., and Salesin D.H. Wavelets for Computer Graphics – Theory and Applications. Morgan Kaufmann, San Francisco, CA, 1996.
11. Vitter J.S. and Wang M. Approximate computation of multidimensional aggregates of sparse data using wavelets. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1999, pp. 193–204.

Weak Consistency Models for Replicated Data

Alan Fekete
University of Sydney, Sydney, NSW, Australia

Synonyms

Weak memory consistency; Copy divergence

Definition

Some designs for a distributed database system involve having several copies or replicas for a data item, at different sites, with algorithms that do not update these replicas in unison. In such a system, clients may detect a discrepancy between the copies. Each particular weak consistency model describes which discrepancies may be seen. If a system provides a weak consistency model, then the clients will require more careful programming than otherwise. Eventual consistency (q.v.) is the best-known weak consistency model.

Historical Background

In the late 1980s and early 1990s, replication research focused on systems that allowed replicas to diverge from one another in controlled ways. Epidemic or multi-master algorithms were introduced in the work of Demers et al. [4]. These researchers identified the importance of session properties [8], which ensure that clients see information that includes changes they could reasonably expect the system to know. Ladin et al. [6] extended this concept by allowing the clients to explicitly indicate information they expect the system to know when responding to requests. Consistency models for single-master designs with replicas that lag behind a master were also explored by several groups [1,7,9]. Much of the work was presented in two Workshops on the Management of Replicated Data, held in 1990 and 1992.

Much later, Bernstein et al. [3] defined a consistency model called relaxed currency, in which stale reads can occur within transactions that also perform updates.

Foundations

In designing a distributed database system, the same data is usually kept replicated at different sites, for faster access and for better availability. However there is generally a substantial performance penalty from trying to keep the replicas perfectly synchronized with one another, so that clients never observe that replicas exist. Thus many system designers have proposed system designs where the replica control (q.v.) allows the clients to see that the replicas have diverged from one another. A consistency model captures the allowed observations that clients will make. In contrast to the models which maintain an illusion of a single-site system, a weaker consistency model allows the clients to see that the system has replicas. Different models are characterized by the ways in which the divergence between replicas is revealed. In describing some of the models, the research community is sometimes divided between definitions that speak directly about the values in the replicas, and definitions that deal with the events that occur at the clients. Another axis of variation in consistency models comes from the possibility of having different consistency requirements for read-only transactions. It is common to require, for example, that all the transactions which modify the data should execute with the strong consistency model of 1-copy-serializability (q.v.), while some weak consistency is allowed for those transactions which contain only read operations. This section does not try to present every known weak consistency model, but rather indicate some characteristic examples of different styles.

To motivate the weak consistency models, consider how badly things can go wrong in a system with uncontrolled read-one-write-all operations, where each update can be done anywhere first (inside the global client transaction) and then the updates are propagated lazily to the other replicas. Suppose there is such a system with one logical data item z which is an initially empty relation R(p,q,r) where p is a primary key, and two sites A and B each with a replica of z. In this example, there will be two different operations that can be performed on the data item: u which inserts a row (INSERT INTO R VALUES (101, 10,"Fred")), and v which modifies some rows (UPDATE R SET q = q +1 WHERE r = "Fred"). Consider $T_{1,C}$ which just performs u and is submitted by client C, and $T_{2,D}$ which performs v and is submitted by client D. Here is a possible sequence of events. In the notation used, $u_1[z^A,1]$ represents the event where u is performed on the local replica of item z at site A, as part of transaction T_1. Since the operations in the following example are modifications, the notation just shows as the return value the number of rows changed (1 in this particular event). Notice that the subscript on the event type indicates the transaction involved, and the superscript on the item name indicates the site of the replica which is affected. The event where the client C running transaction T_1 learns the return value, and thus knows that the operation has been performed, can be represented by $u_{1,C}[z,1]$. Note, there is no superscript on the item, since this is the client's view. Many consistency models need to refer to where transactions start and finish, so there are events like $b_{1,C}$ for the start of transaction T_1 at client C, or $c_{1,C}$ for the commit of that transaction by the client, or indeed

$c_1{}^A$ for the commit of the local subtransaction of T_1 which is running at site A. The symbols $T_{\odot}{}^A$ and $T_{\otimes}{}^B$ are used for the copier transactions that propagate updates previously done at other sites.

$$b_{1,C}\ b_{2,D}\ u_1[z^A, 1]\ u_{1,C}[z, 1]\ c_1^A\ c_{1,C}\ v_2[z^B, 0]$$
$$v_{2,D}[z, 0]\ c_2^B\ c_{2,D}\ b_{\odot}^A\ v_{\odot}[z^A, 1]\ c_{\odot}^A\ b_{\otimes}^B\ u_{\otimes}[z^B, 1]\ c_{\otimes}^B \quad (1)$$

Notice that in sequence (1), the operation v acts in a quite different way when performed at the two replicas (at z^B, no rows are found to match the where clause, while at the other replica a row is modified). This leads to a situation where the final state of replica z^A has a row (101, 11, "Fred"), whereas the final state of z^B has (101, 10, "Fred"). That is, the replicas have diverged from one another, and they will stay this way unless another operation is submitted to reset the row in question. What is worse, the effects of this divergence can spread through other operations, which will act quite differently when performed at the two sites. This situation has been called "system delusion" or "split brain." It is clearly unacceptable, and each weak consistency model should at least prevent this occurring.

Convergence. Several replica consistency models ensure that replicas eventually converge. Because this is a *liveness* feature, defining it formally involves looking at infinite sequences of events. However, informally, one can require that, provided the system enters a quiescent period without further changes, then a *convergence* time will be reached when all replicas have the same value, a value that includes the effects of all the updates on the item, performed in some order. Thus, after the convergence point any read on the logical item will show this value. Before the convergence time, a read is allowed to show any value that reflects some arrangement of some subset of the updates to that logical item (and different reads can show incompatible arrangements of different subsets of updates). This eventual consistency (q.v.) model was first proposed in general distributed computing research where operations are independent requests rather than combined into transactions; in replication of transactional DBMSs, there is an additional requirement that there should be an agreed serial order on all the updating transactions, so that the eventual state of every replica is the outcome of doing all the updating transactions in the given serial order.

The key mechanisms for ensuring eventual consistency involve detecting and then reconciling the situations where different replicas have applied operations in different orders. A common form of reconciliating involves having some deterministic rule to say which update should come first. A any replica where operations have been performed in the wrong order, inverse operations are used to rollback to an earlier state, and then the operations are reapplied in the correct order. For example, in the sequence (1) above, if the authoritative order has $T_{1,C}$ before $T_{2,D}$, then when $T_{\otimes}{}^B$ brings information to z^B about the missing operation u, the reconciler will use an inverse operation $v_{\otimes}{}^{-1}[z^B]$ to reverse the out-of-order operation v. This leads to the following sequence, in which the convergence point is reached at the end of $T_{\otimes}{}^B$ when the replicas are in consistent states.

$$b_{1,C}\ b_{2,D}\ u_1[z^A, 1]\ u_{1,C}[z, 1]\ c_1^A\ c_{1,C}\ v_2[z^B, 0]$$
$$v_{2,D}[z, 0]\ c_2^B\ c_{2,D}\ b_{\odot}^A\ v_{\odot}[z^A, 1]\ c_{\odot}^A\ b_{\otimes}^B\ v_{\otimes}^{-1}[z^B]$$
$$u_{\otimes}[z^B, 1]\ v_{\otimes}[z^B, 1]\ c_{\otimes}^B \quad (2)$$

In practice, the burden of reconciliation can be reduced somewhat, by noticing that the order does not matter, for many pairs of operations (those that *commute*, for example because they deal with different logical data items). Nevertheless, [5] has shown that the cost of reconciliation is a serious limit to scalability of multi-master designs that offer eventual consistency.

The eventual consistency model is focused on the way update operations are done; until convergence is reached, read-only transactions are quite unconstrained. Variations on the eventual consistency model are defined by limiting, in various ways, the nondeterminism in the values that can be returned in the read operations before the convergence point. For example, some consistency models enforce session properties. One notable property is called "read one's own writes." In this model, the subset of updates whose effects are seen in a read must include all writes (or more generally operations of update transactions) that were submitted at the same client, and further the arrangement of these writes must respect any cases where one transaction completed before another was started. More flexibly, some models, such as in [6] allow each client to indicate, when requesting a read, a set of previous transactions that must be included in the set of operations whose effects lead to the read's return value.

Stale replicas. It is hard to write reconciliation code to deal with all the different cases of out-of-order

updates. A major class of system designs avoids this issue entirely, by having a single *master* or primary replica, where all updates of the item are done first (it is common, but not essential, for the master replicas for all logical items to be at a single site). It is easy in system designs like this, to ensure that the updates are propagated one after another from the master replica to the other replicas (called "slaves"). In this model, there is always a "correct" value for each logical data item; this is the value in the master replica. Any slave always has a value which the master held in the past; it reflects some prefix of the transactions whose updates are reflected in the master, and so one says that the slave replica is a (possibly) stale version of the data. The term "quasi-copy" has been applied to a replica with out-of-date information [2]. The weakest, most permissive, consistency model for this system design is "relaxed currency" [3]. In this model, the clients' view is required to be equivalent to that in a serial, unreplicated and multi-version database (Note the change from the definition of 1-copy-serializability (q.v.), where the execution is equivalent to a serial, unreplicated, *single-version* database.). There must be a serial order on all the transactions, so that whenever the client reads a logical item, it sees some version but not necessarily the current one (it may be an older version that is returned).

Most system designs that offer relaxed currency provide a very restricted form: they also require that all the update transactions satisfy 1-copy-serializability (because all the operations of the update transactions are done at the master). In these designs, a read operation inside an update transaction must see the current version of the item; seeing a stale version can happen only in a read-only transaction.

To illustrate the relaxed currency consistency model, here is a sequence of events that might happen in a system with two sites A and B. A is the master site for integer-valued items x and y, while B has a slave copy of y but no replica for x. $T_{3,C}$ executes a transfer of 2 units from y to x by reading and then updating both items at the master site. The read-only transaction $T_{4,C}$ determines the status of the two logical items, first reading x^A (since there is only one copy for x) and then reading the slave replica y^B. $T_{\oplus}$ denotes the copier transaction that propagates the change in y to the slave at site B. The event where a value 12 is written to the local replica of item x at site A, as part of transaction T_3, is shown as $w_3[x^A, 12]$, and $w_{3,C}[x, 12]$ represents the notification of this write to the client at C. Similarly $r_4[y^B, 20]$ is the event where T_4 reads the value 20 in the replica of y at site B. As before one also has events like $b_{4,C}$ for the start of transaction T_4 at client C, or $c_{4,C}$ for the commit of that transaction by the client, or indeed $c_4{}^A$ for the commit of the local subtransaction of T_4 which is running at site A.

$$\begin{aligned} &b_{3,C}\ b_3^A\ r_3[x^A, 10]\ r_{3,C}[x, 10]\ r_3[y^A, 20]\ r_{3,C}[y, 20] \\ &w_3[x^A, 12]\ w_{3,C}[x, 12]\ w_3[y^A, 18]\ w_{3,C}[y, 18] \\ &c_3^A\ c_{3,C} b_{4,C}\ b_4^A\ r_4[x^A, 12]\ r_{4,C}[x, 12]\ b_4^B\ r_4[y^B, 20] \\ &r_{4,C}[y, 20]\ c_4^A\ c_4^B\ c_{4,C}\ b_{\oplus}^B\ w_{\oplus}[y^B, 18]\ c_{\oplus}^B \end{aligned} \tag{3}$$

At the end of the execution, the slave site is up-to-date, with the latest value for the logical item y it keeps a replica of. However, during the execution, the slave replica is sometimes behind the master, in particular this is true when T_4 reads the replica. Thus in sequence (3), the client view is

$$\begin{aligned} &b_{3,C}\ r_{3,C}[x, 10]\ r_{3,C}[y, 20]\ w_{3,C}[x, 12]\ w_{3,C}[y, 18] \\ &c_{3,C}\ b_{4,C}\ r_{4,C}[x, 12]\ r_{4,C}[y, 20]\ c_{4,C} \end{aligned} \tag{4}$$

which could occur in a serial multiversion system with a single site, in the order $T_{3,C}$ then $T_{4,C}$. Notice that in the serial multiversion system, when $T_{4,C}$ reads x it sees the latest version (produced by $T_{3,C}$), but its read of y returns an older "stale" version taken from the initial state.

There has been much work involving enhancing weak consistency definitions with quantitative bounds on the divergence between replicas.

Key Applications

The commercial DBMS vendors all offer replication mechanisms with their products, which can be configured in ways that offer different consistency properties. The most common approaches give either eventual convergence (if multiple masters are allowed, and conflicts are reconciled correctly) or reads from stale replicas (if there is a single master). There are also a range of research prototypes which give the user various consistency models; in general these show a tradeoff, where improved performance comes from offering weaker consistency models.

Future Directions

Most recent research on replica control finds new algorithms that provide one of the known weak consistency models, either eventual consistency or else the combination of 1-copy serializable updates with reads from stale copies. The main focuses of research have been to restrict how stale the reads can be, and how to make reconciliation easier to code correctly.

Cross-references

► Data Replication

Recommended Reading

1. Alonso R., Barbará D., and Garcia-Molina H. Data caching issues in an information retrieval system. ACM Trans. Database Syst., 15(3):359–384, 1990.
2. Alonso R., Barbará D., Garcia-Molina H., and Abad S. Quasi-copies: Efficient data sharing for information retrieval systems. In Advances in Database Technology, Proc. 1st Int. Conf. on Extending Database Technology, 1988, pp. 443–468.
3. Bernstein P.A., Fekete A., Guo H., Ramakrishnan R., and Tamma P. Relaxed-currency serializability for middle-tier caching and replication. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 2006, pp. 599–610.
4. Demers A.J., Greene D.H., Hauser C., Irish W., Larson J., Shenker S., Sturgis H.E., Swinehart D.C., and Terry D.B. Epidemic algorithms for replicated database maintenance. In Proc. ACM SIGACT-SIGOPS 6th Symp. on the Principles of Dist. Comp. 1987, pp. 1–12.
5. Gray J., Helland P., O'Neil P.E., and Shasha D. The dangers of replication and a solution. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1996, pp. 173–182.
6. Ladin R., Liskov B., Shrira L., and Ghemawat S. Providing high availability using lazy replication. ACM Trans. Comput. Syst., 10(4):360–391, 1992.
7. Sheth A.P. and Rusinkiewicz M. Management of interdependent data: Specifying dependency and consistency requirements. In Proc. 1st Workshop on the Management of Replicated Data, 1990, pp. 133–136.
8. Terry D.B., Demers A.J., Petersen K., Spreitzer M., Theimer M., and Welch B.B. Session guarantees for weakly consistent replicated data. In Proc. 3rd Int. Conf. on Parallel and Distributed Information Systems, 1994, pp. 140–149.
9. Wiederhold G. and Qian X. Consistency control of replicated data in federated databases. In Proc. 1st Workshop on the Management of Replicated Data, 1990, pp. 130–132.

Weak Coupling

► Loose Coupling

Weak Equivalence

Christian S. Jensen[1], Richard T. Snodgrass[2]
[1]Aalborg University, Aalborg, Denmark
[2]University of Arizona, Tucson, AZ, USA

Synonyms

Snapshot equivalence; Temporally weak

Definition

Informally, two tuples are *snapshot equivalent* or *weakly equivalent* if all pairs of timeslices with the same time instant parameter of the tuples are identical.

Let temporal relation schema R have n time dimensions, D_i, $i = 1,...,n$, and let τ^i, $i = 1,...,n$ be corresponding timeslice operators, e.g., the valid timeslice and transaction timeslice operators. Then, formally, tuples x and y are weakly equivalent if

$$\forall t_1 \in D_1 ... \forall t_n \in D_n (\tau^n_{t_n}(...(\tau^1_{t_1}(x))...) = \tau^n_{t_n}(...(\tau^1_{t_1}(y))...))$$

Similarly, two relations are snapshot equivalent or weakly equivalent if at every instant their snapshots are equal. Snapshot equivalence, or weak equivalence, is a binary relation that can be applied to tuples and to relations.

Key Points

The notion of weak equivalence captures the information content of a temporal relation in a point-based sense, where the actual timestamps used are not important as long as the same timeslices result. For example, consider the two relations with only a single attribute: {(a, [3, 9]} and {(a, [3, 5]), (a, [6, 9])}. These relations are different, but weakly equivalent.

Both "snapshot equivalent" and "weakly equivalent" are being used in the temporal database community. "Weak equivalence" was originally introduced by Aho et al. in 1979 to relate two algebraic expressions [1,2]. This concept has subsequently been covered in several textbooks. One must rely on the context to disambiguate this usage from the usage specific to temporal databases. The synonym "temporally weak" does not seem intuitive–in what sense are tuples or relations weak?

Cross-references

- ▶ Point-Stamped Temporal Models
- ▶ Snapshot Equivalence
- ▶ Temporal Database
- ▶ Time Instant
- ▶ Timeslice Operator
- ▶ Transaction Time
- ▶ Valid Time

Recommended Reading

1. Aho A.V., Sagiv Y., and Ullman J.D. Efficient optimization of a class of relational expressions. ACM Trans. Database Syst., 4(4):435–454, 1979.
2. Aho A.V., Sagiv Y., and Ullman J.D. Equivalences among relational expressions. SIAM J. Comput., 8(2):218–246, 1979.
3. Gadia S.K. Weak temporal relations. In Proc. 4th ACM SIGACT-SIGMOD Symp. on Principles of Database Systems, 1985, pp. 70–77.
4. Jensen C.S. and Dyreson C.E. (eds.). A consensus glossary of temporal database concepts – February 1998 version. In Temporal Databases: Research and Practice, O. Etzion, S. Jajodia, S. Sripada (eds.). LNCS 1399, Springer-Verlag, Berlin, 1998, pp. 367–405.

Weak Memory Consistency

- ▶ Weak Consistency Models for Replicated Data

Web 2.0 Applications

- ▶ Social Applications

Web 2.0/3.0

Alex Wun
University of Toronto, Toronto, ON, Canada

Definition

Web 2.0 and Web 3.0 are terms that refer to the trends and characteristics believed to be representative of next generation web applications emerging after the dot-com collapse of 2001. The term Web 2.0 became widely used after the first O'Reilly Media Web 2.0 Conference (http://www.web2con.com/) in 2004.

Key Points

The term Web 2.0 does not imply a fundamental change in Internet technologies but rather, captures what industry experts believe to be the key characteristics of successful web applications following the dot-com collapse. In general, Web 2.0 applications typically leverage one or more of the following concepts:

- Active participation, collaboration, and the wisdom of crowds. Consumers explicitly improve and add value to the web application they are using either through volunteerism or in response to incentives and rewards. This typically involves explicitly creating and modifying content (Wikipedia entries, Amazon reviews, Flickr tags, YouTube videos, etc.)
- Passive participation and network effects: Consumers naturally improve and add value to the web application being used as a result of the application's inherent design and architecture. Only a small percentage of users can be relied on to participate actively, so user contribution can be made non-optional by design. Examples include the ranking of most-clicked search results, most-viewed videos, peer-to-peer file sharing, and viral marketing.
- The Long-tail: Catering to niche user interests in volume and breadth can be just as successful if not more so than narrowly catering to only the most popular interests. This is due to the fact that content distribution over the Internet is not constrained by the physical limitations (such as location, storage, and consumer-base) of real-world vendors and service providers. For example, an online music vendor incurs virtually no additional cost to "stock" songs that do not sell well individually but when taken together, have a large population of fringe consumers.
- The Internet as a platform: Web 2.0 applications become extremely lightweight, flexible, and dynamic when the Internet is treated like a globally available platform with vast resources and an international consumer-base. Such applications act more like software services than traditional software products and evolve quickly by staying perpetually in beta. For Web 2.0 applications, global connectivity also extends beyond traditional PCs to portable handheld devices and other mobile devices. Web applications that mimic traditional desktop applications (such as online Operating Systems, work processors,

and spreadsheets) are examples of Web 2.0 applications using the Internet as a platform.

Web 2.0 enabling technologies primarily build on top of existing Internet technologies in the form of APIs and web development tools – AJAX, MashUp development tools, and other tools for creating rich media content for example.

Web 3.0 is still purely hypothetical and intends to capture the trends and characteristics of web applications beyond Web 2.0. Hypotheses vary widely between industry experts, ranging from the semantic web to available bandwidth as being the key influences of Web 3.0 applications.

Cross-references

- ▶ AJAX
- ▶ MashUp
- ▶ Service

Web Advertising

Vanja Josifovski[1], Andrei Broder[2]
[1]Uppsala University, Uppsala, Sweden
[2]Yahoo! Research, Santa Clara, CA, USA

Synonyms

Online advertising; Contextual advertising; Search advertising

Definition

Web advertising aims to place relevant ads on web pages. As in traditional advertising, most of the advertising on the web can be divided into brand advertising and direct advertising. In the majority of cases, brand advertising is achieved by banners – images or multimedia ads placed on the web page. As opposed to traditional brand advertising, on the web the user can interact with the ad and follow the link to a website of advertisers choice. Direct advertising is mostly in the form of short textual ads on the side of the search engine result pages and other web pages. Finally, the web allows for other types of advertising that are hybrids and cross into other media, such as video advertising, virtual worlds advertising, etc. Web advertising systems are built by implementing information retrieval, machine learning, and statistical techniques in a scalable, low-latency computing platform capable of serving billions of requests a day and selecting from hundreds of millions of individual advertisements.

Historical Background

The Web emerged as a new publishing media in the late 1990. Since then, the growth of web advertising has paralleled the growth in the number of web users and the increased time people spend on the Web. Banner advertising started with simple placement of banners on the top of the pages at targeted sites, and has since evolved into an elaborate placement scheme that targets a particular web user population and takes into account the content of the web pages and sites. Search advertising has its beginnings in the specialized search engines for ad search. Combining the web search with ad search was not something web users accepted from the start, but has become mainstream today. Furthermore, today's search advertising platforms have moved from simply asking the advertiser to provide a list of queries for which the ad is to be shown to employing variety of mechanisms to automatically learn what ad is appropriate for which query. Today's ad platforms are large, scalable and reliable systems running over clusters of machines that employ state-of-the-art information retrieval and machine learning techniques to serve ads at rates of tens of thousands times a second. Overall, the technical complexity of the advertising platforms rivals those of the web search engines.

Foundations

Web advertising spans Web technology, sociology, law, and economics. It has already surpassed some traditional mass media like broadcast radio and it is the economic engine that drives web development. It has become a fundamental part of the web eco-system and touches the way content is created, shared, and disseminated – from static html pages to more dynamic content such as blogs and podcasts, to social media such as discussion boards and tags on shared photographs. This revolution promises to fundamentally change both the media and the advertising businesses over the next few years, altering a $300 billion economic landscape.

As in classic advertising, in terms of goals, web advertising can be split into *brand advertising* whose goal is to create a distinct favorable image for the advertiser's product, and direct-marketing advertising that involves a "direct response": buy, subscribe, vote, donate, etc, now or soon.

In terms of delivery, there are two major types:

1. *Search advertising* refers to the ads displayed alongside the "organic" results on the pages of the Internet search engines. This type of advertising is mostly direct marketing and supports a variety of retailers from large to small, including micro-retailers that cover specialized niche markets.
2. *Contextual advertising* refers to ads displayed alongside some publisher-produced content, akin to traditional ads displayed in newspapers. It includes both brand advertising and direct marketing. Today, almost all non-transactional web sites rely on revenue from content advertising. This type of advertising supports sites that range from individual bloggers and small community pages, to the web sites of major newspapers. There would have been a lot less to read on the web without this model!

Web advertising is a big business, estimated in 2005 at $12.5 billion spent in the US alone (Internet Advertising Board – www.iab.com). But this is still less than 10% of the total US advertising market. Worldwide, internet advertising is estimated at $18B out of a $300. Thus, even at the estimated 13% annual growth, there is still plenty of room to grow, hence an enormous commercial interest.

From an ad-platform standpoint, both search and content advertising can be viewed as a matching problem: a stream of queries or pages is matched in real time to a supply of ads. A common way of measuring the performance of an ad-platform is based on the clicks on the placed ads. To increase the number of clicks, the ads placed must be relevant to the user's query or the page and their general interests.

There are several data engineering challenges in the design and implementation of such systems.

The first challenge is the volume of data and transactions: Modern search engines deal with tens of billions of pages from hundreds of millions of publishers, and billions of ads from tens of millions of advertisers. Second, the number of transactions is huge: billions of searches and billions of page views per day. Third, to deal with that, there is only a very short processing time available: when a user requests a page or types her query, the expectation is that the page, including the ads, will be shown in real time, allowing for at most a few tens of milliseconds to select the best ads.

To achieve such performance, ad-platforms usually have two components: a *batch processing component* that does the data collection, processing, and analysis, and a *serving component* that serves the ads in real time. Although both of these are related to the problems solved by today's data management systems, in both cases existing systems have been found inadequate for solving the problem and today's ad-platforms require breaking new grounds.

The batch processing component of an ad-system processes collections of multiple terabytes of data. Usually the data are not shared and a typical processing lasts from a few minutes to a few hours over a large cluster of hundreds, even thousands of commodity machines. The concern here is only about recovering from failures during the distributed computation and most of the time data are produced once and read one or more times. To be able to perform the same operation, a commercial database system would need to be deployed on the same scale that is beyond the scope of both shared-nothing and shared disk architectures. The reason for this is that database systems are designed for sharing data among multiple users in presence of updates and have overly complex distribution protocols to scale and perform efficiently at this scale. The backbone of the batch processing components is therefore being built using simpler distributed computation models and distributed file systems running over commodity hardware. Several challenges lay ahead to make these systems more usable and easier to maintain. The first challenge is to define a processing framework and interfaces such that large scale data analysis tasks (e.g., graph traversal and aggregation) and machine learning tasks (e.g., classification and clustering) are easy to express. So far there are two reported attempts to define such a query language for data processing in this environment [6,10]. However, there has been no reported progress on mapping these languages to a calculus and algebra model that will lend itself to optimization to improve optimization time. To make the task easier, new machine learning and data analysis algorithms for large scale data processing are needed. At the data storage layer, the challenge lies in designing a data storage that resides on the same nodes where the processing is performed.

The serving component of an advertising platform must have high throughput and low latency. To achieve this, in most cases the serving component operates over read-only copy of the data replaced occasionally by the batch component. The ads are

usually pre-processed and matched to an incoming query or page. The serving component has to implement reasoning that, based on a variety of features of the query/page and the ads, estimates the top few ads that have the maximum expected revenue within the constraints of the marketplace design and business rules associated to that particular advertising opportunity. The first challenge here is developing features and corresponding extraction algorithms appropriate for real-time processing. As the response time is limited, today's architectures rely on serving from a large cluster of machines that hold most of the searched data in-memory. One of the salient points of the design of the ad search is an objective function that captures the necessary trade-off between efficient processing and quality of results, both in terms of relevance and revenue. This function requires probabilistic reasoning in real-time.

Several different techniques can be used to select ads. When the number of ads is small, linear programming can be used to optimize for multiple ads grouped in *slates*. Banner ads are sometimes placed using linear programming as optimization technique [1,2]. In textual advertising the number of ads is too large to use linear programming. One alternative in such case is to use unsupervised methods based on information retrieval ranking [4,11]. Scoring formulas can be adapted to take into account the specificity of ads as documents: short text snippets with distinct multiple sections. Information retrieval scoring can be augmented with supervised ranking mechanisms that use the click logs or editorial judgments [9] to learn what ads work for a particular page/user/query. Here, the system needs to explore the space of possible matches. Some explore-exploit methods have been adapted to minimize the cost of exploration based on one-arm-bandit algorithms [3].

In summary, today's search and content advertising platforms are massive data processing systems that apply complicated data analysis and machine learning techniques to select the best advertising for a given query or page. The sheer scale of the data and the real-time requirements make this problem a very challenging task. Today's implementations have grown quickly and often in an ad-hoc manner to deal with a $15 billion fast growing market, but there is a need for improvement in almost every aspect of these systems as they adapt to even larger amounts of data, traffic, and new business models in web advertising.

Experimental Results

The reader is referred to [4,9,11] for the current published results on contextual advertising. A historical overview of search advertising is provided in [5]. Some results on query rewrites for search advertising are presented in [7,8].

Cross-references

- ▶ Information Retrieval Models
- ▶ Information Retrieval Operations

Recommended Reading

1. Abe N. Improvements to the linear programming based scheduling of Web advertisements. J. Electron. Commerce Res., 5(1):75–98, 2005.
2. Abrams Z., Mendelevitch O., and Tomlin J.A. Optimal delivery of sponsored search advertisements subject to budget constraints. In Proc. 8th ACM Conf. on Electronic Commerce, 2007, pp. 272–278.
3. Agarwal D., Chakrabarti D., Josifovski V., and Pandey S. Bandits for taxonomies: a model based approach. In Proc. SIAM International Conference on Data Mining, 2007.
4. Broder A., Fontoura M., Josifovski V., and Riedel L. A semantic approach to contextual advertising. In Proc. 33rd Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 2007, pp. 559–566.
5. Fain D. and Pedersen J. Sponsored search: a brief history. In Proc. 2nd Workshop on Sponsored Search Auctions. Web publication, 2006.
6. Group C.S. Community systems research at Yahoo! ACM SIGMOD Rec., 36(3):47–54, 2005.
7. Jones R. and Fain D.C. Query word deletion prediction. In Proc. 26th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 2003, pp. 435–436.
8. Jones R., Rey B., Madani O., and Greiner W. Generating query substitutions. In Proc. 15th Int. World Wide Web Conference, 2006, pp. 387–396.
9. Lacerda A., Cristo M., Andre M.G., Fan W., Ziviani N., and Ribeiro-Neto B. Learning to advertise. In Proc. 32nd Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 2006, pp. 549–556.
10. Pike R., Dorward S., Griesemer R., and Quinlan S. Interpreting the data: parallel analysis with Sawzall. Sci. Program. J., 13(4): 277–298, 2005.
11. Ribeiro-Neto B., Cristo M., Golgher P.B., and de Moura E.S. Impedance coupling in content-targeted advertising. In Proc. 31st Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 2005, pp. 496–503.

Web Application Server

- ▶ Application Server

Web Characteristics and Evolution

Dennis Fetterly
Microsoft Research, Mountain View, CA, USA

Definition

Web characteristics are properties related to collections of documents accessible via the World Wide Web. There are vast numbers of properties that can be characterized. Some examples include the number of words in a document, the length of a document in bytes, the language a document is authored in, the mime-type of a document, properties of the URL that indentifies a document, HTML tags used to author a document, and the hyperlink structure created by the collection of documents.

As in the physical world, the process of change that the web continually undergoes is identified as web evolution. The web is a tremendously dynamic place, with new users, servers, and pages entering and leaving the system continuously, which causes the web to change very rapidly. Web evolution encompasses changes in all web characteristics, as defined above.

Historical Background

As early as 1994, researchers were interested in studying characteristics of the World Wide Web. As documented by Woodruff et al. [13], the Lycos project at the Center for Machine Translation at Carnegie Mellon University calculated and published statistics on the web (HTTP, FTP, and GOPHER) pages it collected. Those initial statistics included the 100 "most weighty words" by TF-IDF, the document size in bytes, and the number of words in each document. The first estimate of the size of the web also occurred in 1994, when Bray seeded a web crawl with 40,000 documents and estimated the size of the web to be the number of unique URLs that could be fetched.

Using web documents collected in 1995, Woodruff et al. performed an extensive study [12] of 2.6 million HTML documents collected using the Inktomi web crawler. The motivations for this study included collecting information about the HTML markup to improve the language and measure the prevalence of non-standard language extensions. The authors performed a natural language analysis, correctness of the HTML markup, top-level domain distribution, in addition to nine per-page properties. The authors of that study also observed that the web evolved "exceptionally quickly" based on their observations of two data sets collected just months apart. They note that the properties of the documents that exist in both collections vary greatly and that many of the documents that exist in the first collection do not exist in the second.

In addition to characterizing documents discovered on the web, a number of usage based studies were performed under the auspices of the World Wide Web Consortium's (W3C) Jim Pitkow was the chair of the final working group, and he published a summary [11] of previous web characterization studies in 1999. These studies included characterizations of web requests, HTTP traffic, web document lifetime, and user behavior. In 2000, Broder et al. conducted a landmark study measuring the graph structure of the web [4] and demonstrating that connectivity of pages on the web could be classified into a structure that resembled a "bowtie."

Foundations

As highlighted by Baeza-Yates et al. [2], one of the principal issues faced by any initiative to characterize the web is how to select the sample of documents to characterize. The importance of this procedure is highlighted in a recent study that compares statistics from four different large web crawls [1] and finds that there are quantitative differences in the statistical characterizations of these collections. Several different methods for sampling have been utilized. The initial studies referred to in the "Historical Background" section were performed when the web was much smaller than it currently is and it was possible to crawl a reasonable fraction of the web in order to obtain a representative sample. However, even though those studies were able to crawl a reasonable fraction of the web, they were still only able to characterize content that they could fetch. A large portion of web content is inaccessible to the observer. This inaccessible content can be stored in a database, which is commonly called the deep web, be part of a corporate intranet, and thus not accessible from the public web, or could require authorization to access.

In order to estimate the size of the web, and the number of web servers providing web content, Lawrence and Giles chose Internet Protocol (IP) addresses at random and used the number of servers that successfully responded to estimate the total number

of web servers publically accessible [1]. They also crawled all of the accessible pages on the first 2,500 web servers that responded to requests sent to their IP address.

In order to address the issue of collecting a representative sample of connected, publically accessible web pages, Henzinger et al. describe a method to generate a near-uniform sample of URLs using a biased random walk on the web [10].

Another approach to sampling that has been employed in at least ten cases is to seed the collection with all domains registered in a national domain, for example Chile's national domain is .cl. Baeza-Yates et al. compare characterizations from ten national domains and two multinational web spaces (Africa and Indochina) [2]. They find that web characterizations that include a power law are consistent across all tested collections.

Another area of web characterization that has been the focus of significant study is the detection of duplicate and near-duplicate documents on the web. Detecting exact matches is much simpler than detecting near duplicates because each document can be reduced to a single fingerprint. Measures for computing text similarity, such as cosine similarity or the Jaccard similarity coefficient, require pairwise comparisons between documents, which will not complete when dealing with the numbers of documents that exist on the web. To put this in perspective, in their 1997 paper "Syntatic Clustering of the Web" [3] Broder et al. state that a pairwise comparison would involve $O(10^{15})$ (a quadrillion) comparisons. There have been two general approaches for condensing the information retained for each document and minimizing the number or required comparisons. The Broder et al. approach approximates the Jaccard similarity coefficient by retaining the minimum value computed by each of several hash functions, and then using those retained values, called sketches or shingles, to summarize the document. The computational effort required to discover near-duplicate documents using this algorithm is further reduced via the use of super-shingles, or sketches of sketches. Near-duplicate documents are very likely to share at least one super-shingle, which reduces the complexity to checking each of a small number of super-shingles. Charikar developed the other approach, which uses random projections of the words in a document to approximate the cosine similarity of documents [5]. Henzinger later experimentally compared [9] these two approaches.

Several different approaches have been used to measure the evolution of the World Wide Web. One common approach has utilized traces from web proxies to study evolution of web pages which provides insight into improved polices for caching proxies. Douglis et al. investigated [7] the rate of change of documents on the web in order to validate two critical assumptions that apply to web caching: a significant number of requests are duplicates of past requests, and the content associated with those requests does not change between accesses.

The evolution of the web has been extensively studied from a crawling perspective. In 2000, Cho and Garcia-Molina performed the first study of web page evolution [6]. They downloaded 720,000 pages daily from 270 "popular" hosts and computed a checksum over the entire page contents. These checksums were then compared to measure rates of change. Fetterly, Manasse et al. [8] expanded upon this study in 2003 by performing a breadth-first crawl of 150 million pages each week for 11 weeks and measuring the degree of change that occurred within each page.

At the intersection of web characteristics and evolution, Bar-Yossef et al. expand on the studies of link lifetime referenced by Pitkow [12] and investigate the decay of the web. They introduce the notion of a "soft 404," which is an error page returned with a HTTP result code of 200 – "OK" instead of 404 indicating an error has occurred.

Key Applications

A significant cost incurred when operating a search service is the cost of downloading and indexing content. Being able to predict, as accurately as possible, when to re-crawl a particular page allows the search engine to wisely utilize its resources. A search engine also needs to be able to identify near-duplicate documents that exist within its index, in order to either remove them or to crawl them at a reduced rate.

Characterization of the web enables at least two important functions: interested parties can measure the evolution of the web along many axes, barring significant evolution of the measured characteristic, results of past characterizations can be used to validate whether a new sample is a representative one or not.

Data Sets

The Stanford WebBase Project provives access to their repository of crawled web content. The Laboratory

for Web Algorithmics (LAW) at the Dipartimento di Scienze dell'Informazione (DSI) of the Università degli studi di Milano provides several web graph datasets. The Internet Archive is another potential source of data, although at the time of this writing, they do not currently grant new access to researchers.

http://dbpubs.stanford.edu:8091/~testbed/doc2/Web Base/
http://law.dsi.unimi.it/index.php?option=com_include&Itemid=65

Cross-references

- ▶ Data Deduplication
- ▶ Document Links and Hyperlinks
- ▶ Incremental Crawling
- ▶ Term Statistics for Structured Text Retrieval
- ▶ Web Search Result De-duplication and Clustering

Recommended Reading

1. Angeles Serrano M., Maguitman A., Santo Fortunato ná M.B., and Vespignani V. Decoding the structure of the www: A comparative analysis of web crawls. ACM Trans. Web, 1(2):10, 2007.
2. Baeza-Yates R., Castillo C., and Efthimiadis E.N. Characterization of national web domains. ACM Trans. Int. Tech., 7(2): 9, 2007.
3. Broder A.Z., Glassman S.C., Manasse M.S., and Zweig G. Syntactic clustering of the web. In Selected papers from the sixth International Conference on World Wide Web, 1997, pp. 1157–1166.
4. Broder A., Kumar R., Maghoul F., Raghavan P., Rajagopalan S., Stata R., Tomkins A., and Wiener J. Graph structure in the web: Experiments and models. In Proc. 8th Int. World Wide Web Conference, 2002.
5. Charikar M.S. Similarity estimation techniques from rounding algorithms. In Proc. 34th Annual ACM Symp. on Theory of Computing, 2002, pp. 380–388.
6. Cho J. and Garcia-Molina H. The evolution of the web and implications for an incremental crawler. In Proc. 26th Int. Conf. on Very Large Data Bases, 2000, pp. 200–209.
7. Douglis F., Feldmann A., Krishnamurthy B., and Mogul J.C. Rate of change and other metrics: a live study of the world wide web. In Proc. 1st USENIX Symp. on Internet Tech. and Syst., 1997.
8. Fetterly D., Manasse M., Najork M., and Wiener J. A large-scale study of the evolution of web pages. In Proc. 12th Int. World Wide Web Conference, 2003, pp. 669–678.
9. Henzinger M. Finding near-duplicate web pages: a large-scale evaluation of algorithms. In Proc. 32nd Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 2006, pp. 284–291.
10. Henzinger M.R., Heydon A., Mitzenmacher M., and Najork M. On near-uniform url sampling. Comput. Netw., 33(1–6): 294–308, 2000.
11. Lawrence S. and Giles L.C. Accessibility of information on the web. Nature, 400(6740):107–107, July 1999.
12. Pitkow J.E. Summary of www characterizations. Comput. Netw., 30(1–7):551–558, 1998.
13. Woodruff A., Aoki P.M., Brewer E.A., Gauthier P., and Rowe L.A. An investigation of documents from the world wide web. Comput. Netw., 28(7–11):963–980, 1996.

Web Content Extraction

- ▶ Fully Automatic Web Data Extraction

Web Content Mining

- ▶ Data, Text, and Web Mining in Healthcare
- ▶ Data Integration in Web Data Extraction System

Web Crawler

- ▶ Web Crawler Architecture

Web Crawler Architecture

Marc Najork
Microsoft Research, Mountain View, CA, USA

Synonyms

Web crawler; Robot; Spider

Definition

A web crawler is a program that, given one or more seed URLs, downloads the web pages associated with these URLs, extracts any hyperlinks contained in them, and recursively continues to download the web pages identified by these hyperlinks. Web crawlers are an important component of web search engines, where they are used to collect the corpus of web pages indexed by the search engine. Moreover, they are used in many other applications that process large numbers of web pages, such as web data mining, comparison shopping engines, and so on. Despite their conceptual simplicity, implementing high-performance web crawlers poses major engineering challenges due to the

scale of the web. In order to crawl a substantial fraction of the "surface web" in a reasonable amount of time, web crawlers must download thousands of pages per second, and are typically distributed over tens or hundreds of computers. Their two main data structures – the "frontier" set of yet-to-be-crawled URLs and the set of discovered URLs – typically do not fit into main memory, so efficient disk-based representations need to be used. Finally, the need to be "polite" to content providers and not to overload any particular web server, and a desire to prioritize the crawl towards high-quality pages and to maintain corpus freshness impose additional engineering challenges.

Historical Background

Web crawlers are almost as old as the web itself. In the spring of 1993, just months after the release of NCSA Mosaic, Matthew Gray [6] wrote the first web crawler, the World Wide Web Wanderer, which was used from 1993 to 1996 to compile statistics about the growth of the web. A year later, David Eichmann [5] wrote the first research paper containing a short description of a web crawler, the RBSE spider. Burner provided the first detailed description of the architecture of a web crawler, namely the original Internet Archive crawler [3]. Brin and Page's seminal paper on the (early) architecture of the Google search engine contained a brief description of the Google crawler, which used a distributed system of page-fetching processes and a central database for coordinating the crawl. Heydon and Najork described Mercator [8,9], a distributed and extensible web crawler that was to become the blueprint for a number of other crawlers. Other distributed crawling systems described in the literature include PolyBot [11], UbiCrawler [1], C-proc [4] and Dominos [7].

Foundations

Conceptually, the algorithm executed by a web crawler is extremely simple: select a URL from a set of candidates, download the associated web pages, extract the URLs (hyperlinks) contained therein, and add those URLs that have not been encountered before to the candidate set. Indeed, it is quite possible to implement a simple functioning web crawler in a few lines of a high-level scripting language such as Perl.

However, building a web-scale web crawler imposes major engineering challenges, all of which are ultimately related to scale. In order to maintain a search engine corpus of say, ten billion web pages, in a reasonable state of freshness, say with pages being refreshed every 4 weeks on average, the crawler must download over 4,000 pages/second. In order to achieve this, the crawler must be distributed over multiple computers, and each crawling machine must pursue multiple downloads in parallel. But if a distributed and highly parallel web crawler were to issue many concurrent requests to a single web server, it would in all likelihood overload and crash that web server. Therefore, web crawlers need to implement *politeness policies* that rate-limit the amount of traffic directed to any particular web server (possibly informed by that server's observed responsiveness). There are many possible politeness policies; one that is particularly easy to implement is to disallow concurrent requests to the same web server; a slightly more sophisticated policy would be to wait for time proportional to the last download time before contacting a given web server again.

In some web crawler designs (e.g., the original Google crawler [2] and PolyBot [11]), the page downloading processes are distributed, while the major data structures – the set of discovered URLs and the set of URLs that have to be downloaded – are maintained by a single machine. This design is conceptually simple, but it does not scale indefinitely; eventually the central data structures become a bottleneck. The alternative is to partition the major data structures over the crawling machines. Ideally, this should be done in such a way as to minimize communication between the crawlers. One way to achieve this is to assign URLs to crawling machines based on their host name. Partitioning URLs by host name means that the crawl scheduling decisions entailed by the politeness policies can be made locally, without any communication with peer nodes. Moreover, since most hyperlinks refer to pages on the same web server, the majority of links extracted from downloaded web pages is tested against and added to local data structures, not communicated to peer crawlers. Mercator and C-proc adopted this design [9,4].

Once a hyperlink has been extracted from a web page, the crawler needs to test whether this URL has been encountered before, in order to avoid adding multiple instances of the same URL to its set of pending URLs. This requires a data structure that supports set membership test, such as a hash table.

Care should be taken that the hash function used is collision-resistant, and that the hash values are large enough (maintaining a set of n URLs requires hash values with $\log_2 n^2$ bits each). If RAM is not an issue, the table can be maintained in memory (and occasionally persisted to disk for fault tolerance); otherwise a disk-based implementation must be used. Implementing fast disk-based set membership tests is extremely hard, due to the physical limitations of hard drives (a single seek operation takes on the order of 10 ms). For a disk-based design that leverages locality properties in the stream of discovered URLs as well as the domain-specific properties of web crawling, see [9]. If the URL space is partitioned according to host names among the web crawlers, the set data structure is partitioned in the same way, with each web crawling machine maintaining only the portion of the set containing its hosts. Consequently, an extracted URL that is not maintained by the crawler that extracted it must be sent to the peer crawler responsible for it.

Once it has been determined that a URL has not been previously discovered, it is added to the *frontier set* containing the URLs that have yet to be downloaded. The frontier set is generally too large to be maintained in main memory (given that the average URL is about 100 characters long and the crawling system might maintain a frontier of ten billion URLs). The frontier could be implemented by a simple disk-based FIFO queue, but such a design would make it hard to enforce the politeness policies, and also to prioritize certain URLs (say URLs referring to fast-changing news web sites) over other URLs. URL prioritization could be achieved by using a priority queue implemented as a heap data structure, but a disk-based heap would be far too expensive, since adding and removing a URL would require multiple seek operations. The Mercator design uses a frontier data structure that has two stages: a front-end that supports prioritization of individual URLs and a back-end that enforces politeness policies; both the front-end and the back-end are composed of a number of parallel FIFO queues [9]. If the URL space is partitioned according to host names among the web crawlers, the frontier data structure is partitioned along the same lines.

In the simplest case, the frontier data structure is just a collection of URLs. However, in many settings it is desirable to attach some attributes to each URL, such as the time when it was discovered, or (in the scenario of continuous crawling) the time of last download and a checksum or sketch of the document. Such historical information makes it easy to determine whether the document has changed in a meaningful way, and to adjust its crawl priority.

In general, URLs should be crawled in such a way as to maximize the utility of the crawled corpus. Factors that influence the utility are the aggregate quality of the pages, the demand for certain pages and topics, and the freshness of the individual pages. All these factors should be considered when deciding on the crawl priority of a page: a high-quality, highly-demanded and fast-changing page (such as the front page of an online newspaper) should be recrawled frequently, while high-quality but slow-changing and fast-changing but low-quality pages should receive a lower priority. The priority of newly discovered pages cannot be based on historical information about the page itself, but it is possible to make educated guesses based on per-site statistics. Page quality is hard to quantify; popular proxies include link-based measures such as PageRank and behavioral measures such as page or site visits (obtained from web beacons or toolbar data).

In addition to these major data structures, most web-scale web crawlers also maintain some auxiliary data structures, such as caches for DNS lookup results. Again, these data structures may be partitioned across the crawling machines.

Key Applications

Web crawlers are a key component of web search engines, where they are used to collect the pages that are to be indexed. Crawlers have many applications beyond general search, for example in web data mining (e.g., Attributor, a service that mines the web for copyright violations, or ShopWiki, a price comparison service).

Future Directions

Commercial search engines are global companies serving a global audience, and as such they maintain data centers around the world. In order to collect the corpora for these geographically distributed data centers, one could crawl the entire web from one data center and then replicate the crawled pages (or the derived data structures) to the other data centers; one could perform independent crawls at each data center and thus serve different indices to different

geographies; or one could perform a single geographically-distributed crawl, where crawlers in a given data center crawl web servers that are (topologically) close-by, and then propagate the crawled pages to their peer data centers. The third solution is the most elegant one, but it has not been explored in the research literature, and it is not clear if existing designs for distributed crawlers would scale to a geographically distributed setting.

URL to Code

Heritrix is a distributed, extensible, web-scale crawler written in Java and distributed as open source by the Internet Archive. It can be found at http://crawler.archive.org/

Cross-references

- ▶ Focused Web Crawling
- ▶ Incremental Crawling
- ▶ Indexing the Web
- ▶ Web Harvesting
- ▶ Web Page Quality Metrics

Recommended Reading

1. Boldi P., Codenotti B., Santini M., and Vigna S. UbiCrawler: a scalable fully distributed web crawler. Software Pract. Exper., 34(8):711–726, 2004.
2. Brin S. and Page L. The anatomy of a large-scale hypertextual search engine. In Proc. 7th Int. World Wide Web Conference, 1998, pp. 107–117.
3. Burner M. Crawling towards eternity: building an archive of the World Wide Web. Web Tech. Mag., 2(5):37–40, 1997.
4. Cho J. and Garcia-Molina H. Parallel crawlers. In Proc. 11th Int. World Wide Web Conference, 2002, pp. 124–135.
5. Eichmann D. The RBSE Spider – Balancing effective search against web load. In Proc. 3rd Int. World Wide Web Conference, 1994.
6. Gray M. Internet Growth and Statistics: Credits and background. http://www.mit.edu/people/mkgray/net/background.html
7. Hafri Y. and Djeraba C. High performance crawling system. In Proc. 6th ACM SIGMM Int. Workshop on Multimedia Information Retrieval, 2004, pp. 299–306.
8. Heydon A. and Najork M. Mercator: a scalable, extensible web crawler. World Wide Web, 2(4):219–229, December 1999.
9. Najork M. and Heydon A. High-performance web crawling. Compaq SRC Research Report 173, September 2001.
10. Raghavan S. and Garcia-Molina H. Crawling the hidden web. In Proc. 27th Int. Conf. on Very Large Data Bases, 2001, pp. 129–138.
11. Shkapenyuk V. and Suel T. Design and Implementation of a high-performance distributed web crawler. In Proc. 18th Int. Conf. on Data Engineering, 2002, pp. 357–368.

Web Data Extraction

▶ Web Harvesting

Web Data Extraction System

Robert Baumgartner[1,2], Wolfgang Gatterbauer[3], Georg Gottlob[4]

[1]Vienna University of Technology, Vienna, Austria

[2]Lixto Software GmbH, Vienna, Austria

[3]University of Washington, Seattle, WA, USA

[4]Oxford University, Oxford, UK

Synonyms

Web information extraction system; Wrapper generator; Web macros; Web scraper

Definition

A web data extraction system is a software system that automatically and repeatedly extracts data from web pages with changing content and delivers the extracted data to a database or some other application. The task of web data extraction performed by such a system is usually divided into five different functions: (i) *web interaction*, which comprises mainly the navigation to usually pre-determined target web pages containing the desired information; (ii) support for *wrapper generation and execution*, where a wrapper is a program that identifies the desired data on target pages, extracts the data and transforms it into a structured format; (iii) *scheduling*, which allows repeated application of previously generated wrappers to their respective target pages; (iv) *data transformation*, which includes filtering, transforming, refining, and integrating data extracted from one or more sources and structuring the result according to a desired output format (usually XML or relational tables); and (v) *delivering* the resulting structured data to external applications such as database management systems, data warehouses, business software systems, content management systems, decision support systems, RSS publishers, email servers, or SMS servers. Alternatively, the output can be used to generate new web services out of existing and continually changing web sources.

Historical Background

The precursors of web data extraction systems were screen scrapers which are systems for extracting screen

formatted data from mainframe applications for terminals such as VT100 or IBM 3270. Another related issue are *ETL* methods (ETL = Extract, Transform, Load), which extract information from various business processes and feed it into a database or a data warehouse. A huge gap was recognized to exist between web information and the qualified, structured data as usually required in corporate information systems or as envisioned by the Semantic Web. In many application areas there has been a need to automatically extract relevant data from HTML sources and translate this data into a structured format, e.g., XML, or into a suitable relational database format.

A first and obvious solution was the evolution from screen scrapers to web scrapers, which can navigate to web pages and extract textual content. However, web scrapers usually lack the logic necessary to define highly structured output data as opposed to merely text chunks or textual snippets. Moreover, they are usually unable to collect and integrate data from different related sources, to define extraction patterns that remain stable in case of minor layout changes, to transform the extracted data into desired output formats, and to deliver the refined data into different kinds of applications. For this reason, specific research on web data extraction was needed and several academic projects and some commercial research projects on web data extraction were initiated before or around the year 2000. While the academic projects such as XWRAP [11], Lixto [2], Wargo [14], and the commercial systems RoboMaker of Kapow technologies (Kapow Technologies. http://www.kapowtech.com/) and WebQL of QL2 Software (QL2 Software Inc. http://www.ql2.com/) focused on methods of strongly supervised "semi-automatic" wrapper generation, providing a wrapper designer with visual and interactive support for declaring extraction and formatting patterns, other projects were based on machine learning techniques. For example, WIEN [9], Stalker [13], and DEByE [10] focused on automatic wrapper induction from annotated examples. Some other projects focused on specific issues such as automatic generation of structured data sets from web pages [3] and particular aspects of navigating and interacting with web pages [1]. Finally, some systems require the wrapper designer to program the wrapper in a high level language (e.g., SQL-like languages for selecting data from a web page) and merely give visual support to the programmer; an example is the W4F system [15]. A number of early academic prototypes gave rise to fully fledged systems that are now commercially available. For example, Stalker gave rise to the Fetch Agent Platform of Fetch Technologies (Fetch Technologies, Inc. http://www.fetch.com/), the Lixto system gave rise to the Lixto Suite of the Lixto Software company (Lixto Software GmbH. http://www.lixto.com/), and the Wargo system evolved into the Denodo Platform [14]. A good and almost complete survey of web information extraction systems up to 2002 is given in [8].

Foundations

Formal foundations and semantics of data extraction

There are four main formal approaches to define (the semantics of) web wrappers:

- In the *functional approach*, a web wrapper is seen as a mapping f from the parse or DOM tree T of a web page to a set $f(T)$ of sub-trees of T, where each sub-tree corresponds to an extracted data object. Another step specifies the relabeling of extracted data to fit a previously defined output schema.
- The *logical approach* (see Logical foundations of web data extraction) consists of the specification of a finite number of monadic predicates, i.e., predicates, that define sets of parse-tree nodes and that evaluate to true or false on each node of a tree-structured document. The predicates can be defined either by logical formulas, or by logic programs such as monadic datalog, which was shown to be equivalent in expressive power to monadic second-order logic (MSO) [6]. Note that languages such as XPATH or regular path expressions are subsumed by MSO.
- The *automata theoretic approach* relies on tree automata, which are a generalization of finite state automata from words to trees. The equivalence of the two logical approaches, MSO and monadic datalog, and the unranked query automata approach shows that the class of wrappers definable in these formalisms is quite natural and robust. Fortunately, this class is also quite expressive, as it accommodates most of the relevant data extraction tasks and properly contains the wrappers definable in some specifically designed wrapper programming languages [7].
- Finally, the *textual approach* interprets web pages as text strings and uses string pattern matching for data extraction [13].

Methods of wrapper generation

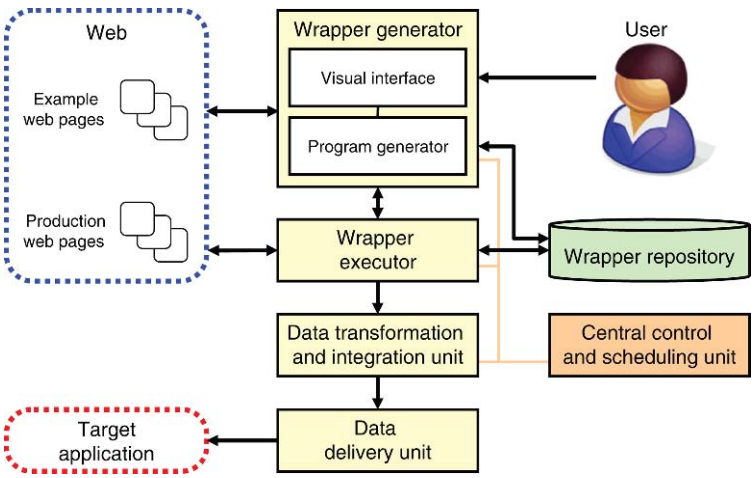
With regard to user involvement, three principal approaches for wrapper generation can be distinguished: (i) *Manual wrapper programming*, in which the system merely supports a user in writing a specific wrapper but cannot make any generalizations from the examples provided by the user; (ii) *wrapper induction*, where the user provides examples and counterexamples of instances of extraction patterns, and the system induces a suitable wrapper using machine learning techniques; and (iii) *semi-automatic interactive wrapper generation*, where the wrapper designer not only provides example data for the system, but rather accompanies the wrapper generation in a systematic computer-supported interactive process involving generalization, correction, testing, and visual programming techniques.

Architecture

Figure 1 depicts a high-level view of a typical fully-fledged semi-automatic interactive web data extraction system. This system comprises several tightly connected components and interfaces three external entities: (i) *the Web*, which contains pages with information of interest; (ii) a *target application*, to which the extracted and refined data will be ultimately delivered; and (iii) *the user*, who interactively designs the wrapper.

The wrapper generator supports the user during the wrapper design phase. It commonly has a *visual interface* that allows the user to define which data should be extracted from web pages and how this data should be mapped into a structured format such as XML. The visual interface itself can include several windows, such as: (i) a browser window that renders example web pages for the user; (ii) a parse tree window that shows the HTML parse tree of the current web page rendered by the browser; (iii) a control window that allows the user to view and control the overall progress of the wrapper design process and to input textual data (e.g., attribute names or XML tags); and (iv) a program window that displays the wrapper program constructed so far and allows the user to further adjust or correct the program.

The subunit that actually generates the wrapper is contained in what is referred to here as the *program generator*. This submodule interprets the user actions on the example web pages and successively generates the wrapper. Semi-automatic wrapper generators allow the user to either specify the URL of example web pages, which are structurally similar to the target web pages, or to navigate to such example pages. In the latter case, the navigation is recorded and can be automatically reproduced. The example pages are rendered in a browser window. Most wrapper generators also display the parse tree of the current page. In most systems, the wrapper designer can click on a node of the HTML parse tree, and the system immediately highlights the corresponding part of the example page in the browser, and vice versa. The designer can, thus, iteratively narrow down an example of a relevant data item after a few trials. Many web data extraction systems exhibit an XPATH-like path expression that precisely identifies the selected data item. The wrapper designer can then generalize this path expression by replacing some of the elements with wildcards.



Web Data Extraction System. Figure 1. Architecture of a typical web data extraction system.

The result is a generalized pattern that matches several similar data items on similarly structured pages. Some systems (e.g., Denodo and Lixto) allow a wrapper designer to select data items directly on the rendered web page. The wrapper generator often also provides support for associating a name or tag to each extraction pattern and for organizing patterns hierarchically. Systems based on wrapper induction allow the user to supply a large number of sample web pages with positive and/or negative examples of desired data items, from which the program generator tries to learn a generalized pattern. Systems for manual wrapper programming simply provide a visual environment for developing a wrapper in the underlying wrapper programming language, for testing the wrapper against example pages and for debugging. In addition, the visual interface usually allows an application designer to specify when a wrapper should be executed.

The *wrapper executor* is the engine that facilitates the deployment of the previously generated wrappers. Typically, a wrapper program comprises functions such as deep web navigation (e.g., the means to fill out forms), identification of relevant elements with declarative rules or procedural scripts, and generation of a structured output format (e.g., XML or relational tables). Typically, the wrapper executor can also receive additional input parameters such as a start URL, a predefined output structure (e.g., a particular XML schema), and special values for forms. In some approaches, the output includes metadata (e.g., verification alerts in case of violated integrity constraints) in addition to the actual data of interest.

The *wrapper repository* stores the generated wrappers together with metadata. In some approaches, wrapper templates and libraries of domain concepts can be reused and extended. Systems such as Dapper (Dapper. http://www.dapper.net/) offer a community-based extensible wrapper repository. In principle, systems can offer automatic wrapper maintenance and adaptation as part of the repository.

The *data transformation and integration unit* provides a means to streamline the results of multiple wrappers into one harmonized homogeneous result. This step includes data integration, transformation and cleaning, basically functions commonly found in the Transform step of ETL processes of data warehouses and mediation systems. Technologies and tools comprise query languages, visual data mapping, automatic schema matching and de-duplication techniques. Additionally, the data can be composed into desired result formats such as HTML, Excel or PDF.

The *data delivery unit* (capturing the Load step in the ETL philosophy) decides about the appropriate output channel such as email, Open Database Connectivity (ODBC) connection, or FTP. Advanced wrapper generation systems offer a number of system connectors to state-of-the-art databases and enterprise software such as Customer Relationship Management (CRM) or Enterprise Resource Planning (ERP), or partner with software providers in the area of Enterprise Application Integration (EAI) or Business Intelligence (BI) who offer such solutions.

The *central control and scheduling unit* is the heart of the processing engine. In general, synchronous and asynchronous extraction processes can be distinguished. A typical example of an asynchronous triggering is a real-time user request in a flight meta-search application. The request is queued, the respective wrappers are triggered, parameter mappings are performed, and the result is integrated and presented to the user in real-time. Sophisticated approaches offer a workflow-based execution paradigm, including parallelism, returning partial results, and interception points in complex booking transactions. On the other hand, market monitoring or web process integration are typical synchronous scenarios. An intelligent scheduler is responsible for distributing requests to the wrapper executor, iterating over a number of possible form inputs, and dealing with exception handling.

Commercial wrapper generation systems

Dapper. The Dapp Factory of Dapper offers fully server-based wrapper generation, and supports a community driven reusable wrapper repository. Dapper strongly relies on machine learning techniques for wrapper generation and concentrates exclusively on web pages that can be reached without deep web navigation. Users designing a wrapper label positive and negative examples and debug the result on a number of similarly structured web pages. Wrappers are hosted on the server and wrapper results can be queried via REST. Commercial APIs are offered for using Dapper in Enterprise scenarios.

Denodo (Denodo. http://www.denodo.com). The Denodo ITPilot, formerly known as Wargo, is a platform for creating and executing navigation and extraction scripts that are loosely tied together. It offers graphical wizards for configuring wrappers and allows DOM events to be processed while navigating web pages. Deep Web navigation can be executed in

Internet Explorer, and the result pages are passed on to the extraction program. Furthermore, ITPilot offers some wrapper maintenance functionalities. Denodo additionally offers a tool called Aracne for document crawling and indexing.

Lixto. The Lixto Suite comprises the Lixto Visual Developer (VD), a fully visual and interactive wrapper generation framework, and the Java-based Lixto Transformation Server providing a scalable runtime and data transformation environment, including various Enterprise Application connectors. VD is ideally suited for dynamic Web 2.0 applications as it supports a visual Deep Web macro recording tool that makes it possible to simulate user clicks on DOM elements. Tightly connected with the navigation steps and hidden from the wrapper designer, the expressive language Elog is used for data extraction and linking to further navigation steps. VD is based on Eclipse and embeds the Mozilla browser.

Kapowtech. The Kapow RoboSuite (recently rebranded to Kapow Mashup Server) is a Java-based visual development environment for developing web wrappers. It embeds a proprietary Java browser, implements a GUI on top of a procedural scripting language, and maps data to relational tables. The RoboServer offers APIs in different programming languages for synchronous and asynchronous communication. In addition, variants of the Mashup Server for specific settings such as content migration are offered.

WebQL. QL2 uses a SQL-like web query language called WebQL for writing wrappers. By default, WebQL uses its own HTML DOM tree model instead of relying on a standard browser, but a loose integration with Internet Explorer is offered. The QL2 Integration Server supports concepts such as server clustering and HTML diffing. Furthermore, an IP address anonymization environment is offered.

Key Applications

Web market monitoring. Nowadays, a lot of basic information about competitors can be retrieved legally from public information sources on the Web, such as annual reports, press releases or public data bases. On the one hand, powerful and efficient tools for Extracting, Transforming and Loading (ETL) data from internal sources and applications into a Business Intelligence (BI) data warehouse are already available and largely employed. On the other hand, there is a growing economic need to efficiently integrate external data, such as market and competitor information, into these systems as well. With the World Wide Web as the largest single database on earth, advanced data extraction and information integration techniques as described in this paper are required to process this web data automatically. At the same time, the extracted data has to be cleaned and transformed into semantically useful formats and delivered in a "web-ETL" process into a BI system. Key factors in this application area include scalable environments to extract and schedule processing of very large data sets efficiently, capabilities to pick representative data samples, cleaning extracted data to make it comparable, and connectivity to data warehouses.

Web process integration. In markets such as the automotive industry, business processes are largely carried out by means of web portals. Business critical data from various divisions such as quality management, marketing and sales, engineering, procurement and supply chain management have to be manually gathered from web portals. Through automation, suppliers can dramatically reduce the cost while at the same time improving speed and reliability of these processes. Additionally, the automation means to leverage web applications to web services, and hence wrapper generation systems can be considered as an enabling technology for Service Oriented Architectures (SOA) as envisioned and realized in Enterprise Application Integration (EAI) and B2B processes. Key factors in this application area are workflow capabilities for the whole process of data extraction, transformation and delivery, capabilities to treat all kinds of special cases occurring in web interactions, and excellent support of the latest web standards used during secure transactions.

Mashups. Increasingly, leading software vendors have started to provide mashup platforms such as Yahoo! Pipes or IBM QEDWiki. A mashup is a web application that combines a number of different websites into an integrated view. Usually, the content is taken via APIs by embedding RSS or atom feeds similar to REST (Representational State Transfer). In this context, wrapper technology transforms legacy web applications to light-weight APIs that can be integrated in mashups in the same way. As a result, web mashup solutions no longer need to rely on APIs offered by the providers of websites, but can rather extend the scope to the whole Web. Key factors for this application scenario include efficient real-time extraction capabilities for a large number of concurrent queries and detailed understanding of how to map queries to particular web forms.

Future Directions

►*Generic web wrapping.* Without explicit semantic annotations on the current Web data extraction systems that allow general web wrapping will have to move towards fully automatic wrapping of the existing World Wide Web. Important research challenges are (i) how to optimally bring semantic knowledge into the extraction process, (ii) how to make the extraction process robust (in particular, how to deal with false positives), and (iii) how to deal with the immense scaling issues. Today's web harvesting and automated information extraction systems show much progress (see e.g., [4,12]), but still lack the combined recall and precision necessary to allow for very robust queries. A related topic is that of *auto-adapting wrappers.* Most wrappers today depend on the tree structure of a given web page and suffer from failure when the layout and code of web pages change. Auto-adapting wrappers, which are robust against such changes, could use existing knowledge of the relations on previous versions of web page in order to automatically "heal" and adapt the extraction rules to the new format. The question here is how to formally capture change actions and execute the appropriate repair actions. ►*Wrapping from visual layouts.* Whereas web wrappers today dominantly focus on either the flat HTML code or the DOM tree representation of web pages, recent approaches aim at extracting data from the CSS box model and, hence, the visual representation of web pages [5]. This method can be particularly useful for layout-oriented data structures such as web tables and allows to create automatic and domain-independent wrappers which are robust against changes of the HTML code implementation. ►*Data extraction from non-HTML file formats.* There is a substantial interest from industry in wrapping documents in formats such as PDF and PostScript. Wrapping of such documents must be mainly guided by a visual reasoning process over white space and Gestalt theory, which is substantially different from web wrapping and will, hence, require new techniques and wrapping algorithms including concepts borrowed from the document understanding community. ►*Learning to deal with web interactions.* As web pages are becoming increasingly dynamic and interactive, efficient wrapping languages have to make it possible to record, execute and generalize macros of web interactions and, hence, model the whole process of workflow integration. An example of such a web interactions is a complicated booking transaction. ►*Web form understanding and mapping.* In order to automatically or interactively query deep web forms, wrappers have to learn the process of filling out complex web search forms and the usage of query interfaces. Such systems have to learn abstract representation for each search form and map them to a unified meta form and vice versa, taking into account different form element types, contents and labels.

Cross-references

- ►Business Intelligence
- ►Data Integration in Web Data Extraction System
- ►Data Mining
- ►Data Warehouse
- ►Datalog
- ►Deep-web Search
- ►Enterprise Application Integration
- ►Extraction, Transformation, and Loading
- ►Information Extraction
- ►Information Integration
- ►Logical Foundations Of Web Data Extraction
- ►MashUp
- ►Screen Scraper
- ►Semantic Web
- ►Service Oriented Architecture
- ►Snippet
- ►Web Harvesting
- ►Web Information Extraction
- ►Web Services
- ►Wrapper Induction
- ►Wrapper Maintenance
- ►Wrapper Stability
- ►XML
- ►Xpath/Xquery

Recommended Reading

1. Anupam V., Freire J., Kumar B., and Lieuwen D. Automating web navigation with the WebVCR. Comput. Network., 33(1–6):503–517, 2000.
2. Baumgartner R., Flesca S., and Gottlob G. Visual web information extraction with Lixto. In Proc. 27th Int. Conf. on Very Large Data Bases, 2001, pp. 119–128.
3. Crescenzi V., Mecca G., and Merialdo P. Road runner: towards automatic data extraction from large Web sites. In Proc. 27th Int. Conf. on Very Large Data Bases, 2001, pp. 109–118.
4. Etzioni O., Cafarella M.J., Downey D., Kok S., Popescu A., Shaked T., Soderland S., Weld D.S., and Yates Y. Web-scale information extraction in KnowItAll: (preliminary results). In Proc. 12th Int. World Wide Web Conference, 2004, pp. 100–110.

5. Gatterbauer W., Bohunsky P., Herzog M., Krüpl B., and Pollak B. Towards domain-independent information extraction from web tables. In Proc. 16th Int. World Wide Web Conference, 2007, pp.71–80.
6. Gottlob G. and Koch C. Monadic datalog and the expressive power of languages for web information extraction. J. ACM 51(1):74–113, 2002.
7. Gottlob G. and Koch C.A. Formal comparison of visual web wrapper generators. In Proc. 32nd Conf. Current Trends in Theory and Practice of Computer Science, 2006, pp. 30–48.
8. Kuhlins S. and Tredwell R. Toolkits for generating wrappers: a survey of software toolkits for automated data extraction from Websites. NODe 2002, LNCS:2591, 2003.
9. Kushmerick N., Weld D.S., and Doorenbos R.B. Wrapper induction for information extraction. In Proc. 15th Int. Joint Conf. on AI, 1997, pp. 729–737.
10. Laender A.H.F., Ribeiro-Neto B.A., and da Silva A.S. DEByE – data extraction by example. Data Knowl. Eng., 40(2):121–154, 2000.
11. Liu L., Pu C., and Han W. XWRAP: an XML-enabled wrapper construction system for web information sources. In Proc. 16th Int. Conf. on Data Engineering, 2000, pp. 611–621.
12. Liu B., Grossman R.L., and Zhai Y. Mining web pages for data records. IEEE Intell. Syst., 19(6):49–55, 2004.
13. Muslea I., Minton S., and Knoblock C.A. Hierarchical Wrapper Induction for Semistructured Information Sources. Autonom. Agents Multi-Agent Syst., 4(1/2):93–114, 2001.
14. Pan A., Raposo J., Álvarez M., Montoto P., Orjales V., Hidalgo J., Ardao L., Molano A., and Viña Á. The Denodo data integration platform. In Proc. 28th Int. Conf. on Very Large Data Bases, 2002.
15. Sahuguet A. and Azavant F. Building intelligent web applications using lightweight wrappers. Data Knowl. Eng., 36(3):283–316, 2001.

Web Data Mining

► Data, Text, and Web Mining in Healthcare

Web Directories

► Lightweight Ontologies

Web ETL

Oliver Frölich
Lixto Software GmbH, Vienna, Austria

Synonyms

ETL using web data extraction techniques

Definitions

As ETL (acronym for Extraction, Transformation and Loading) is a well-established technology for the extraction of data from several sources, their cleansing, normalization and insertion into a Data Warehouse (e.g., a Business Intelligence System), *Web ETL* stands for an ETL process where the external data to be inserted into the Data Warehouse is extracted from semi-structured Web pages (e.g., in HTML or PDF format) using Web Data Extraction techniques.

Particularly, back-end interchange of structured data just *using* the Web, e.g., two database systems exchanging data with Web EDI technology (EDI (Electronic Data Interchange) stands for techniques and standards for the transmission of structured data, for example over the Web, in an application-to-application context.), is not a Web ETL process as no semi-structured data needs to be transformed using Web Data Extraction techniques.

Key Points

Powerful and efficient tools supporting ETL processes (Extracting, Transforming and Loading of data) in a Data Warehouse context exist for years now. They concentrate on the data extraction from *internal* applications. But there is also a growing need to integrate *external* data: For example in Competitive Intelligence, information about competitor activities and market developments is becoming a more and more important success factor for enterprises. The largest information source on earth is the World Wide Web. Unfortunately, data on the Web requires intelligent interpretation and cannot be easily used by programs, since the Web is primarily intended for human users. Therefore, the extraction from semi-structured information sources like Web pages was mostly done manually and was very time consuming just a few years ago. Today, sophisticated toolsets for Web Data Extraction exist for retrieving relevant data automatically from Web sites and for transforming it into structured data formats. An example of such a toolset is the Lixto Suite [1], allowing for the extraction and transformation of data from Web pages into structured XML formats. This structured XML data can be integrated in Data Warehouse systems. This whole process from Web Data Extraction to the integration of the cleansed and normalized information is called a Web ETL process. Web ETL itself can be a part of a Business Intelligence process, turning semi-structured data into

business-relevant information in a Business Intelligence Data Warehouse. An example of a Business Intelligence application of Web ETL is given in [2].

Another application field of Web ETL besides Competitive Intelligence and Business Intelligence is Front-End System Integration, where Web interfaces of business applications are utilized for coupling these systems using Web ETL processes.

Cross-references

- ▶ Business Intelligence
- ▶ Competitive Intelligence
- ▶ Data Warehouse
- ▶ ETL
- ▶ Semi-Structured Data

Recommended Reading

1. Baumgartner R., Flesca S., Gottlob G. Visual web information extraction with Lixto. In Proc. 27th Int. Conf. on Very Large Data Bases, 2001, pp. 119–128.
2. Baumgartner R., Frölich O., Gottlob G., Harz P., Herzog M., and Lehmann P. Web data extraction for business intelligence: the Lixto approach, In Proc. Datenbanksysteme in Business, Technologie und Web (BTW), 2005, pp. 48–65.
3. Frölich O. Optimierung von Geschäftsprozessen durch Integrierte Wrapper-Technologien. Dissertation, Institute of Information Systems, Vienna University of Technology, 2006.

Web Harvesting

Wolfgang Gatterbauer
University of Washington, Seattle, WA, USA

Synonyms

Web data extraction; Web information extraction; Web mining

Definition

Web harvesting describes the process of gathering and integrating data from various heterogeneous web sources. Necessary input is an appropriate knowledge representation of the domain of interest (e.g., an *ontology*), together with example instances of concepts or relationships (*seed knowledge*). Output is structured data (e.g., in the form of a relational database) that is gathered from the Web. The term *harvesting* implies that, while passing over a large body of available information, the process gathers only such information that lies in the domain of interest and is, as such, relevant.

Key Points

The process of web harvesting can be divided into three subsequent tasks: (i) *data or information retrieval*, which involves finding relevant information on the Web and storing it locally. This task requires tools for searching and navigating the Web, i.e., crawlers and means for interacting with dynamic or deep web pages, and tools for reading, indexing and comparing the textual content of pages; (ii) *data or information extraction*, which involves identifying relevant data on retrieved content pages and extracting it into a structured format. Important tools that allow access to the data for further analysis are parsers, content spotters and adaptive wrappers; (iii) *data integration* which involves cleaning, filtering, transforming, refining and combining the data extracted from one or more web sources, and structuring the results according to a desired output format. The important aspect of this task is organizing the extracted data in such a way as to allow unified access for further analysis and data mining tasks.

The ultimate goal of web harvesting is to compile as much information as possible from the Web on one or more domains and to create a large, structured knowledge base. This knowledge base should then allow querying for information similar to a conventional database system. In this respect, the goal is shared with that of the Semantic Web. The latter, however, tries to solve extraction *à priori* to retrieval by having web sources present their data in a semantically explicit form.

Today's search engines focus on the task of finding content pages with relevant data. The important challenges for web harvesting, in contrast, lie in extracting and integrating the data. Those difficulties are due to the variety of ways in which information is expressed on the Web (*representational heterogeneity*) and the variety of alternative, but valid interpretations of domains (*conceptual heterogeneity*). These difficulties are aggravated by the Web's sheer size, its level of heterogeneity and the fact that information on the Web is not only complementary and redundant, but often contradictory too.

An important research problem is the optimal combination of automation (high recall) and human involvement (high precision). At which stages and in which manner a human user must interact with an otherwise fully automatic web harvesting system for optimal performance (in terms of speed, quality, minimum human involvement, etc.) remains an open question.

Cross-references

- ▶ Data Extraction
- ▶ Data Integration
- ▶ Fully-Automatic Web Data Extraction
- ▶ Information Retrieval
- ▶ Semantic Web
- ▶ Web Data Extraction
- ▶ Web Data Extraction System
- ▶ Web Scraper
- ▶ Wrapper

Recommended Reading

1. Ciravegna F., Chapman S., Dingli A., and Wilks Y. Learning to harvest information for the Semantic Web. In Proc. 1st European Semantic Web Symposium, 2004, pp. 312–326.
2. Crescenzi V. and Mecca G. Automatic information extraction from large websites. J. ACM, 51(5):731–779, 2004.
3. Etzioni O., Cafarella M.J., Downey D., Kok S., Popescu A.M., Shaked T., Soderland S., Weld D.S., and Yates A. Web-scale information extraction in KnowItAll: (preliminary results). In Proc. 12th Int. World Wide Web Conference, 2004, pp. 100–110.

Web Indexing

- ▶ Indexing the Web

Web Information Extraction

Rajasekar Krishnamurthy, Yunyao Li,
Sriram Raghavan, Frederick Reiss,
Shivakumar Vaithyanathan, Huaiyu Zhu
IBM Almaden Research Center, San Jose, CA, USA

Synonyms

Text Analytics; Information Extraction

Definition

Information extraction (IE) is the process of automatically extracting structured pieces of information from unstructured or semi-structured text documents. Classical problems in information extraction include *named-entity recognition* (identifying mentions of persons, places, organizations, etc.) and *relationship extraction* (identifying mentions of relationships between such named entities). Web information extraction is the application of IE techniques to process the vast amounts of unstructured content on the Web. Due to the nature of the content on the Web, in addition to named-entity and relationship extraction, there is growing interest in more complex tasks such as extraction of reviews, opinions, and sentiments.

Historical Background

Historically, information extraction was studied by the Natural Language Processing community in the context of identifying organizations, locations, and person names in news articles and military reports [9]. From early on, information extraction systems were based on the *knowledge engineering approach* of developing carefully crafted sets of *rules* for each task. These systems view the text as an input sequence of symbols and extraction rules are specified as regular expressions over the lexical features of these symbols. The formalism underlying these systems is based on cascading grammars and the theory of finite-state automata. One of the earliest languages for specifying such rules is the Common Pattern Specification Language (CPSL) developed in the context of the TIPSTER project [1]. To overcome some of the drawbacks of CPSL resulting from a sequential view of the input, the AfST system [2] uses a more powerful grammar that views its input as an object graph.

Beginning in the mid-1990s, as the unstructured content on the Web continued to grow, information extraction techniques were applied in building popular Web applications. One of the earliest such uses of information extraction was in the context of *screen scraping* for online comparison shopping and data integration applications. By manually examining a number of sample pages, application designers would develop ad hoc rules and regular expressions to eke out relevant pieces of information (e.g., the name of a book, its price, the ISBN number, etc.) from multiple Web sites to produce a consolidated Web page or query interface. Recently, more sophisticated IE techniques are being employed on the Web to improve search result quality, guide ad placement strategies, and assist in reputation management [7,12].

Foundations

Knowledge-engineered rules have the advantage that they are easy to construct in many cases (e.g., rules to recognize prices, phone numbers, zip codes, etc.), easier to debug and maintain when written in a high-level rule

language, and provide a natural way to incorporate domain or corpus-specific knowledge. However, such rules are extremely labor intensive to develop and maintain. An alternative paradigm for producing extraction rules is the use of learning-based methods [6]. Such methods work well when training data is readily available and the extraction tasks are hard to encode manually. Finally, there has been work in the use of complex statistical models, such as Hidden Markov Models and Conditional Random Fields, where the rules of extraction are implicit within the parameters of the model [11].

For ease of exposition, the rule-based paradigm is used to present the core concepts of Web information extraction. Rule-based extraction programs are called *annotators* and their output is referred to as *annotations*. The two central concepts of rule-based extraction are *rules* and *spans*. A span corresponds to a substring of the document text represented as a pair of offsets *(begin, end)*. A rule is of the form $A \leftarrow P$ (Fig. 1), where A is an annotation (specified within angled brackets) and P is a pattern specification. Evaluating the rule associates any span of text that matches pattern P with the annotation A.

The description of information extraction is organized around four broad categories of extraction tasks: *entity extraction*, *relationship extraction*, *complex composite extraction*, and *application-driven extraction*. The first two categories, while relevant and increasingly used in Web applications, are classical IE tasks that have been extensively studied in the literature even before the advent of the Web.

Category 1. *Entity extraction* refers to the identification of mentions of named entities (such as persons, locations, organizations, phone numbers, etc.) in unstructured text. While the task of entity extraction is intuitive and easy to describe, the corresponding annotators are fairly complex and involve a large number of rules and carefully curated dictionaries. For example, a high-quality annotator for person names would involve several tens of rules to capture all of the different conventions, shorthands, and formats used in person names all over the world.

Example 1. As an illustrative example, consider the simple annotator shown in Fig. 1 for recognizing mentions of person names. The annotator uses a CPSL-style cascading grammar specification. Assume that the input text has already been tokenized and is available as a sequence of ⟨`Token`⟩ annotations. Rules R_4, R_5, and R_6 are the lowest level grammar rules since they operate only on the input ⟨`Token`⟩ annotations. Each of these rules attempts to match the text of a token against a particular regular expression and produces output annotations whenever the match succeeds. Rule R_6 states that a span of text consisting of a single token whose text matches a dictionary of person names ("Michael," "Richard," etc.) is a ⟨`PersonDict`⟩ annotation. Rule R_4 similarly defines ⟨`Salutation`⟩ annotations (e.g., Dr., Prof., Mr., etc.) and R_5 defines annotations consisting of a single token beginning with a capital letter. Finally, rules R_1, R_2, and R_3 are the higher-level rules of the cascading grammar since they operate on the annotations produced by the lower-level rules. For example, R_1 states that a salutation followed by two capitalized words is a person name. Such a rule will recognize names such as "Dr. Albert Einstein" and "Prof. Michael Stonebraker."

Category 2. *Binary relationship extraction* refers to the task of associating pairs of named entities based on the identification of a relationship between the entities. For instance, Example 2 describes the task of extracting instances of the *CompanyCEO* relationship, i.e., finding pairs of person and organization names such that the person is the CEO of the organization.

Example 2. Assume that mentions of persons and organizations have already been annotated (as in Category 1 above) and are available as ⟨`Person`⟩ and ⟨`Organization`⟩ annotations respectively. Figure 2 shows rules that identify instances of the *CompanyCEO* relationship. Rule R_7 looks for pairs of ⟨`Person`⟩ and ⟨`Organization`⟩ annotations such that the text

⟨Person⟩ ← ⟨Salutation⟩ ⟨CapsWord⟩ ⟨CapsWord⟩ (R_1)
⟨Person⟩ ← ⟨PersonDict⟩ ⟨PersonDict⟩ (R_2)
⟨Person⟩ ← ⟨CapsWord⟩ ⟨PersonDict⟩ (R_3)
⟨Salutation⟩ ← ⟨Token⟩ [≃"Mr. |Mrs. |Dr. | ..."] (R_4)
⟨CapsWord⟩ ← ⟨Token⟩ [≃text() ≃ "[A-Z] [a-z] *"] (R_5)
⟨PersonDict⟩ ← ⟨Token⟩ [text() ≃ "Michael |Richard |Smith| ...."] (R_6)

Web Information Extraction. Figure 1. Simple rules for identifying person names.

between these annotations satisfies a particular regular expression. The regular expression that is used in this example will match any piece of text containing the phrase "CEO of" with an optional comma before the phrase and an arbitrary amount of whitespace separating the individual words of the phrase. Thus, it will correctly extract an instance of this relationship from the text "Sam Palmisano, CEO of IBM." As with Example 1, the rules presented here are merely illustrative and high-quality relationship annotators will involve significantly more rules.

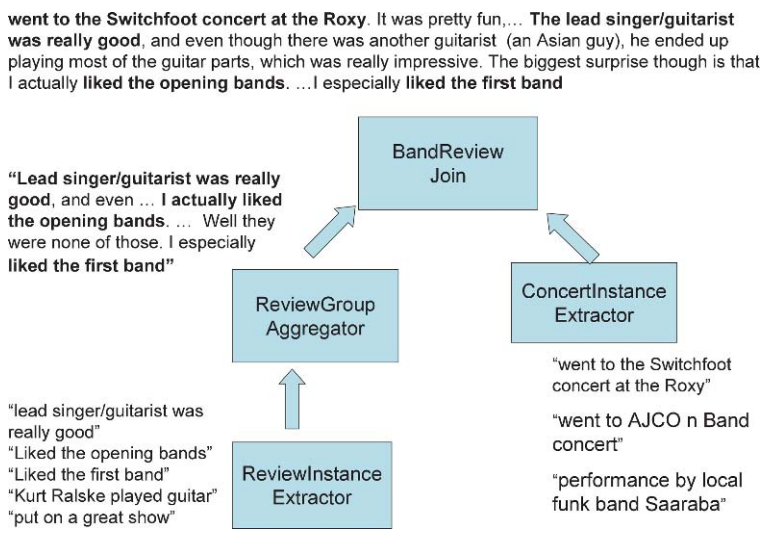
Category 3. *Complex composite extraction.* The presence of vast amounts of user generated content on the Web (in blogs, wikis, discussion forums, and mailing lists) has engendered a new class of complex information extraction tasks. The goal of such tasks is the extraction of reviews, opinions, and sentiments on a wide variety of products and services. Annotators for such complex tasks are characterized by two main features: (i) the use of entity and relationship annotators as sub-modules to be invoked as part of a higher level extraction workflow, and (ii) the use of aggregation-like operations in the extraction process.

Example 3. Consider the task of identifying informal reviews of live performance of music bands embedded within blog entries. Figure 3 shows the high-level workflow of an annotator that accomplishes this task. The two individual modules, *ReviewInstance Extractor* and *ConcertInstance Extractor*, identify specific snippets of text in a blog. The ⟨ReviewInstance⟩ module identifies snippets that indicate portions of a concert review – e.g., "show was great," "liked the opening bands" and "Kurt Ralske played guitar." Similarly, the *ConcertInstance Extractor* module identifies occurrences of bands or performers – e.g., "performance by the local funk band Saaraba" and "went to the Switchfoot concert at the Roxy." The output from the *ReviewInstance Extractor* module is fed into the *ReviewGroup Aggregator* module to identify contiguous blocks of text containing ⟨ReviewInstance⟩ snippets. Finally, a ⟨ConcertInstance⟩ snippet is associated with one or more ⟨ReviewGroups⟩ to obtain ⟨BandReviews⟩. Note that in addition to the complex high-level workflow, each individual module in Fig. 3 is itself fairly involved and consists of tens of regular expression patterns and dictionaries.

Category 4. *Application-driven extraction.* The last category of extraction tasks covers a broad spectrum of scenarios where IE techniques are applied to perform extraction that is unique to the needs of a particular

⟨CompanyCEO⟩ ← ⟨Person⟩ RegExp(" \s+,?\s+CE0\s+of\s+ ") ⟨Organization⟩ (R_7)
⟨CompanyCEO⟩ ← ⟨Organization⟩ RegExp(" \s+CE0\s+ ") ⟨Person⟩ (R_8)

Web Information Extraction. Figure 2. Simple rules for identifying CompanyCEO relationship instances.



Web Information Extraction. Figure 3. Extraction of informal band reviews.

Web application. While the early applications of IE were ad hoc and limited to screen scraping, there has been recent widespread use of IE in the context of improving search quality. A prototypical example is the use of specially crafted extraction rules applied to the URLs and titles of Web pages to identify high-quality index terms. Consider the following URLs:

U_1 http://en.wikipedia.org/wiki/Michelangelo
U_2 http://www.ibiblio.org/wm/paint/auth/michelangelo/
U_3 http://www.michelangelo.com/
U_4 http://www.artcyclopaedia.com/artists/michelangelo_buonarotti.html

It is easy to see that certain key segments of the URL string, such as the last segment of the path (as in U_1 and U_2), the portion of the hostname following the "www" (in U_3) and the actual resource name (as in U_4) provide fairly reliable clues as to the actual content of the corresponding Web page. Modern search engines, on the Web and in the intranet, are applying sophisticated patterns to extract such key segments from the URL and produce terms for their search index. In the above examples, such a technique would enable the term "michelangelo" or the phrase "michelangelo buonarotti" to be identified as high-quality index terms for the Web pages. In a similar fashion, extraction of key phrases from the captions of images and videos are being used to effectively index and search over online multimedia repositories. Other specialized search engines for verticals such as healthcare, finance, and people search, also make significant use of information extraction to identify domain-specific concepts.

Key Applications

As the vast majority of information on the Web is in unstructured form, there is growing interest, within the database, data mining, and IR communities, in the use of information extraction for Web applications. Several research projects in the areas of *intranet search*, *community information management*, and *Web analytics*, are already employing IE techniques to bring order to unstructured data. The common theme in all of these applications is the use of IE to process the input text and produce a structured representation to support search, browsing, and mining applications.

Web search engines are also employing IE techniques to recognize key entities (persons, locations, organizations, etc.) associated with a web page. This semantically richer understanding of the contents of a page is used to drive corresponding improvements to their search ranking and ad placement strategies.

Finally, there is continuing interest in techniques to reliably extract reviews, opinions, and sentiments about products and services from the content present in online communities and portals. The extracted information can be used to guide business decisions related to product placement, targeting, and marketing. The complex nature of these extraction tasks as well as the heterogeneous and noisy nature of the data pose interesting research challenges.

Future Directions

While the area of information extraction has made considerable progress since inception, several important challenges still remain open. Two of these challenges that are under active investigation in the research community are described below.

Scalability

As complex information tasks move to operating over Web-size data sets scalability has become a major focus [8]. In particular, IE systems need to scale to large numbers of input documents (data set size) and to large numbers of rules and patterns (annotator complexity). Two classes of optimizations being considered in this regard are

- Low-Level Primitives: Improving the performance of the low-level operations that dominate annotator execution time.
- High-Level Optimization: Automatically choosing more efficient orders for evaluating annotator rules.

Low-Level Primitives In many information extraction systems, low-level text operations like tokenization, regular expression evaluation and dictionary matching dominate execution time. Speeding up these low-level operations leads to direct improvements in scalability, both in terms of the number of documents the system can handle and the number of basic operations the system can afford to perform on a given document [5]. Some of the problems being addressed currently in this context are given below.

- When a large number of dictionaries are evaluated in an annotator (e.g., the *BandReview* annotator (Fig. 3) evaluates over 30 unique dictionaries),

evaluating each dictionary separately is expensive as the document text is tokenized once per dictionary evaluation. To address this inefficiency, techniques are being explored to evaluate multiple dictionaries efficiently in a single pass over the document text.

- Evaluating complex regular expressions over every document is an expensive operation. An active area of research is the design of faster regular expression matchers for special classes of regular expressions. Another approach being considered is the design of "filter" regular expression indexes. These indexes can be used to quickly eliminate many of the documents that do not contain a match.

High-Level Optimization As web information extraction moves towards Categories 3 and 4 (more complex tasks) there is more opportunity for improving efficiency [3–14]. Some of these efficiency gains can be obtained using traditional relational optimizations such as "join reordering" and "pushing down selection predicates." There are also additional text-specific optimizations that may give significant benefit, two of which are described below.

- *Restricted Span Evaluation*: Let R denote a set of rules for a given information extraction task. Let $spans(d, R')$be a set of spans obtained by evaluating $R' \subset R$ over a document d. Let $r \in (R - R')$ be a rule to be evaluated over the complete text of d. Restricted Span Evaluation is the optimization technique by which theannotator specification can be rewritten so that rule r is evaluated only on $spans(d, R')$and not over the complete text of d.

  A specific instantiation of restricted span evaluation is presented below, using Rule R_1 in the *Person* annotator (Fig. 1). This rule identifies a salutation followed by two capitalized words. Note that R_1 uses R_4 and R_5 for identifying ⟨`Salutation`⟩ and ⟨`CapsWord`⟩ respectively. A naive evaluation strategy identifies all occurrences of ⟨`Salutation`⟩ and ⟨`CapsWord`⟩ over the complete document before R_1 is evaluated. An alternative approach is to first identify ⟨`Salutation`⟩ in the document and then look for ⟨`CapsWord`⟩ only in the immediate vicinity of all the ⟨`Salutation`⟩ annotations. The latter approach evaluates the ⟨`CapsWord`⟩ rule only on a smaller amount of text as the ⟨`Salutation`⟩ annotations occur infrequently. This can result in considerable performance improvements.
- *Conditional Evaluation*: Let R denote a set of rules for a given information extraction task and R_1 and R_2 be two non-overlapping subsets of R. Conditional Evaluation is the optimization technique by which the annotatorspecification can be rewritten so that R_1 is conditionally evaluated on a document d depending on whether the evaluation of R_2 over $d$satisfies certain predicates. An example predicate is whether the evaluationof R_2 produces at least one annotation.

  A specific instantiation of Conditional Evaluation in the context of the *BandReview* annotator (Fig. 3) is given below. There are two main modules in the annotator, *ConcertInstance* and *ReviewGroup* which can be conditionally evaluated – i.e., the absence of one in a document implies that the other need not be evaluated on that document. In general, Conditional Evaluation is applicable in large workflows at join conditions such as the *BandReview Join* in Fig. 3.

Initial results on combining efficient evaluation of lower-level primitives and higher-level optimizations are very encouraging and recent results have reported an order of magnitude improvement in execution time [13,14].

Uncertainty Management

Real-world annotators typically consist of a large number of rules with varying degrees of accuracy. For example, most annotations generated by Rule R_1 (Fig. 1) are likely to be correct since salutation is a very strong indicator of the existence of a person name. Rule R_3, on the other hand, will identify some correct names such as "James Hall" and "Dan Brown" along with some spurious annotations such as "Town Hall" and "Dark Brown." Since "Hall" and "Brown" are ambiguous person names that also appear in other contexts and these dictionary matches are combined with a capitalized word, R_3 has a lower accuracy than R_1. Formally this difference in accuracy between different rules can be captured by the notion of the *precision* of a rule.

Rule Precision Suppose rule R identifies N annotations over a given document corpus, of which $n_{correct}$ annotations are correct. Then the precision of rule R is the fraction $\frac{n_{correct}}{N}$. Each annotation a is associated with

a confidence value that is directly related to the precision of the corresponding rule.

The confidence value associated with an annotation enables applications to meaningfully represent and manipulate the imprecision of information extracted from text [4]. For Category 1 tasks, the associating of such confidence values (such as probabilities) has been addressed in existing literature. On the other hand, associating confidence values with annotations for more complex extraction tasks is still open [10]. To illustrate, consider Rule R_6 in Fig. 2 for identifying the *CompanyCEO* relationship. This rule uses existing annotations ⟨Person⟩ and ⟨Organization⟩ . Therefore the confidence of a *CompanyCEO* annotation is dependent not only on the precision of rule R_6 but also on the confidences of the ⟨Person⟩ and ⟨Organization⟩ annotations. By modeling the confidence numbers as probabilities it is possible to view the individual rules as queries over a probabilistic database. However, direct application of current probabilistic database semantics (possible worlds) is precluded as illustrated in the following example. Rule R_7 identifies an instance of the *CompanyCEO* relationship from the text "Interview with Lance Brown, CEO of PeoplesForum.com." This rule looks for the existence of the text "CEO of" between ⟨Person⟩ and ⟨Organization⟩ thereby making it a highly accurate rule. Even though the participating ⟨Person⟩ annotation has a low confidence (identified by a rule with low precision, Rule R_3), the resulting ⟨CompanyCEO⟩ annotations should have high confidence due to the high precision of Rule R_7. Such behavior cannot be modeled under "possible worlds semantics" where the confidence of ⟨CompanyCEO⟩ annotations is bounded by the confidences of the input ⟨Person⟩ and ⟨Organization⟩ annotations. Handling such anomalies in the calculation of probabilities for relationship and complex composite extraction tasks is an open problem.

Cross-references

- ▶ Fully Automatic Web Data Extraction
- ▶ Information Extraction
- ▶ Languages for Web Data Extraction
- ▶ Metasearch Engines
- ▶ Probabilistic Databases
- ▶ Query Optimization
- ▶ Text Analytics
- ▶ Text Mining
- ▶ Uncertainty and Data Quality Mgmt
- ▶ Web Advertising
- ▶ Web Data Extraction System
- ▶ Web Harvesting
- ▶ Wrapper Induction

Recommended Reading

1. Appelt D.E. and Onyshkevych B. The common pattern specification language. In Tipster Text Program Phase 3, 1999.
2. Boguraev B. Annotation-based finite state processing in a large-scale NLP architecture. In Proc. Recent Advances in Natural Language Processing. 2003, pp. 61–80.
3. Cafarella M.J. and Etzion O. A search engine for natural language applications. In Proc. 14th Int. World Wide Web Conference, 2005, pp. 442–452.
4. Cafarella M.J. et al. Structured querying of Web text: A technical challenge. In Proc. 3rd Biennial Conf. on Innovative Data Systems Research, 2007, pp. 225–234.
5. Chandel A., Nagesh P.C., and Sarawagi S. Efficient batch top-k search for dictionary-based entity recognition. In Proc. 22nd Int. Conf. on Data Engineering, 2006.
6. Cohen W. and McCallum A. Information extraction from the world wide web Tutorial at Proc. 9th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, 2003.
7. Cunningham H. Information Extraction, Automatic. In Encyclopedia of Language and Linguistics, 2nd edn. 2005.
8. Doan A., Ramakrishnan R., and Vaithyanathan S. Managing information extraction: state of the art and research directions. Tutorial in Proc. ACM SIGMOD Int. Conf. on Management of Data, 2006.
9. Grishman R. and Sundheim B. Message understanding conference-6: a brief history. In Proc. 16th Conf. on Computational, 1996, pp. 446–471.
10. Jayram T.S., Krishnamurthy R., Raghavan S., Vaithyanathan S., and Zhu H. Avatar information extraction system. Q. Bull. IEEE TC on Data Engineering, 29(1):40–48, May 2006.
11. Lafferty J., McCallum A., and Pereira F. Conditional random fields: probabilistic models for segmenting and labeling sequence data. In Proc. 18th Int. Conf. on Machine Learning, 2001, pp. 282–289.
12. Li Y., Krishnamurthy R., Vaithyanathan S., and Jagadish H. Getting work done on the web: supporting transactional queries. In Proc. 32nd Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 2006, pp. 557–564.
13. Reiss F., Raghavan S., Krishnamurthy R., Zhu H., and Vaithyanathan S. An algebraic approach to rule-based information extraction. In Proc. 24th Int. Conf. on Data Engineering, 2008, pp. 933–942.
14. Shen W., Doan A., Naughton J., and Ramakrishnan R. Declarative information extraction using datalog with embedded extraction predicates. In Proc. 33rd Int. Conf. on Very Large Data Bases, 2007, pp. 1033–1044.

Web Information Extraction System

- ▶ Web Data Extraction System

Web Information Integration and Schema Matching

► Data Integration in Web Data Extraction System

WEB Information Retrieval Models

Craig MacDonald, Iadh Ounis
University of Glasgow, Glasgow, UK

Synonyms

Web search engines

Definition

The Web can be considered as a large-scale document collection, for which classical text retrieval techniques can be applied. However, its unique features and structure offer new sources of evidence that can be used to enhance the effectiveness of Information Retrieval (IR) systems. Generally, Web IR examines the combination of evidence from both the textual content of documents and the structure of the Web, as well as the search behavior of users and issues related to the evaluation of retrieval effectiveness in the Web setting.

Web Information Retrieval models are ways of integrating many sources of evidence about documents, such as the links, the structure of the document, the actual content of the document, the quality of the document, etc. so that an effective Web search engine can be achieved. In contrast with the traditional library-type settings of IR systems, the Web is a hostile environment, where Web search engines have to deal with subversive techniques applied to give Web pages artificially high search engine rankings. Moreover, the Web content in heavily duplicated, for example by mirroring, which search engines need to account for, while the size of the Web requires search engines to address the scalibility of their algorithms to create efficient search engines.

The commonly known PageRank algorithm – based on a documents hyperlinks – is an example of a source of evidence about a document to identify high quality documents. Another example is the commonly applied anchor text of the incoming hyperlinks. These sources of evidence are two of the most popular examples of link-based sources of evidence.

Historical Background

The first search engines for the Web appeared around 1992–1993, notably with the full-text indexing WebCrawler and Lycos both arriving in 1994. Soon after, many other search engines arrived, including Altavista, Excite, Inktomi and Northern Light. These often competed directly with directory-based services, like Yahoo!, which added search engine facilities later.

The rise in prominence of Google, in 2001, was due to its recognition that the underlying Web search user task is not an adhoc task (where users want lots of relevant documents, but not any ones in particular) to a more precision oriented task, where the relevance of the top-ranked result is important. This set Google apart from other search engines, since by using link analysis techniques (such as PageRank), hyperlink anchor text and other heuristics such as the title of the page, it could find itself at rank #1 for the query "google." This is something none of the other search engines at that time could achieve.

Since then, there has been a distinct consolidation in the Web search engine market, with only three major players taking the majority of the English market: Google, Yahoo! and MSN Live. However, additional search engines are thriving in other areas: Baidu and Yandex have high penetration in the Chinese and Russian markets respectively, while Technorati, BlogPulse and other blog search engines are popular in the blogosphere. However, since the arrival of Google, the Web IR research field has become much more active, with many more research groups, and more conferences than before, to the point that it is almost a separate research field in its own right, with two separate ACM conferences (World Wide Web Conference, and Web Search and Data Mining Conference), in addition to the existing IR conferences.

Foundations

The Web can be considered as a large-scale document collection, for which classical text retrieval techniques can be applied. However, its unique features and structure offer new sources of evidence that can be used to enhance the effectiveness of IR systems. Generally, Web IR examines the combination of evidence from both the textual content of documents and the structure of the Web, as well as the search behavior of users, and issues related to the evaluation of retrieval effectiveness.

The information available on the Web is very different from the information contained in either

libraries or classical IR collections. A large amount of information on the Web is duplicated, and content is often mirrored across many different sites. Moreover, many documents can be unintentionally, or intentionally inaccurate, or are intended to mislead search engines since this is their purpose. This means that Web IR models need to be developed so they can perform well in such a hostile environment.

The Web is based on a hypertext document model, where documents are connected with direct hyperlinks. This results in a virtual network of documents. A major development in the Web IR field was that of query independent evidence. In particular, Page et al's PageRank algorithm [1] determines the quality of a document by examining the quality of the documents linked to it. In particular, the PageRank scores correspond to the probability of visiting a particular node in a Markov chain for the whole Web graph, where the states represent Web documents, and the transitions between states represent hyperlinks. PageRank was reported to be a fundamental component of the early versions of the Google search engine [1], and is beneficial in high-precision user tasks, where the relevance and quality of the top-ranked documents are important.

Many other similar link analysis algorithms have been proposed, including those that can be applied in a query-dependent or independent fashion. Most are based on random-walks, calculating the probability of a random Web user visiting a given page. Examples are Kleinberg's HITS [6] and the Absorbing Model [13].

Other sources of document quality have been reported, including the use of the URL evidence to determine the type of the page (for instance, whether the URL is short or long, or how many "/" characters it contains [7]).

The integration of such query independent evidence into the ranking strategy is an important issue. In the language modeling framework, it is natural to see the integration of query independent evidence as a document prior, that defines the overall likeliness of the document's retrieval [7]. Craswell et al. [3] introduce a framework (FLOE) for integrating query independent evidence with the retrieval score from BM25. The integration of several priors remains an important problem. Peng et al. [10] propose a probabilistic mehod of combining several document priors in the language modeling framework, while Plachouras [11] examines how various query independent evidence can be applied on a query-by-query basis.

The algorithms HITS and PageRank, along with their extensions, explicitly employ the hyperlinks between Web documents, in order to find high quality, or authoritative Web documents. A form of implicit use of the hyperlinks in combination with content analysis is to use the anchor text associated with the incoming hyperlinks of documents. Web documents can be represented by an anchor text surrogate, which is formed from collecting the anchor text associated with the hyperlinks pointing to the document. The anchor text of the incoming hyperlinks provides a concise description for a Web document. The used terms in the anchor text may be different from the ones that occur in the document itself, because the author of the anchor text is not necessarily the author of the document. Indeed, Craswell et al. [2] showed that anchor text is very effective for navigational search tasks and more specifically for finding home pages of Web sites.

Several models have since been developed that use the anchor text of a document in addition to the content. Kraaij et al. [7] and Ogilvie and Callan [9] describe mixture language modeling approaches, where the probability of a term's occurrence in a document is the mixture of the probability of its occurrence in different textual representations of the document (e.g., content, title, anchor text).

Later, Robertson et al. [14] showed that due to the different term occurrence distributions of the different representations of a document, it is better to combine frequencies rather than scores. Indeed, shortly thereafter, Zaragoza et al. [16] and Macdonald et al. [8] devised weighting models where the frequency of a term occurring in each of a document's representations is normalized and weighted before scoring by the weighting model. This allows a fine-grained control over the importance of each representation of the document in the document scoring process. This has been further investigated by the use of multinomial Divergence from Randomness models to score structured documents [12].

Key Applications

Web Search Engines are heavy developers and users of Web IR technology. Yet, to prevent exploitation by spammers and imitation by commercial rivals, the models applied by the search engines remain closely-guarded secrets. However, in recent years, the IR research field has grown, and many groups are now researching topics related to Web IR. The TREC forum,

discussed below, is a facilitator of much of this research, providing samples of the Web for research purposes

Future Directions

There are many open problems in Web IR research. In particular, spam is an ever-growing issue – many sites are created with the purpose of manipulating search engine rankings for financial gain (e.g., advertising revenue). Search engines are in a constant battle with the spammers, and are constantly becoming more robust in adversial conditions.

Large Web search engines have often identified a huge number of features for every web page. Indeed, recent reports from the Microsoft Live search engine suggest that they use over 300 features when ranking web pages. At this scale, how should these features be combined? This area of interest has been recently gaining much interest from the IR and machine learning communities, and was the subject of the Learning To Rank for Information Retrieval workshop [5]. Generally, speaking, the idea is that machine learning techniques can be applied to learn a function from the input data and the evaluation results, instead of developing a function from a theory. The Learning to Rank sub-field also encompasses the large-scale learning that search engines perform using the click-through data gleaned from user sessions, allowing them to learn the best results for popular queries.

The advent of the blogging phenomenon has created many new and interesting search and data mining tasks for search engines in the era of user-generated content. The blogging community (known as the blogosphere) often responds to real-life events with comment and rhetoric. The TREC Blog track has been created to investigate retrieval issues in this niche area of the Web, and has thus far investigated the retrieval of opinionated posts – finding blog posts which not only discuss a target, but also disseminate an opinion about the target.

As Web search becomes ubiquitious for the average knowledge worker, the need for internal company search engines to allow employees to search and mine the company intranet are becoming important. To this end, the TREC Enterprise track has been investigating retrieval tasks in the Enteprise setting.

Data Sets

Much research into techniques for Web IR has centered around the Text REtrieval Conference (TREC) Web track and other related tracks. In particular, TREC has investigated retrieval from various samples of the Web, and produced several test collections. Each test collection consists of a corpus of Web documents crawled from the Internet, combined with queries with information needs (known as topics), and relevance assessments.

Table 1 introduces the years that Web tasks ran at TREC. The document corpora used are described further in Table 2. It is of note over the early years of the Web tracks, the tasks were being developed, and hence in the early years, the user tasks are not the most common task that users perform in search engines (e.g., Adhoc retrieval). Instead, the focus on early precision and known-item retrieval tasks (e.g., Home page and Named page finding) were developed from studies of user interactions with search engines [15].

Table 2 describes the collections used for the TREC Web and Terabyte tracks, over the years 1999–2006. When developing Web IR techniques, where it is appropriate, it is common to apply the techniques on these

WEB Information Retrieval Models. Table 1. Tasks and collections applied by various Web IR tracks at TREC

TREC Track	Collection	Tasks
2006 Terabyte	.GOV2	Adoc, Named page
2005 Terabyte	.GOV2	Adhoc, Named page
2004 Terabyte	.GOV2	Adhoc
2004 Web	.GOV	Mixed (Home page, Named page, Topic Distillation)
2003 Web	.GOV	Home page, Named page, Topic Distillation
2002 Web	.GOV	Named page, Topic Distillation
2001 Web	WT10G	Adhoc, Home page
2000 Web	WT10G	Adhoc
1999 Web	WT2G	Adhoc

WEB Information Retrieval Models. Table 2. Web IR research test collections

Collection	# Documents	# Links
.GOV2	25,205,179	261,937,150
.GOV	1,247,753	11,110,989
WT10G	1,692,096	8,063,026
WT2G	247,491	1,166,146

standard test collections, and compare to appropriately trained baselines. Hawking and Crasswell [4] contains an overview of the Web track at TREC. Relatedly, TREC also has two test collections for Enterprise IR research, and a further collection for Blog IR research.

URL to Code

Text REtrieval Conference: http://trec.nist.gov Web and Blog Test Collections: http://ir.dcs.gla.ac.uk/test_collections ACM Special Interest Group IR: http://www.sigir.org/

Cross-references

- ▶ BM25
- ▶ Divergence from Randomness Models
- ▶ Indexing the Web
- ▶ Information Retrieval Models
- ▶ Probabilistic Retrieval Models and Binary Independence Retrieval (BIR) Model
- ▶ Web Indexing
- ▶ Web Page Quality Metrics
- ▶ Web Spam Detection

Recommended Reading

1. Brin S. and Page L. The anatomy of a large-scale hypertextual Web search engine. Comput. Netw. ISDN Syst., 30(1–7): 107–117, 1998.
2. Craswell N., Hawking D., and Robertson S. Effective site finding using link anchor information. In Proc. 24th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 2001, pp. 250–257.
3. Craswell N., Robertson S., Zaragoza H., and Taylor M. Relevance weighting for query independent evidence. In Proc. 31st Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 2005, pp. 416–423.
4. Hawking D. and Craswell N. The very large collection and Web tracks. In TREC: Experiment and Evaluation in Information Retrieval. Kluwer Academic Publishers, Dordrecht, 2004, pp. 199–232.
5. Joachims T., Li H., Liu T.Y., and Zhai C. SIGIR workshop report: learning to rank for information retrieval (LR4IR 2007). SIGIR Forum, 41(2):55–62, 2007.
6. Kleinberg J.M. Authoritative sources in a hyperlinked environment. J. ACM, 46(5):604–632, 1999.
7. Kraaij W., Westerveld T., and Hiemstra D. The importance of prior probabilities for entry page search. In Proc. 25th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 2002, pp. 27–34.
8. Macdonald C., Plachouras V., He B., Lioma C., and Ounis I. University of Glasgow at WebCLEF 2005: Experiments in per-field normlisation and language specific stemming. In Proc. 6th Workshop, Cross-Language Evaluation Forum, 2005, pp. 898–907.
9. Ogilvie P. and Callan J. Combining document representations for known-item search. In Proc. 26th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 2003, pp. 143–150.
10. Peng J., Macdonald C., He B., and Ounis I. Combination of document priors in Web information retrieval. In Proc. 8th Int. Conf. Computer-Assisted Information Retrieval, 2007.
11. Plachouras V. Selective Web Information Retrieval. PhD thesis, Department of Computing Science, University of Glasgow, 2006.
12. Plachouras V. and Ounis I. Multinomial randomness models for retrieval with document fields. In Proc. 29th European Conf. on IR Research, 2007, pp. 28–39.
13. Plachouras V., Ounis I., and Amati G. The static absorbing model for the Web. J. Web Eng., 165–186, 2005.
14. Robertson S., Zaragoza H., and Taylor M. Simple BM25 extension to multiple weighted fields. In Proc. Int. Conf. on Information and Knowledge Management, 2004, pp. 42–49.
15. Silverstein C., Henzinger M., Marais H., and Moricz M. Analysis of a very large AltaVista Query Log. Technical Report 1998-014, Digital SRC, 1998.
16. Zaragoza H., Craswell N., Taylor M., Saria S., and Robertson S. Microsoft cambridge at TREC-13: Web and HARD tracks. In Proc. the 4th Text Retrieval Conf., 2004.

Web Macros

- ▶ Web Data Extraction System

Web Mining

- ▶ Data, Text, and Web Mining in Healthcare
- ▶ Languages for Web Data Extraction
- ▶ Web Harvesting

Web Mashups

Maristella Matera
Politecnico di Milano University, Milan, Italy

Synonyms

Composite Web applications

Definition

Web mashups are innovative Web applications, which rely on heterogeneous content and functions retrieved from external data sources to create new composite services.

Key Points

Web mashups characterize the second generation of Web applications, known as Web 2.0. They are composite applications, usually generated by combining content, presentation, or application functionality from disparate Web sources.

Typical components that may be mashed-up, i.e., composed, are RSS/Atom feeds, Web services, programmable APIs, and also content wrapped from third party Web sites. Components may have a proper user interface that can be reused to build the interface of the composite application, they may provide computing support, or they may just act as plain data sources. Content, presentation and functionality, as provided by the different components, are then combined in disparate ways: via JavaScript in the browser, via server-side scripting languages (like PHP), or via traditional languages like Java.

Cross-references

► Visual Interaction
► Web 2.0/3.0

Web Ontology Language

► OWL: Web Ontology Language

Web Page Quality Metrics

Ravi Kumar
Yahoo Research, Santa Clara, CA, USA

Synonyms

Link analysis

Definition

The primary mission of web search engines is to obtain the best possible results for a given user query. To accomplish this effectively, they rely on two crucial pieces of information: the relevance of a web page to the query and some aspect of the quality of the web page that is independent of the query. Relevance, the extent to which the query matches the content of the web page, is formalized and extensively studied in the field of information retrieval. Quality, on the other hand, is more nebulous and less well-defined. Nevertheless, one can identify three concrete and somewhat complementary aspects to the quality of a web page. The first is based on the absolute goodness of the web page and its associated meta-data. This might depend on a variety of parameters, including the worth of content that exists on the web page, the reputation of the person who authored the web page, the importance of the web site that hosts the web page, and so on. The second way is to focus on the quality of the web page as perceived by other pages that exist on the web. This can be captured by links that point to the web page: a hyperlink from one web page to another can be viewed as an endorsement of the latter by the former. The third way to measure quality is to focus on the how the web users perceive this page. This can be realized by studying the traffic on the web page. The main focus of this entry will be the second aspect of web page quality.

Historical Background

Web search engines were originally built using mainly classical information retrieval techniques, which were used to calculate the relevance of a query to a web page. The techniques were often modified in simple ways to account for the fact that most of the content on the web was written in HTML; for example, the presence of a query term in the title of the web page can connote a higher relevance. These methods served well in the early days of the web, but became increasingly inadequate as the web grew. One reason was they failed to take into account a distinctive feature of web pages: the presence of explicit hyperlinks created by people to link one document to another.

At a coarse level, the role of hyperlinks in web pages is two-fold. The first is to aid in the browsability and navigability from one web page to another on a web site. The second is to refer to outside sources of information that are considered relevant and authoritative by the author of the web page. Independently and almost concurrently, Brin and Page [4] and Kleinberg [12] came up with different methods to exploit the presence of links in order to ascribe a notion of quality to a web page, in the context of a collection of web pages with hyperlinks between them. The idea of using links between documents to infer a quality measure has precedents in the field of bibliometrics. For example, *impact factor* was proposed by Garfield [7] to measure the quality of scientific journals. It was defined as the average number of citations to the journal. *Bibliographic coupling* was proposed by Kessler [11] to measure similarity of

two hyperlinked documents. It was defined as the total number of documents linked to by both documents. An alternate measure for document similarity was proposed by Small [15]: count of the total number of documents that have a link to both the documents.

The above measures treat all links alike and do not account for the quality of the citing document. A pervading theme in the works of Brin and Page and Kleinberg is to treat different links differently. This simple principle has proved to be very effective in practice in terms of improving web search and has found tremendous use in commercial search engines, even beyond web search.

Foundations

For the remainder of the entry, it is convenient to view the web as a directed graph or as a matrix. The web pages are nodes in this graph and the hyperlinks from one web page to another constitute the directed edges. This graph defines a natural adjacency matrix M whose (p,q)-th entry M_{pq} is 1 if and only if page p has a hyperlink to page q and 0 otherwise.

In 1998, Brin and Page [4] proposed a simple iterative method to define quality of a web page. Their method, called *PageRank*, used the hyperlinks to arrive at the quality. One way to think about this method is to focus on the following browsing behavior of a web user. Most of the times, the user visits a web page and after browsing, chooses one of the hyperlinks on the page at random to select the next page to browse. There is also a small chance that the user abandons browsing the current page entirely and chooses a random web page to continue browsing.

This behavior can be formalized as follows. Each page p has a non-negative value $\pi(p)$, updated iteratively. A stochastic process can be defined to abstract the user behavior, and in the limit, $\pi(p)$ will be the fraction of the time spent at page p by the process. Let $\alpha \in (0,1)$ be a fixed constant. At each step of the process, with probability α, the process jumps to a page chosen uniformly at random from the entire set of web pages. With the remaining probability $1 - \alpha$, the process jumps to a web page q chosen uniformly at random from the set of web pages to which page p has a hyperlink. If p has no hyperlinks, then the process jumps to a web page chosen uniformly at random from the entire set of pages. Observe that this process can be modeled by the iteration $\vec{\pi}_{t+1} = P^T\vec{\pi}_t$ with $P = (1 - \alpha)M' + \alpha J$. Here, M' is a stochastic matrix corresponding to the above process, J is the matrix where each entry is inverse of the total number of nodes in the graph, and α is the teleportation probability, usually chosen around 0.15. Notice that M' can be easily derived from the adjacency matrix M.

Since the above iteration corresponds to a linear system, it converges to a fixed-point $\vec{\pi} = \lim_{t\to\infty}\vec{\pi}_t$, which is the principal eigenvector of P^{T}. The quality of a page p is given by the value $\pi(p)$.

Many variants and extensions of the basic PageRank method have been proposed. A particularly interesting extension, called topic-sensitive PageRank, was proposed by Haveliwala [10]. The goal of this method is to define a quality measure that is biased with respect to a given topic. First, a subset of pages that correspond to the topic is identified; this can be done, for instance, by building a suitable classifier for the topic. Next, the basic PageRank process is modified in the following manner. Instead of jumping with probability α to a random page chosen from the entire set of web pages, the process jumps to a page chosen uniformly at random from this subset of pages. As before, the iteration converges and a quality measure for each page that is specific to this topic can be obtained.

Also in 1998, Kleinberg [12] proposed a different iterative method to define a quality of a web page. His method, called *HITS*, is aimed at finding the most authoritative sources in a given collection of web pages. The intuition behind HITS is the following. Each web page has two attributes, namely, how good is the page as an authority and how good is the page as a hub in terms of the quality of web pages to which it links. These two attributes mutually reinforce one another: the quality of a page as an authority is determined by the hub quality of pages that have a hyperlink to it and the quality of a page as a hub is determined by the authority quality of the pages to which it hyperlinks.

This reinforcement can be mathematically written in the following manner. Each page p has a hub value $h(p)$ and an authority value $a(p)$. Translating the above intuition into iterative equations, one can obtain $a_{t+1}(p) = \sum_{q|q\to p} M_{qp}h_t(q)$ and $h_{t+1}(p) = \sum_{q|p\to q} M_{pq}a_t(q)$. Since these iterations correspond to linear systems, they converge to their respective fixed points $\vec{a}$ and $\vec{h}$. It is easy to see that $\vec{a}$ is the principal eigenvector of M^TM and $\vec{h}$ is the principal eigenvector of MM^T. The quality of a web page p as an authoritative page is then given by its authority value $a(p)$.

There have been several modifications and extensions to Kleinberg's original work. For instance, [2,5,6,13]. Chakrabarti et al. [5] used the anchortext of a link to generalize the entries in M to [0,1]. Bharat and Henzinger [2] proposed many heuristics, including ones to address the issue of a page receiving unduly high authority value by virtue of many pages from the same web site pointing to it. Lempel and Moran [13] proposed a HITS-like method with slightly different definitions of authority and hub values.

For a comparative account of PageRank and HITS, the readers are referred to the survey by Borodin et al. [3].

Key Applications

As mentioned earlier, web page quality can be a vital input to the effectiveness of search engines. Besides this obvious application, ideas behind PageRank and HITS have been used to tackle many other problems on the web. Four such applications are mentioned below. Gyöngyi et al. [9] developed link analysis methods to identify spam web pages. Their method is based on identifying a small, reputable set of pages and using the hyperlinks to discover more pages that are likely to be good. Rafiei and Mendelzon [14] used random walks and PageRank to compute the topics for which a given web page has a good reputation. Bar-Yossef et al. [1] proposed a random walk with absorbing states to compute the decay value of a web page, where the decay value of a page measures how well-maintained and up-to-date is the page. Gibson et al. [8] used the HITS algorithm as a means to identifying web communities.

Cross-references

- ▶ Information Retrieval
- ▶ Web Data Management

Recommended Reading

1. Bar-Yossef Z., Broder A., Kumar R., and Tomkins A. Sic transit gloria telae: towards and understanding of the web's decay. In Proc. 12th Int. World Wide Web Conference, 2004, pp. 328–337.
2. Bharat K. and Henzinger M. Improved algorithms for topic distillation in a hyperlinked environment. In Proc. 21st Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 1998, pp. 104–111.
3. Borodin A., Roberts G.O., Rosenthal J.S., and Tsaparas P. Link analysis ranking algorithms, theory, and experiments. ACM Trans. Internet Tech., 5:231–297, 2005.
4. Brin S. and Page L. The anatomy of a large-scale hypertextual web search engine. Comput. Netw., 30:107–117, 1998.
5. Chakrabarti S., Dom B., Gibson D., Kleinberg J., Raghavan P., and Rajagopalan S. Automatic resource compilation by analyzing hyperlink structure and associated text. Comput. Netw., 30:65–74, 1998.
6. Chakrabarti S., Dom B., Gibson D., Kumar R., Raghavan P., Rajagopalan S., and Tomkins A. Spectral filtering for resource discovery. In Proc. ACM SIGIR Workshop on Hypertext Analysis. 1998, pp. 13–21.
7. Garfield E. Citation analysis as a tool in journal evaluation. Science, 178:471–479, 1972.
8. Gibson D., Kleinberg J., and Raghavan P. Inferring Web communities from link topology. In Proc. ACM Conference on Hypertext, 1998, pp. 225–234.
9. Gyöngyi Z., Garcia-Molina H., and Pedersen J. Combating web spam with TrustRank. In Proc. 30th Int. Conf. on Very Large Data Bases, 2004, pp. 576–587.
10. Haveliwala T.H. Topic-sensitive PageRank: A context-sensitive ranking algorithm for web search. IEEE Trans. Knowl. Data Eng., 15:784–796, 2003.
11. Kessler M.M. Bibliographic coupling between scientific papers. Am. Doc., 14:10–25, 1963.
12. Kleinberg J. Authoritative sources in a hyperlinked environment. J. ACM, 46:604–632, 2000.
13. Lempel R. and Moran S. SALSA: the stochastic approach for link-structure analysis. ACM Trans. Inform. Syst., 19:131–160, 2001.
14. Rafiei D. and Mendelzon A.O. What is this page known for? Computing web page reputations. Comput. Netw., 33:823–835, 2000.
15. Small H. Co-citaton in the scientific literature: a new measure of the relationship between two documents. J. Am. Soc. Inform. Sci., 24:265–269, 1973.

Web QA

▶ Web Question Answering

Web Query Languages

▶ Semantic Web Query Languages

Web Question Answering

Charles L. A. Clarke
University of Waterloo, Waterloo, ON, Canada

Synonyms

Web QA

Definition

A question answering (QA) system returns exact answers to questions posed by users in natural language, together

with evidence supporting those answers. A Web QA system maintains a corpus of Web pages and other Web resources in order to determine these answers and to provide the required evidence.

A basic QA system might support only simple factual (or "factoid") questions. For example, the user might pose the question

Q1. *What is the population of India?*

and receive the answer "1.2 billion," with evidence provided by the CIA World Factbook. A more advanced QA system might support more complex questions, seeking opinions or relationships between entities. Answering these complex questions might require the combination and integration of information from multiple sources. For example, the user might pose the question

Q2. *What methods are used to transport drugs from Mexico to the U.S.?*

and hope to receive a summary of information drawn from newspapers articles and similar documents. These two examples bracket the capabilities of current systems. Most QA systems can easily answer Q1, while questions such as Q2 represent an area of active research.

It is important to note that this definition of Web QA does not encompass Websites that allow users to answer each other's questions, nor sites that employ human experts to answer questions.

Historical Background

While experimental question answering systems have existed since the 1960's, research interest in question answering increased substantially in 1999, with the introduction of a question answering track at the Text REtreival Conference (TREC) [14,19]. TREC is an annual evaluation effort supervised by the US National Institute of Standards and Technology (NIST). Since its inception in 1991, TREC has conducted evaluation efforts for a wide range of information retrieval technologies, including question answering.

Inspired by the TREC QA track, Kwok et al. [8] developed one of the earliest Web QA systems. Their MULDER system answered questions by issuing queries to a commercial Web search engine and selecting answers from the pages it returned. They compared the performance of their system to that of several commercial search engines on the task of answering factoid questions taken from the TREC QA track.

Also inspired by TREC, Radev et al. [16] developed a method for converting questions into Web queries by applying techniques borrowed from statistical machine translation. Clarke et al. [5] crawled pages from the Web in order to gather information for answering trivia questions. Work by Dumais et al. [6] demonstrated the value of fully exploiting the volume of information available on the Web. Lam and Özsu [9] applied information extraction techniques to mine answers from multiple Web resources.

Much of the work on question answering was conducted outside the context of the Web. A general survey of this work is given by Prager [14]. An overview of many influential methods and systems is provided by Strzalkowski and Harabagiu [17].

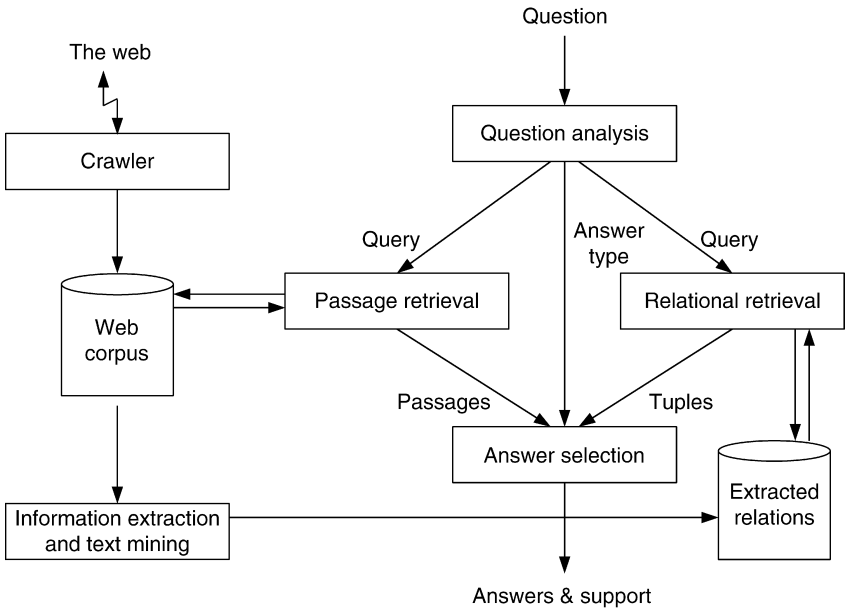
Foundations

Question answering is an extremely broad area, integrating aspects of natural language processing, machine learning, data mining and information retrieval. This entry focuses primarily on the use of Web resources for question answering.

Figure 1 provides a conceptual overview of the architecture of a basic QA system, showing the main components and processing steps. The system analyzes questions entered by the user, generates queries to collections of source material downloaded from the Web, and then selects appropriate answers from the resulting passages and tuples. The architecture shown in this figure emphasizes those components that are related to information retrieval and database systems, representing a composite derived from the published descriptions of a number of experimental systems [5,6,8,12,14,20]. The details of specific QA systems may differ substantially from this figure; an advanced QA system may include many additional components.

Sources for question answering are downloaded from the Web using a *crawler*, which is similar in construction and operation to the crawlers used by commercial Web search engines. After crawling, an inverted index is constructed, to allow the resulting collection to be searched efficiently.

Before construction of this inverted index, the crawler may annotate the collection to aid in question answering. For example, named entities, such as people, places, companies, quantities and dates, may be identified [15]. NLP techniques such as part-of-speech tagging and lightweight parsing may be applied. The creation of the index may also incorporate any of the standard indexing techniques associated with Web search, including link analysis.



Web Question Answering. Figure 1. Architecture of a Web question answering system.

Since the operation of a general Web crawler requires substantial effort and resources, many experimental QA systems substitute a commercial Web search engine for this crawling component [8]. To answer questions, one or more queries are issued to the search engine. Answers are then selected from the top documents returned by these queries.

As an additional post-processing step, distinct from crawling, the QA system may perform *information extraction and text mining* to identify events and relationships within the crawled pages. This extracted information, in the form of tuples, is used to populate tables of facts for reference by the QA system, essentially providing "prepared" answers for many factoid questions [2]. For example, associations between books and their authors, or between companies and their CEOs might be established in this way [1,4].

In addition, facts may be collected from pre-established sources that are known to be reliable. For example, current scores and schedules might be scraped from a sports Website; up-to-date movie and TV trivia might be taken from an entertainment site. Wrappers to collect this information might be constructed manually or learned from examples [7]. Since all answers should be appropriately supported by evidence, tuples containing extracted information should include links to the sources from which they are derived.

The result of these crawling and information extraction steps are two indexed collections. The first collection is a *Web corpus* of pre-processed Web pages, annotated to support question answering. The second collection is a database of *extracted relations*, providing pre-computed facts and answers. In its function and form, the first collection resembles a traditional information retrieval system; the second collection resembles a traditional relational database system.

These collections are prepared in advance of user queries. Answers are then determined by accessing and merging information from both collections. In a deployed QA system, these collections would be maintained on an on-going basis, much as an incremental crawler maintains the index of a commercial Web search engine.

After a user enters a question, the first stage in processing it is a *question analysis* step, in which an answer type is derived and queries to the retrieval components are constructed. The nature of the response required from the system will vary from question to question. The answer to a question might require a fact ("How many planets are there?"), a list ("What are the names of the planets?"), a yes-no response ("Is Pluto a planet?"), a definition ("What is a dwarf planet?"), or a description ("Why isn't Pluto a planet anymore?"). The role of the answer type is to convey the nature of

the expected response to the answer selection component, providing a mapping of the question onto the domain supported by the system.

The depth and detail provided by these answer types varies substantially from system to system. Broadly, the answer type encodes constraints implied by the question, which must be satisfied by the answer. For example, the answer type associated with Q1 might be any of

- QUANTITY
- POPULATION
- POPULATION (COUNTRY ("India"))

where the notation in this example is strictly for illustrative purposes. Depending on the system, an answer type might be encoded in a wide variety of ways. The answer type associated with Q2, might be any of

- SNIPPET
- SUMMARY
- SUMMARY ("transport", "drugs", COUNTRY ("United States"), COUNTRY("Mexico"))

These examples merely suggest the range of possibilities for expressing an answer type, which may include enumerations, patterns, and trees. Since a QA system should ideally be able to handle any question, many systems will support a catch-all OTHER answer type.

In addition to the answer type, the question analysis component generates queries to the two retrieval components: the *passage retrieval* component, which searches the Web corpus, and the *relational retrieval* component, which searches the extracted relations. The passage retrieval component searches the Web corpus and returns passages that may contain the answer. At its simplest, a query to the passage retrieval component is a list of keywords and phrases, resembling a typical query to a Web search engine. For example, for Q1 the generated query might be "`population india`," and for Q2 it might be "`transport drugs mexico`". At its simplest, the relational component may be queried using SQL, returning tuples that may contain the answer. For Q1, the population of India could be retrieved from a table of country populations and passed to the answer selection component.

For more complex questions, the queries issued to the retrieval components will be correspondingly more complex. In some cases, multiple queries may be issued to either or both components, particularly when the answer requires more than a simple fact. For example, as part of processing Q2, the relational component might retrieve lists of illicit drugs, the names of drug cartels and their leaders, and geographic locations along the US/Mexican border. It is also possible for the retrieval components to associate quality or certainty scores with passages and tuples they return, and these scores may then be taken into consideration when selecting an answer.

Constraints implied by the answer type might also be incorporated into the query. For example, the query "`population india <quantity>`" includes a annotation tag indicating that a retrieved passage should contain a positive numeric value. The annotations generated during the creation of the Web corpus may allow the passage retrieval component to support structured queries that precisely specify the required content of an answer-bearing passage [3].

For question answering, passage retrieval is normally preferred over the document-oriented retrieval typical of other IR systems. In many cases, answers to questions are contained within a few sentences or paragraphs, and the use of passages reduces the processing load placed on the answer selection component. Even when an answer requires information from multiple sources, the information required from each source can often be found within a short passage.

Typically, passages are one or more sentences in length, and the passage retrieval component might return tens or hundreds of passages for analysis by the answer selection component. For Q1, the passage retrieval component might return passages such as:

- *Uttar Pradesh is not only the most populous state in India but is also one of the largest in area . . .*
- *The population of India grew by near 20% during the 1990's to more than 1.1 billion . . .*
- *India is home to roughly 1.2 billion people . . .*

Keywords associated with the query are underlined. As seen in this example, the terms in passages need not match the keywords exactly; matches against morphological variants and other related terms ("populous" and "people") may also be permitted. Passage retrieval algorithms for question answering usually treat proximity of query keywords as an important feature for retrieval. Good passages may contain most or all of the query keywords in close proximity [18].

Guided by the answer type, the *answer selection* component analyzes the passages and tuples returned by the retrieval components to generate answers.

Depending on the answer type, the output from this component may be a ranked list, a single best answer, or even a null answer, indicating that the system was unable to answer the question. In determining possible answers, the selection component may start by identifying a set of *candidate answers*. For example, given an answer type of QUANTITY and the list of passages above, the set of answer candidates might include "20%," "1990," "more than 1.1 billion" and "roughly 1.2 billion." If the answer type is more restrictive, the candidates "20%" and "1990" might be excluded from the set.

If a candidate answer appears in multiple passages, this repetition or *redundancy* may strengthen the support for that candidate. For example, the candidate answers "more than 1.1 billion" and "roughly 1.2 billion" provide redundant support for an answer of "1.2 billion." Redundancy is an important feature in many answer selection algorithms [5,6]. The answer selection component must balance the support provided by redundancy with other factors, in order to select its final answer.

In the leading research systems, answer selection is a complex process, and the development effort underlying this component may substantially outweigh the development effort underlying the rest of the system combined. In these advanced systems, the answer selection component may issue additional queries to the retrieval components as it tests hypotheses and evaluates candidate answers [13]. A discussion of advanced techniques for answer selection is beyond the scope of this entry. Additional information, and references to the literature, may be found in Prager [14].

Evaluation efforts have been undertaken annually by NIST since 1999. Each year, dozens of research groups from industry and academia participate in these experimental efforts. While these efforts do not specifically focus on question answering in a Web context, the evaluation methodologies developed as part of these efforts may be applied to evaluate Web QA systems.

Instead of a Web corpus, a test collection of newspaper articles and similar documents is provided by NIST, and this collection forms the source for answers and evidence. However, many of the participating groups augment this collection with Web pages and other resources in order to improve the performance of their systems [20]. These systems use a combination of all available resources to determine an answer, and then project this answer back onto the test collection to provide the required evidence.

While the details vary from year to year, the basic experimental structure remains the same. The test collection is distributed to participating groups well before the start of a formal experiment. At the start of an experiment, NIST distributes a test set of questions to each group. Participants are free to annotate the test collection and modify their systems prior to receiving this test set, but they must freeze the development of their systems once they receive it. Each group executes the questions using their QA system and returns the answers to NIST, along with supporting evidence.

Answers are manually judged by NIST assessors, where the evidence must fully support an answer for it to judged correct. The answers to factoid questions are judged on a binary basis (correct or not). For more complex questions, answers are evaluated against a list of information "nuggets," where each nugget represents a single piece of knowledge associated with the answer [11]. For example, the use of low-water crossings on the Rio Grande might form a single nugget in the answer to Q2. Answers are then scored on the basis of the nuggets they contain. While all judging is performed manually for the official NIST results, efforts have been made to construct automatic judging tools and re-usable collections, which allow QA system evaluation to take place outside the framework of the official experiments [10,11].

Key Applications

By providing exact answers, rather than ranked lists of Web pages, question answering has the potential to reduce the effort associated with finding information on the Web. Commercial search engines already incorporate basic question answering features. For example, most commercial search engines provide an exact answer in response to Q1. However, none currently treat Q2 as anything other than a keyword query. Bridging the gap between these questions would represent a major step in the evolution of Web search.

Data Sets

From 1999 to 2007, the evaluation of experimental QA system was conducted as part of TREC (http://trec.nist.gov). Starting in 2008, the evaluation of QA systems was incorporated into a new experimental conference series, the Text Analysis Conference (http://www.nist.gov/tac/). Test collections and other experimental data be found by visiting those sites.

Cross-references

- ▶ Indexing the Web
- ▶ Information Retrieval
- ▶ Inverted Files
- ▶ Snippet
- ▶ Text Mining
- ▶ Web Crawler Architecture
- ▶ Web Harvesting
- ▶ Web Information Extraction
- ▶ Web Search Relevance Ranking

Recommended Reading

1. Agichtein E. and Gravano L. Snowball: Extracting relations from large plain-text collections. In Proc. ACM International Conference on Digital Libraries. 2000, pp. 85–94.
2. Agichtein E. and Gravano L. Querying text databases for efficient information extraction. In Proc. 19th Int. Conf. on Data Engineering, 2003, pp. 113–124.
3. Bilotti M.W., Ogilvie P., Callan J., and Nyberg E. Structured retrieval for question answering. In Proc. 33rd Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 2007, pp. 351–358.
4. Brin S. Extracting patterns and relations from the World Wide Web. In Proc. Int. Workshop on the World Wide Web and Databases, 1998, pp. 172–183.
5. Clarke C.L.A., Cormack G.V., and Lynam T.R. Exploiting redundancy in question answering. In Proc. 24th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 2001, pp. 358–365.
6. Dumais S., Banko M., Brill E., Lin J., and Ng A. Web question answering: Is more always better? In Proc. 25th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 2002, pp. 291–298.
7. Kushmerick N., Weld D.S., and Doorenbos R.B. Wrapper induction for information extraction. In Proc. 15th Int. Joint Conf. on AI, 1997, pp. 729–737.
8. Kwok C., Etzioni O., and Weld D.S. Scaling question answering to the Web. ACM Trans. Inf. Syst., 19(3):242–262, 2001.
9. Lam S.K.S. and Özsu M.T. Querying Web data – the WebQA approach. In Proc. 3rd Int. Conf. on Web Information Systems Eng., 2002, pp. 139–148.
10. Lin J. and Katz B. Building a reusable test collection for question answering. J. Am. Soc. Inf. Sci. Technol., 57(7): 851–861, 2006.
11. Marton G. and Radul A. Nuggeteer: Automatic nugget-based evaluation using descriptions and judgements. In Proc. Human Language Technology Conf. of the North American Chapter of the Association of Computational Linguistics, 2006, pp. 375–382.
12. Narayanan S. and Harabagiu S. Question answering based on semantic structures. In Proc. 20th Int. Conf. on Computational linguistics, 2004, pp. 693–701.
13. Pasca M.A. and Harabagiu S.M. High performance question/answering. In Proc. 24th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 2001, pp. 366–374.
14. Prager J. Open-domain question-answering. Found. Trends Inf. Retr., 1(2):91–231, 2006.
15. Prager J., Brown E., Coden A., and Radev D. Question-answering by predictive annotation. In Proc. 23rd Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 2000, pp. 184–191.
16. Radev D.R., Qi H., Zheng Z., Blair-Goldensohn S., Zhang Z., Fan W., and Prager J. Mining the Web for answers to natural language questions. In Proc. Int. Conf. on Information and Knowledge Management, 2001, pp. 143–150.
17. Strzalkowski T. and Harabagiu S. (eds.). Advances in Open Domain Question Answering. Springer, Secaucus, NJ, USA, 2006.
18. Tellex S., Katz B., Lin J., Fernandes A., and Marton G. Quantitative evaluation of passage retrieval algorithms for question answering. In Proc. 26th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 2003, pp. 41–47.
19. Voorhees E.M. Question answering in TREC. In TREC: Experiment and Evaluation in Information Retrieval, E.M. Voorhees and D.K. Harman (eds.). MIT, Cambridge, MA, USA, 2005, pp. 233–257.
20. Yang H., Chua T.-S., Wang S., and Koh C.-K. Structured use of external knowledge for event-based open domain question answering. In Proc. 26th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 2003, pp. 33–40.

Web Resource Discovery

- ▶ Focused Web Crawling

Web Scraper

- ▶ Web Data Extraction System

Web Scraping

- ▶ Languages for Web Data Extraction

Web Search and Crawling

- ▶ Biomedical Scientific Textual Data Types and Processing

Web Search Engines

► WEB Information Retrieval Models

Web Search Query Rewriting

Rosie Jones[1], Fuchun Peng[2]
[1]Yahoo! Research, Burbank, CA, USA
[2]Yahoo! Inc., Sunnyvale, CA, USA

Synonyms

Query reformulation; Query expansion; Query assistance; Query suggestion

Definition

Query rewriting in Web search refers to the process of reformulating an original input query to a new query in order to achieve better search results. Reformulation includes but not limited to the following:

1. Adding additional terms to express the search intent more accurately
2. Deleting redundant terms or re-weighting the terms in the original query to emphasize important terms
3. Finding alternative morphological forms of words by stemming each word, and searching for the alternative forms as well
4. Finding synonyms of words, and searching for the synonyms as well
5. Fixing spelling errors and automatically searching for the corrected form or suggesting it in the results

Historical Background

Web search queries are the words users type into web search engines to express their information need. These queries are typically 2–3 words long [7]. Traditional information retrieval allows retrieval of documents that do not contain all of the query words. While early search engines such as Lycos and AltaVista also retrieved documents which contained a subset of the query terms, current search engines such as Yahoo, Google, and MSN typically require web documents to contain all of the query terms (with the exception of *stop-words* – words such as prepositions and conjunctions which do not have a large impact on overall topic meaning).

This is generally an appropriate behavior for web search, since the large number of documents available ensure that some documents will be found, and the overall task is then reduced to ranking those documents. It also greatly reduces the number of documents which must be considered.

A typographical error, such as a misspelling, or use of non-conventional terminology (such as colloquial rather than medical terminology), or use of an ambiguous term can lead the search engine to retrieve fewer or different documents than the full set of documents relevant to the user's intent. The tendency of web searchers to look just at the first 10–20 results [8] exacerbates the severity of this, since it is then even more important to have the most relevant documents at the top of the list.

Most web search engines offer interactive or automated *query rewriting*, to correct spelling, suggest synonyms for terms, and suggest related topics and subtopics the user might be interested in. In interactive query rewriting, the web searcher is shown one or more options for a spell correction or rephrasing of a query [1]. In automated query rewriting, a query is automatically modified to correct spelling, add terms or otherwise modify the original query, then used to retrieve documents without further interaction from the user [12]. Automatic query rewriting needs to be accurate in order to avoid query intent drifting. Hence, most query rewriting techniques are for interactive query reformulation.

The pre-cursors to query rewriting in web search are spell correction (first used in word processing), and relevance and pseudo-relevance feedback [3] in information retrieval.

Foundations

Stemming

Stemming is the process of merging words with different morphological forms, such as singular and plural nouns, and verbs in past and present tense. Stemming is a long studied technology. Many stemmers have been developed, such as the Lovins stemmer [11] and the Porter stemmer [13]. The Porter stemmer is widely used due to its simplicity and effectiveness in many applications. However, the Porter stemming makes many mistakes because its simple rules cannot fully describe English morphology. Corpus analysis is used to improve Porter stemmer [17] by creating equivalence classes for words that are morphologically similar and occur in similar context.

Traditionally, stemming has been applied to Information Retrieval tasks by transforming words in documents to the their root form before indexing, and applying a similar transformation to query terms. Although it increases recall, this naive strategy does not work well for Web Search since it lowers precision. Recently, a context-sensitive based on statistical machine translation [12] is successfully applied to Web search to improve both recall and precision, by considering the likelihood of different variants occurring in web pages. The most likely variants are used to expand the query, and both the original query, and the modified version can be used to retrieve web pages.

Spelling Correction

In Web search, spelling correction takes a user input query and provides a "corrected" form. Spelling correction is typically modeled as a *noisy channel process*. Separate models are built for character level errors (edit models which represent probabilities of character level insertions, substitutions and deletions), and overall word or word sequence probability (typically referred to as a language model) [2]. Web query logs can be used to improve the quality of models, by providing data for the language model, as well as the edit model [4]. Spell correction is commonly used as a query rewriting step before retrieving documents in web search, though the user is also sometimes consulted, with an interface asking "Did you mean ...?"

Recently, major search engines are taking one step further on spelling correction by directly retrieving results with the corrected query instead of asking users to click on "Did you mean ...?" suggestions. This significantly improves users experience as users may not notice that their original input query are misspelled. For example, a query "brittney spears" input to major search engines now can directly retrieve results for the correct form "Britney Spears." However, it is also quite risky as it largely depends on the quality of misspelling correction. If a wrong suggestion is used, the retrieve search results would be totally different from the original user intent. Thus, this implicit query rewriting has to be very conservative.

Query Expansion

Query expansion adds words to a query to better reflect the search intent. For example, query "palo alto real estate" may be expanded to "palo alto California real estate" if it is known that the user is interested in city of "Palo Alto" in California. The expanded word "California" can be inferred from many features, such as IP address, session analysis, geographic mapping information.

Another type of expansion is to expand an abbreviated form to its full form. For example, "HISD" can be expanded to "Houston Independent School District." This is referred as abbreviation or acronym rewriting. One word may be mapped to multiple expansions, for example, "abc" could mean "American border collie," "american baseball coaches," "American born Chinese" or "american broadcasting company." Depending on the query context, it can be expanded to the correct form [16].

Query expansion can be done on concept level instead of individual word level [6,14]. For example, "job hunting" can be expanded to "recruiting" or "employment opportunity." One way of extracting concepts from query is through query segmentation [15].

Query Substitution and Suggestion

Some queries can be reformulated with a totally different query through query substitution and suggestion. Approaches to query substitution and suggestion involve identifying pairs of queries for which there is evidence that their meaning is related. One approach looks at sequences of queries issued by web searchers as the reformulate and modify their queries [10]. Others identify pairs of queries as related if they lead to clicks on the same documents or retrieve similar documents [5]. For example, query "the biggest sports event this year" and query "olympic 2008" have the same intent (assuming one infers that *this year* in the first query is 2008).

Query Word Deletion

It is obvious that stop words can be dropped in most cases in search. Some non-stop words can also be dropped [9]. Query word deletion is not so popular as query expansion, as in most queries can have enough matched documents in the whole Web. Still, word deletion from query is important as it can help improving recall of tail queries (queries that are not very frequent).

Key Applications

Query rewriting and suggestion is useful for improving performance of web search, as well as more generally identifying related terms which can be used for other tasks.

Future Directions

One area of research for query rewriting is how to detect query intent drifting after rewriting. For example, stemming query "marching network" to "march networks," or "blue steel" to "blued steel" is bad since the query after rewriting has different intent of the input. How to model intent drifting is a challenging task in Web search.

Cross-references

- ▶ Query Expansion
- ▶ Query Expansion models
- ▶ Query Rewriting
- ▶ Web Search Relevance Feedback

Recommended Reading

1. Anick P. Using terminological feedback for Web search refinement – a log-based study. In Proc. 26th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 2003, pp. 88–95.
2. Brill E. and Moore R.C. An improved error model for noisy channel spelling correction. In Proc. 38th Annual Meeting of the Assoc. for Computational Linguistics, 2000, pp. 86–293.
3. Croft W.B. and Harper D.J. Using probabilistic models of document retrieval without relevance information. J. Doc., 35 (4):285–295, 1979.
4. Cucerzan S. and Brill E. Spelling correction as an iterative process that exploits the collective knowledge of Web users. In Proc. Conf. on Empirical Methods in Natural Language Processing, 2004, pp. 293–300.
5. Cui H., Wen J.R., Nie J.Y., and Ma W.Y. Probabilistic query expansion using query logs. In Proc. 11th Int. World Wide Web Conference, 2002, pp. 325–332.
6. Fonseca B.M., Golgher P., Pssas B., Ribeiro-Neto B., and Ziviani N. Concept-based interactive query expansion. In Proc. 14th ACM Int. Conf. on Information and Knowledge Management, 2008, pp. 696–703.
7. Jansen B.J., Spink A., and Saracevic T. Real life, real users, and real needs: a study and analysis of user queries on the web. Inf. Process. Manage. Int. J., 36(2):207–227, 2000.
8. Joachims T., Granka L., Pang B., Hembrooke H., and Gay G. Accurately interpreting clickthrough data as implicit feedback. In Proc. 31st Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 2005, pp. 154–161.
9. Jones R. and Fain D. Query word deletion prediction. In Proc. 26th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 2003, pp. 435–436.
10. Jones R., Rey B., Madani O., and Greiner W. Generating query substitutions. In Proc. 15th Int. World Wide Web Conference, 2006, pp. 387–396.
11. Lovins J.B. Development of a stemming algorithm. Mech. Translat. Comput. Ling., 2:22–31, 1968.
12. Peng F., Ahmed N., Li X., and Lu Y. Context sensitive stemming for Web search. In Proc. 33rd Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 2007, pp. 639–646.
13. Porter M.F. An algorithm for suffix stripping. Program, 14(3):130–137, 1980.
14. Qiu Y. and Frei H.P. Concept based query expansion. In Proc. 16th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 1993, pp. 160–169.
15. Tan B. and Peng F. Unsupervised query segmentation using generative language models and wikipedia. In Proc. 17th Int. World Wide Web Conference, 2008, pp. 347–356.
16. Wei X., Peng F., and Dumoulin B. Analyzing Web text association to disambiguate abbreviation in queries. In Proc. 34th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 2008, pp. 751–752.
17. Xu J. and Croft B. Query expansion using local and global document analysis. In Proc. 19th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 1996, pp. 4–11.

Web Search Relevance Feedback

Hui Fang[1], ChengXiang Zhai[2]
[1]University of Delaware, Newark, DE, USA
[2]University of Illinois at Urbana-Champaign, Urbana, IL, USA

Definition

Relevance feedback refers to an interactive cycle that helps to improve the retrieval performance based on the relevance judgments provided by a user. Specifically, when a user issues a query to describe an information need, an information retrieval system would first return a set of initial results and then ask the user to judge whether some information items (typically documents or passages) are relevant or not. After that, the system would reformulate the query based on the collected feedback information, and return a set of retrieval results, which presumably would be better than the initial retrieval results. This procedure could be repeated.

Historical Background

Quality of retrieval results highly depends on how effective a user's query (usually a set of keywords) is in distinguishing relevant documents from non-relevant ones. Ideally, the keywords used in the query should occur only in the relevant documents and not in any non-relevant document. Unfortunately, in reality, it is often difficult for a user to come up with good keywords,

mainly because the same concept can be described using different words and the user often has no clue about which words are actually used in the relevant documents of a collection. A main motivation for relevance feedback comes from the observation that although it may be difficult for a user to formulate a good query, it is often much easier for the user to judge whether a document or a passage is relevant.

Relevance feedback was first studied in the vector space model by Rocchio [8]. After that, many other relevance feedback methods have been proposed and studied in other retrieval models [6,15], including classical probabilistic models and language modeling approach. Although these methods are different, they all try to make the best use of the relevance judgments provided by the user and generally rely on some kind of learning mechanism to learn a more accurate representation of the user's information need. Relevance feedback has proven to be one of the most effective methods for improving retrieval performance [8,6,10].

Unfortunately, due to the extra effort that a user must make in relevance feedback, users are often reluctant to provide such explicit relevance feedback. When there are no explicit relevance judgments available, *pseudo feedback* may be performed, also known as *blind relevance feedback* [2,7]. In this method, a small number of top-ranked documents in the initial retrieval results are assumed to be relevant, and relevance feedback is then applied. This method also tends to improve performance on average (especially recall). However, pseudo feedback method does not always work well, especially for queries where none of the top N documents is relevant. The poor performance is expected because the non-relevant documents are assumed to be relevant, which causes the reformulated query to shift away from the original information need.

Somewhere inbetween relevance feedback and pseudo feedback is a technique called *implicit feedback* [5,4,11]. In implicit feedback, a user's actions in interacting with a system (e.g., clickthroughs) are used to infer the user's information need. For example, a result viewed by a user may be regarded as relevant (or more likely relevant than a result skipped by a user). Search logs serve as the primary data for implicit feedback; their availability has recently stimulated a lot of research in learning from search logs to improve retrieval accuracy (e.g., [3]).

Relevance feedback bears some similarity to query expansion in the sense that both methods attempt to select useful terms to expand the original query for improving retrieval performance. However, these two methods are not exactly same. First, query expansion does not necessarily assume the availability of relevance judgments. It focuses on identifying useful terms that could be used to further elaborate the user's information need. The terms can come from many different sources, not just feedback documents (often called local methods for query expansion). For example, they can also come from any document in the entire collection (called global methods) [14] or external resources such as a thesaurus. Second, while relevance feedback is often realized through query expansion, it can be otherwise. For example, when all available examples are non-relevant, other techniques than query expansion may be more appropriate [13].

Foundations

The basic procedure for relevance feedback includes the following steps. A user first issues a query, and the system returns a set of initial retrieval results. After returning the initial results, the user is asked to judge whether the presented information items (e.g., documents or passages) are relevant or non-relevant. Finally, the system revises the original query based on the collected relevance judgments typically by adding additional terms extracted from relevant documents, promoting weights of terms that occur often in relevant documents, but not so often in non-relevant documents, and down-weighting frequent terms in non-relevant documents. The revised query is then executed and a new set of results is returned. The procedure can be repeated more than once.

Technically, relevance feedback is a learning problem in which one learns from the relevant and non-relevant document examples how to distinguish new relevant documents from non-relevant documents. The learning problem can be cast as to learn either a binary classifier that can classify a document into relevant vs. non-relevant categories or a ranker that can rank relevant documents above non-relevant documents. Thus in principle, any standard supervised learning methods can be potentially applied to perform relevance feedback. However, as a learning problem, relevance feedback poses several special challenges, and as a result, a direct application of a standard supervised learning method is often not very

effective. First, many supervised learning methods only work well when there are a relatively large number of training examples, but in the case of relevance feedback, the number of positive training examples is generally very small. Second, the training examples in relevance feedback are usually from the top-ranked documents, thus they are not a random sample and can be biased. Third, it is unclear how to handle the original query. The query may be treated as a special short example of relevant documents, but intuitively this special relevant example is much more important than other relevant examples and the terms in the query should be sufficiently emphasized to avoid drifting away from the original query concept. These challenges are exacerbated in the case of pseudo feedback as the query would be the only reliable relevance information provided by the user.

In the information retrieval community, many relevance feedback algorithms have been developed for different retrieval models. A commonly used feedback algorithm in vector space models is the *Rocchio algorithm* developed in early 1970s [8], which remains a robust effective state-of-the-art method for relevance feedback. In Rocchio, the problem of relevance feedback is defined as finding an optimal query to maximize its similarity to relevant documents and minimize its similarity to non-relevant documents. The revised query can be computed as

$$Q_r = \alpha \cdot Q_o + \frac{\beta}{|D_r|} \cdot \sum_{d_i \in D_r} d_i - \frac{\gamma}{|D_n|} \cdot \sum_{d_j \in D_n} d_j,$$

where Q_r is the revised query vector, Q_o is the original query vector, D_r and D_n are the sets of vectors for the known relevant and non-relevant documents, respectively. α, β, and γ are the parameters that control the contributions from the two sets of documents and the original query. Empirical results show that positive feedback is much more effective than negative feedback, so the optimal value of γ is often much smaller than that of β. A term would have a higher weight in the new query vector if it occurs frequently in relevant documents but infrequently in non-relevant documents. The IDF weighting further rewards a term that is rare in the whole collection.

The feedback method in classical probabilistic models is to select expanded terms primarily based on *Robertson/Sparck-Jones weight* [6] defined as follows:

$$w_i = log \frac{r_i/(R - r_i)}{(n_i - r_i)/(N - n_i - R + r_i)},$$

where w_i is the weight of term i, r_i is the number of relevant documents containing term i, n_i is the number of documents containing term i, R is the number of relevant documents in the collection, and N is the number of documents in the collection.

In language modeling approaches, feedback can be achieved through updating the query language model based on feedback information, leading to *model-based feedback methods*[15]. Relevance feedback received little attention in logical model, but some possible feedback methods have been discussed in [9].

Despite the difference in their way of performing relevance feedback, all these feedback methods implement the same intuition, which is to improve query representation by introducing and rewarding terms that occur frequently in relevant documents but infrequently in non-relevant documents. When optimized, most of these methods tend to perform similarly. Improvements over these basic feedback methods include (1) *passage feedback* [1] which can filter out the non-relevant part of a long relevant document in feedback, (2) *query zone* [12] which can improve the use of negative feedback information, and (3) *pure negative feedback* [13] which aims at exploiting purely negative information to improve performance for difficult topics.

Most studies focus on how to use the relevance judgments to improve the retrieval performance. The issue of choosing optimally documents to judge for relevance feedback has received considerably less attention. Recent studies on active feedback [11] have shown that choosing documents with more diversity for feedback is a better strategy than choosing the most relevant documents for feedback in the sense that the relevance feedback information collected with the former strategy is more useful for learning. However, this benefit may be at the price of sacrificing the utility of presented documents from a user's perspective.

Although relevance feedback improves retrieval accuracy, the improvement comes at the price of possibly slowing down the retrieval speed. Indeed, virtually all relevance feedback algorithms involve iterating over all the terms in all the judged examples, so the computational overhead is not negligible, making it a challenge to use relevance feedback in applications where response speed is critical.

Key Applications

Relevance feedback is a general effective technique for improving retrieval accuracy for all search engines, which is applicable when users are willing to provide explicit relevance judgments. Relevance feedback is particularly useful when it is difficult to completely and precisely specify an information need with just keywords (e.g., *multimedia search*); in such a case, relevance feedback can be exploited to learn features that can characterize the information need from the judged examples (e.g., useful non-textual features in multimedia search) and combine such features with the original query to improve retrieval results. Relevance feedback is also a key technique in *personalized search* where both explicit and implicit feedback information could be learned from all the collected information about a user to improve the current search results for the user.

Despite the fact that relevance feedback is a mature technology, it is not a popular feature provided by the current web search engines possibly because of the computational overhead. However, the feature "finding similar pages" provided by Google can be regarded as relevance feedback with just one relevant document.

Future Directions

Although relevance feedback has been studied for decades and many effective methods have been proposed, it is still unclear what is the optimal way of performing relevance feedback. The recent trend of applying machine learning to information retrieval, especially research on learning to rank and statistical language models, will likely shed light on the answer to this long-standing question.

So far, research on relevance feedback has focused on cases where at least several relevant examples can be obtained in the top-ranked documents. However, when a topic is difficult, a user may not see any relevant document in the top-ranked documents. In order to help such a user, negative feedback (i.e., relevance feedback with only non-relevant examples) must be performed. Traditionally negative feedback has not received much attention due to the fact that when relevant examples are available, negative feedback information tends not to be very useful. An important future research direction is to study how to exploit negative feedback to improve retrieval accuracy for difficult topics as argued in [13].

Active feedback [11] is another important topic worth further studying. Indeed, to minimize a user's effort in providing explicit feedback, it is important to select the most informative examples for a user to judge so that the system can learn most from the judged examples. It would be interesting to explore how to apply/adapt many active learning techniques developed in the machine learning community to perform active feedback.

Experimental Results

The effectiveness of a relevance feedback method is usually evaluated using the standard information retrieval evaluation methodology and test collections. A standard test collection includes a document collection, a set of queries and judgments indicating whether a document is relevant to a query. The initial retrieval results are compared with the feedback results to show whether feedback improves performance. The performance is often measured using Mean Average Precision (MAP) which reflects the overall ranking accuracy or precision at top k (e.g., top 10) documents which reflects how many relevant documents a user can expect to see in the top-ranked documents.

When comparing different relevance feedback methods, the amount of feedback information is often controlled so that every method uses the same judged documents. The judged or seen documents are usually excluded when computing the performance of a method to more accurately evaluate the performance of a method on *unseen* documents. The evaluation is significantly more challenging when the amount of feedback information can not be controlled, such as when different strategies for selecting documents for feedback are compared or a feedback method and a non-feedback method are compared. In such cases, it is tricky how to handle the judged documents. If they are not excluded when computing the performance of a method, the comparison would be unfair and favor a feedback that has received more judged examples. However, if they are excluded, the performance would not be comparable either because the test set used to compute the performance of each method would likely be different and may contain a different set of relevant and non-relevant documents.

Data Sets

Many standard information retrieval test collections can be found at: http://trec.nist.gov/

URL to Code

Lemur project contains the code for most existing relevance feedback method, which can be found at: http://lemurpoject.org/.

Cross-references

- ▶ Implicit Feedback
- ▶ Pseudo Feedback
- ▶ Query Expansion

Recommended Reading

1. Allan J. Relevance feedback with too much data. In Proc. 18th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 1995, pp. 337–343.
2. Buckley C. Automatic query expansion using SMART: TREC-3. In Overview of the Third Text Retrieval Conference (TREC-3), D. Harman (ed.), 1995, pp. 69–80.
3. Burges C., Shaked T., Renshaw E., Lazier A., Deeds M., Hamilton N., and Hullender G. Learning to rank using gradient descent. In Proc. 22nd Int. Conf. on Machine Learning, 2005, pp. 89–96.
4. Joachims T. Optimizing search engines using clickthrough data. In Proc. 8th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, 2002, pp. 133–142.
5. Kelly D. and Teevan J. Implicit feedback for inferring user preference. SIGIR Forum, 37(2):18–28, 2003.
6. Robertson S.E. and Jones K.S. Relevance weighting of search terms. J. Am. Soc. Inf. Sci., 27(3):129–146, 1976.
7. Robertson S.E., Walker S., Jones S., Hancock-Beaulieu M.M., and Gatford M. Okapi at TREC-3. In Proc. The 3rd Text Retrieval Conference, 1995, pp. 109–126.
8. Rocchio J. Relevance feedback in information retrieval. In The SMART Retrieval System: Experiments in Automatic Document Processing. Prentice-Hall, Englewood Cliffs, NJ, 1971, pp. 313–323.
9. Ruthven I. and Lalmas M. A survey on the use of relevance feedback for information access system. Knowl. Eng. Rev., 18(2):95–145, 2003.
10. Salton G. and Buckley C. Improving retrieval performance by relevance feedback. J. Am. Soc. Inf. Sci., 44(4):288–297, 1990.
11. Shen X. and Zhai C. Active feedback in ad hoc information retrieval. In Proc. 31st Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 2005, pp. 59–66.
12. Singhal A., Mitra M., and Buckley C. Learning routing queries in a query zone. In Proc. 20th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 1997, pp. 25–32.
13. Wang X., Fang H., and Zhai C. A study of methods for negative relevance feedback. In Proc. 34th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 2008, pp. 219–226.
14. Xu J. and Croft W.B. Query expansion using local and global document analysis. In Proc. 19th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 1996, pp. 4–11.
15. Zhai C. and Lafferty J. Model-based feedback in the language modeling approach to information retrieval. In Proc. Int. Conf. on Information and Knowledge Management, 2001, pp. 403–410.

Web Search Relevance Ranking

Hugo Zaragoza[1], Marc Najork[2]
[1]Yahoo! Research, Barcelona, Spain
[2]Microsoft Research, Mountain View, CA, USA

Synonyms

Ranking; Search ranking; Result ranking

Definition

Web search engines return lists of web pages sorted by the page's relevance to the user query. The problem with web search relevance ranking is to estimate relevance of a page to a query. Nowadays, commercial web-page search engines combine hundreds of features to estimate relevance. The specific features and their mode of combination are kept secret to fight spammers and competitors. Nevertheless, the main types of features at use, as well as the methods for their combination, are publicly known and are the subject of scientific investigation.

Historical Background

Information Retrieval (IR) Systems are the predecessors of Web and search engines. These systems were designed to retrieve documents in curated digital collections such as library abstracts, corporate documents, news, etc. Traditionally, IR relevance ranking algorithms were designed to obtain high recall on medium-sized document collections using long detailed queries. Furthermore, textual documents in these collections had little or no structure or hyperlinks. Web search engines incorporated many of the principles and algorithms of Information Retrieval Systems, but had to adapt and extend them to fit their needs.

Early Web Search engines such as Lycos and AltaVista concentrated on the scalability issues of running web search engines using traditional relevance ranking algorithms. Newer search engines, such as Google, exploited web-specific relevance features such as hyperlinks to obtain significant gains in quality. These measures were partly motivated by research in citation analysis carried out in the bibliometrics field.

Foundations

For most queries, there exist thousands of documents containing some or all of the terms in the query. A search engine needs to rank them in some appropriate way so that the first few results shown to the user will be the ones that are most pertinent to the user's need. The interest of a document with respect to the user query is referred to as "document relevance." this quantity is usually unknown and must be estimated from features of the document, the query, the user history or the web in general. Relevance ranking loosely refers to the different features and algorithms used to estimate the relevance of documents and to sort them appropriately.

The most basic retrieval function would be a Boolean query on the presence or absence of terms in documents. Given a query "word1 word2" the Boolean AND query would return all documents containing the terms word1 and word2 at least once. These documents are referred to as the query's "AND result set" and represent the set of potentially relevant documents; all documents not in this set could be considered irrelevant and ignored. This is usually the first step in web search relevance ranking. It greatly reduces the number of documents to be considered for ranking, but it does not rank the documents in the result set. For this, each document needs to be "scored", that is, the document's relevance needs to be estimated as a function of its relevance features. Contemporary search engines use hundreds of features. These features and their combination are kept secret to fight spam and competitors. Nevertheless, the general classes of employed features are publicly known and are the subject of scientific investigation. The main types of relevance features are described in the remainder of this section, roughly in order of importance. Note that some features are query-dependent and some are not. This is an important distinction because query-independent features are constant with respect to the user query and can be pre-computed off-line. Query-dependent features, on the other hand, need to be computed at search time or cached.

Textual Relevance

Modern web search engines include tens or hundreds of features which measure the textual relevance of a page. The most important of these features are matching functions which determine the term similarity to the query. Some of these matching functions depend only on the frequency of occurrence of query terms; others depend on the page structure, term positions, graphical layout, etc. In order to compare the query and the document, it is necessary to carry out some non-trivial preprocessing steps: tokenization (splitting the string into word units), letter case and spelling normalization, etc. Beyond these standard preprocessing steps, modern web search engines carry out more complex query reformulations which allow them to resolve acronyms, detect phrases, etc.

One of the earliest textual relevance features (earliest both in information retrieval systems and later in commercial web search engines) is the vector space model scoring function. This feature was used by early search engines. Since then other scoring models have been developed in Information Retrieval (e.g., Language Models and Probabilistic Relevance Models) [8] and have probably been adopted by web search engines. Although web search engines do not disclose details about their textual relevance features, it is known that they use a wide variety of them, ranging from simple word counts to complex nonlinear functions of the match frequencies in the document and in the collection.

Furthermore, web search engines make use of the relative and absolute position of the matches in the document. In Information Retrieval publications, there have been many different proposals to make use of term position information, but no consensus has been reached yet on the best way to use it. Most known approaches are based on features of the relative distances of the match terms such as the minimum (or average, or maximum) size of the text span containing all (or some, or most) of the term matches. Web search engines have not disclosed how they use position information.

Besides match position, web search engines exploit the structure or layout of documents, especially HTML documents. There are a number of ways to do this. One of the simplest is to compute textual similarity with respect to each document element (title, subtitles, paragraphs). More complex solutions integrate matches of different structural elements into a single textual relevance score (see for example [8]).

Another type of textual relevance information is provided by the overall document quality. For example, Web search engines use automatic document classifiers to detect specific document genres such as adult content, commercial sites, etc. Specialized techniques

are also used to detect spam pages. Pages may be eliminated or demoted depending on the result of these analyses.

Hyperlink Relevance

The web is a hyperlinked collection of documents (unlike most previously existing digital collections, which had only implicit references). A hyperlink links a span of text in the source page (the "anchor text") to a target page (the "linked page"). Because of this, one can think of a hyperlink as a reference, an endorsement, or a vote by the source page on the target page. Similarly, one can think of the anchor text as a description or an explanation of the endorsement. One of the innovations introduced by Web Search Engines was leveraging the hyperlink-structure of the web for relevance ranking purposes.

The hyperlink graph structure can be used to determine the importance of a page independently on the textual content of the pages. This idea of using web hyperlinks as endorsements was originally proposed by Marchiori [9] and further explored by Kleinberg [6] and Page et al. [11] (who also introduced the idea of using the anchor text describing the hyperlink to augment the target page).

Exploiting User Behavior

As stated above, hyperlink analysis leverages human intelligence, namely peer endorsement between web page authors. Web search engines can also measure users' endorsements by observing the search result links that are being clicked on. This concept was first proposed by Boyan et al. [5], and subsequently commercialized by DirectHit [3]. For an up-to-date summary of the state of the art, the reader is referred to [5]. Besides search result clicks, commercial search engines can obtain statistics of page visitations (i.e., *popularity*) from browser toolbars, advertising networks or directly from Internet service providers. This form of quality feedback is query-independent and thus less informative but more abundant.

Performance

There are several aspects of the performance of a web relevance ranking algorithm. There are the standard algorithmic performance measures such as speed, disk and memory requirements, etc. Running time efficiency is crucial for web search ranking algorithms, since billions of documents need to be ranked in response to millions of queries per hour. For this reason most features need to be pre-computed off-line and only their combination is computed at query time. Some features may require specialized data structures to be retrieved especially fast at query time. This is the case for example of term-weights (which are organized in inverted indices [1]), or query-dependent hyperlink features [10].

A more fundamental aspect of the performance of a relevance ranking algorithm is its accuracy or precision: how good is the algorithm at estimating the relevance of pages? This is problematic because relevance is a subjective property, and can only be observed experimentally, asking a human subject. Furthermore, the performance of a ranking algorithm will not depend equally on each page: the best ranked pages are those seen by most users and therefore the most important to determine the quality of the algorithm in practice. Performance evaluation measures used for the development of relevance ranking algorithms take this into account [8].

There exist other, less explicit measures of performance. For example, as users interact with a search engine, the distribution of their clicks on the different ranks give an indication of the quality of the ranking (i.e., rankings leading to many clicks on the first results may be desired). However, this information is private to the search engines, and furthermore it is strongly biased by the order of presentation of results.

Feature Combination

All of the features of a page need to be combined to produce a single relevance score. Early web search engines had only a handful of features, and they were combined linearly, manually tuning their relative weights to maximize the performance obtained on a test set of queries. Modern search engines employ hundreds of features and use statistical methods to tune these features. Although the specific details remain secret, a number of research publications exist on the topic (see for example [13]).

Key Applications

The key application of web search relevance ranking is in the algorithmic search component of web search engines. Similar methods are also employed to bias the ranking of the advertisements displayed in search results. Some of the principles have been applied in other types of search engines such as corporate search (intranet, email, document archives, etc.).

Future Directions

Research continues to improve all of the relevance features discussed here. This research has lead to a continuous improvement of search engine quality. Nevertheless, current relevance features are becoming increasingly hard to improve upon. Considerable research is centered today on discovering new types of features which can significantly improve search quality. Only two of the most promising areas are mentioned here:

Query-understanding: Different types of queries may require very different types of relevance ranking algorithms. For example, a shopping query may require very different types of analysis from a travel or a health query. Work on algorithms that understand the intent of a query and select different relevance ranking methods accordingly could lead to dramatic increases in the quality of the ranking.

Personalization: In principle it is possible to exploit user information to "personalize" web search engine results. Different results would be relevant to a query issued by a layperson than a topic expert, for example. There are many different ways to personalize results: with respect to the user search history, with respect to the user community, with respect to questionnaires or external sources of knowledge about the user, etc. Many scientific papers have been written on this topic, but the problem remains unsolved. Commercial web search engines have mainly shied away from personalized algorithms. Google has proposed several forms of personalized search to its users, but this feature has not had much success. Nevertheless, the search continues for the right way to personalize relevance ranking.

Experimental Results

Evaluation of web relevance ranking is difficult and very costly, since it involves human judges labeling a collection of queries and results as to their relevance. The most careful evaluations of web relevance features are carried out by web search engine companies, but they are not disclosed to the public. There have been very many partial evaluations of search engines published, but they are always controversial due to their small scale, their experimental biases and their indirect access to the search engine features.

A number of experimental benchmarks have been constructed for public scientific competitions. Although they are small and partial they can be used for experimentation (see below).

Data Sets

Commercial search engines dispose of very large data sets comprising very many documents (e.g., hundreds of millions), queries (e.g., tens of thousands), and human relevance evaluations (e.g., hundreds of thousands). These data sets are routinely used to develop and improve features for relevance ranking. See [10,13] for examples of this.

Publicly available data sets for experimentation are very small compared to those used by commercial search engines. Nevertheless, they may be used to investigate some features and their combination. The most important datasets for web relevance ranking experiments are those developed in the Web-track of the TREC competition organized by NIST [4].

URL to Code

Due to the extraordinary cost of developing and maintaining a full-scale web search engine, there are no publically available systems so far. The Nutch project (http://lucene.apache.org/nutch/) is aiming to build an open-source web-scale search engine based on the Lucene search engine. Other retrieval engines capable of crawling and indexing up to several millions of documents include INDRI (http://www.lemurproject.org/indri/), MG4J (http://mg4j.dsi.unimi.it/) and TERRIER (http://ir.dcs.gla.ac.uk/terrier/).

Cross-references

- ▶ Anchor Text
- ▶ BM25
- ▶ Document Links and Hyperlinks
- ▶ Field-Based Information Retrieval Models
- ▶ Information Retrieval
- ▶ Language Models
- ▶ Relevance
- ▶ Relevance Feedback
- ▶ Text Categorization
- ▶ Text Indexing and Retrieval
- ▶ Vector-Space Model
- ▶ WEB Information Retrieval Models
- ▶ Web Page Quality Metrics
- ▶ Web Search Relevance Feedback
- ▶ Web Spam Detection

Recommended Reading

1. Baeza-Yates R. and Ribeiro-Neto B. Modern Information Retrieval. Addison Wesley, Reading, MA, 1999.
2. Boyan J., Freitag D., and Joachims T. A machine learning architecture for optimizing web search engines. In Proc. AAAI Workshop on Internet Based Information Systems, 1996.

3. Culliss G. The Direct Hit Popularity Engine Technology. A White Paper, DirectHit, 2000. Available online at https://www.uni-koblenz.de/FB4/Institutes/ICV/AGKrause/Teachings/SS07/DirectHit.pdf. Accessed on 27 Nov 2007.
4. Hawking D. and Craswell N. Very large scale retrieval and Web search. In TREC: Experiment and Evaluation in Information Retrieval, E. Voorhees and D. Harman (eds.). MIT Press, Cambridge, MA, 2005.
5. Joachims T. and Radlinski F. Search engines that learn from implicit feedback. IEEE Comp., 40(8):34–40, 2007.
6. Kleinberg J. Authoritative sources in a hyperlinked environment. Technical Report RJ 10076, IBM, 1997.
7. Langville A.N. and Meyer C.D. Google's PageRank and Beyond: The Science of Search Engine Rankings. Princeton University Press, Princeton, NJ, 2006.
8. Manning C.D., Raghavan P., and Schütze H. Introduction to Information Retrieval. Cambridge University Press, Cambridge, UK, 2008.
9. Marchiori M. The quest for correct information on the Web: hyper search engines. In Proc. 6th Int. World Wide Web Conference, 1997.
10. Najork M. Comparing the effectiveness of HITS and SALSA. In Proc. Conf. on Information and Knowledge Management, 2007, pp. 157–164.
11. Page L., Brin S., Motwani R., and Winograd T. The PageRank citation ranking: bringing order to the Web. Technical Report, Stanford Digital Library Technologies Project.
12. Richardson M., Prakash A., and Brill E. Beyond PageRank: machine learning for static ranking. In Proc. 15th Int. World Wide Web Conference, 2006, pp. 707–715.
13. Taylor M., Zaragoza H., Craswell N., Robertson S., and Burges C. Optimisation methods for ranking functions with multiple parameters. In Proc. Conf. on Information and Knowledge Management, 2006, pp. 585–593.

Web Search Result Caching and Prefetching

Ronny Lempel[1], Fabrizio Silvestri[2]
[1]Yahoo! Research, Haifa, Israel
[2]ISTI-CNR, Pisa, Italy

Synonyms

Search engine caching and prefetching; Search engine query result caching; Paging in Web search engines

Definition

Caching is a well-known concept in systems with multiple tiers of storage. For simplicity, consider a system storing N objects in relatively slow memory, that also has a smaller but faster memory buffer of capacity k which can store copies of k of the N objects ($N >> k$). This fast memory buffer is called the *cache*. The storage system is presented with a continuous stream of queries, each requesting one of the N objects. If the object is stored in the cache, a *cache hit* occurs and the object is quickly retrieved. Otherwise, a *cache miss* occurs, and the object is retrieved from the slower memory. At this point, the storage system can opt to save the newly retrieved object in the cache. When the cache is full (i.e., already contains k objects), this entails *evicting* some currently cached object. Such decisions are handled by a *replacement policy*, whose goal is to maximize the *cache hit ratio* (or *rate*) – the proportion of queries resulting in cache hits.

Often, access patterns to objects, as found in query streams, are temporally correlated. For example, object y might often be requested shortly after object x has been requested. This motivates *prefetching* – the storage system can opt to retrieve and cache y upon encountering a query for x, anticipating the probable future query for y.

The setting with respect to search result caches in Web search engines is somewhat different. There, the objects to be cached are result pages of *search queries*. A search query is defined as a triplet $q = (qs, from, n)$ where qs is a query string, $from$ denotes the relevance rank of the first result requested, and n denotes the number of requested results. The result page corresponding to q would contain the results whose relevance rank with respect to qs are $from$, $from + 1$,..., $from + n - 1$. The value of n is typically 10. Search engines set aside some storage to cache such result pages. However, a search engine is not a typical two-tiered storage structure, as results not found in the cache are not stored in slower storage but rather need to be generated through the query evaluation process of the search engine. Prefetching of search results occurs when the engine computes and caches $j \cdot n$ results for the query $(qs, from, n)$, with j being some small integer, in anticipation of follow-up queries requesting additional result pages for the same query string.

Historical Background

Markatos was the first to study search result caching in depth in 2000 [11]. He experimented with various known cache replacement policies on a log of queries submitted to the Excite search engine, and compared the resulting hit-ratios. In 2001 Saraiva et al. [12]

proposed a two-level caching scheme that combines caching of search results with the caching of frequently accessed postings lists.

Prefetching of search engine results was studied from a theoretical point of view by Lempel and Moran in 2002 [6]. They observed that the work involved in query evaluation scales in a sub-linear manner with the number of results computed by the search engine. Then they proceeded to minimize the computations involved in query evaluations by optimizing the number of results computed per query. The optimization is based on a workload function that models both (i) the computations performed by the search engine to produce search results and (ii) the probabilistic manner by which users advance through result pages in a *search session*.

In 2003, two papers proposed caching algorithms specifically tailored to the locality of reference present in search engine query streams: *PDC – Probability Driven Caching* [7], and *SDC – Static Dynamic Caching* [14] (extended version in [4]). The next section will focus primarily on those caching strategies and follow-up papers, e.g., the AC strategy proposed by Baeza-Yates et al. in [2].

In 2004, Lempel and Moran studied the problem of caching search engine results in the theoretical framework of competitive analysis [8]. For a certain stochastic model of search engine query streams, they showed an online caching algorithm whose expected number of cache misses is no worse than four times that of any online algorithm.

Note that search engine result caching is just one specific application of caching or paging; the theoretical analysis of page replacement policies dates back to 1966, when Belady [3] derived the optimal offline page replacement algorithm. In 1985, Sleator and Tarjan published their seminal paper on *online paging* [15], showing the optimal competitiveness of the Least Recently Used (LRU) policy.

Foundations

The setting for the caching of search results in search engines is as follows. The engine, in addition to its *index*, dedicates some fixed-size fast memory cache that can store up to k result pages. It is then presented with a stream of user-submitted search queries. For each query it consults the cache, and upon a cache hit returns the cached page of results to the user. Upon a cache miss, the query undergoes evaluation by the index, and the requested page of results (along with perhaps additional result pages) are computed. The requested results are returned to the user, and all newly computed result pages are forwarded to the cache replacement algorithm. In case the cache is not full, these pages will be cached. If the cache is full, the replacement algorithm may decide to evict some currently cached result pages to make room for the newly computed pages.

The primary goal of caching schemes (replacement policies and prefetching strategies) is to exploit the *locality of reference* that is present in many real life query streams in order to exhibit high hit ratios. Roughly speaking, locality of reference means that after an object x is requested, x and a small set of related objects are likely to be requested in the near future. Two types of locality of reference present in search engine query streams are detailed below.

Topical Locality of Reference

The variety of queries encountered by modern Web search engines is enormous. People submit queries on any and all walks of life, in all languages spoken around the world. Nevertheless, some query strings are much more popular than others and occur with high frequency, motivating the caching of their results by the search engine. Frequent queries can be roughly divided into two classes: (i) queries whose popularity is stable over time, and (ii) queries with bursty popularity, which may rise suddenly (e.g., due to some current event) and drop quickly soon thereafter.

Sequential Locality of Reference

This is due to the discrete manner in which search engines present search results to users – in batches of n (typically 10) results at a time. Some users viewing the page of results for the query $q = (qs, from, n)$ will not be fully satisfied, and will soon thereafter submit the follow-up query $(qs, from + n, n)$. In general, when many users have recently queried for $(qs, from, n)$, the higher the likelihood that at least one of them querying for $(qs, from + n, n)$. This behavior of users, coupled with the observation that the work involved in computing a batch of size $j \cdot n$, $j > 1$ results for a query string is significantly smaller than j independent computations of batches of n results, motivates the prefetching of search results in search engines [9].

Several studies have analyzed the manner in which users query search engines and view result

pages [11,7, 13,1]. In general, it has been observed on numerous query logs that query popularities obey a power-law distribution, i.e., the number of query strings appearing n times in the log is proportional to n^{-c}. For example, Lempel and Moran [7] report a value of c equaling about 2.4 on a log of seven million queries submitted to AltaVista in late 2001 (The number of extremely popular queries typically exceeds the value predicted by the power law distribution.). While the 25 or so most popular queries in the query stream may account for over 1% of an engine's query load, a large fraction of any log is composed of queries that appear in it only once. For example, Baeza-Yates et al. [1] analyze a log in which the vast majority (almost 88%) of unique query strings were submitted just once, accounting for about 44% of the total query load. Such queries are bound to cause cache misses, and along with the misses incurred for the first occurrence of any repeating query imply an upper bound of about 50% for cache hit rates on that log.

Many of the studies cited above report that users typically browse through very few result pages of a query, and that the "deeper" the result page, the less users view it. While the exact numbers vary in each study, the reports agree that the majority of queries will have only their top-ten results viewed, and that the percentage of queries for which more than three result pages are viewed are very small. However, note that statistical data in the order of 10^{-2} (single percentage points) are powerful predictors for the purpose of caching result pages.

Beyond achieving high hit ratios, caching schemes must also be efficient, i.e., the logic involved in cache replacement decisions must be negligible as compared with the cost of a cache miss. Furthermore, they should be designed for high throughput, as search engines serve many queries concurrently and it would be counterproductive to have cache access become a bottleneck in their computation pipeline.

Different caching schemes have been applied to search results, ranging from "classic" ones to policies specifically designed for search engines workloads.

"Classic" Caching Schemes

These schemes include the well-known *Least Recently Used* (LRU) replacement policy and its variations (e.g., SLRU and LRU2) [15]. To quickly recap, upon a cache miss, LRU replaces the least recently accessed of the currently cached objects with the object that caused the cache miss. SLRU is a two-tiered LRU scheme: cache misses result in new objects entering the lower-tier LRU. Cache hits in the lower-tier LRU are promoted to the upper tier, whose least recently accessed object is relegated to be the most recently accessed object of the lower tier. The above replacement policies are very efficient, requiring O(1) time per replacement of an item. Furthermore, they are orthogonal to prefetching policies in the sense that one can decide to prefetch and cache any number of result pages following a cache miss.

PDC – Probability Driven Cache

In PDC the cache is divided between an SLRU segment that caches result pages for top-n queries (i.e., queries of the form (qs, $from$, n) where $from = 1$), and a priority queue that caches result pages of follow-up queries. The priority queue estimates the probability of each follow-up results page being queried in the near future by considering all queries issued recently by users, and estimating the probability of at least one user viewing each follow-up page. The priority queue's eviction policy is to remove the page least likely to be queried. One drawback of PDC is that maintaining its probabilistic estimations on the worthiness of each results page requires an amortized time that is logarithmic in the size of the cache per each processed query, regardless of whether the query caused a hit or a miss. An advantage of PDC is that it is better suited for prefetching than other schemes, since prefetched pages are treated separately and independently by the priority queue and are cached according to their individual worthiness rather than according to some fixed prefetching policy.

SDC – Static Dynamic Cache

SDC divides its cache into two areas: the first is a read-only (static) cache of results for the perpetually popular queries (as derived from some pre-analysis on the query log), while the second area dynamically caches results for the rest of the queries using any replacement policy (e.g., LRU or PDC).

Introducing a static cache to hold results for queries that remain popular over time has the following advantages:

- The results of these queries are not subject to eviction in the rare case that the query stream exhibits some "dry spell" with respect to them.
- The static portion of the cache is essentially read-only memory, meaning that multiple query threads

can access it simultaneously without the need to synchronize access or lock portions of the memory. This increases the cache's throughput in a real-life multithreaded system that serves multiple queries concurrently.

SDC can be applied with any prefetching strategy. See the discussion below for the specific prefetching strategy proposed in [14].

AC

AC was proposed by Baeza-Yates et al. in 2006 [2]. Like PDC and SDC, this scheme also divides its cache into two sections. The intuition behind AC is to identify the large fraction of rare "tail" queries and to relegate them to a separate portion of the cache, thus preventing them from causing the eviction of more worthy results.

Prefetching Policies

The prefetching policies appearing so far in the literature are rather simple. One straightforward policy is to simply choose some fixed amount k of result pages to prefetch upon any cache miss. The value of k can be optimized to maximize the cache hit ratio [7] or to minimize the total workload on the query evaluation servers [9]. These two objective functions do not necessarily result in the same value of k, since it could be that it is easier to compute more queries with fewer results per query than less queries with more results per query.

An adaptive prefetching scheme that also prefetches, in some cases, following a cache hit was proposed in [14]:

- Whenever a cache miss occurs for the top-n results of a query (i.e., when $from = 1$), evaluate the query and also prefetch its second page of results.
- For any cache miss of a follow-up query ($from > 1$), evaluate the query and prefetch some fixed number k of additional result pages.
- If the second page of results for a query is requested and is a cache hit, return the cached page and prefetch in the background the next k result pages.

Note that performing query evaluations following cache hits is a speculative operation that does not reduce the overall load on the backend in terms of query evaluations. It may balance that load somewhat and reduce the latency experienced by searchers when submitting follow-up queries.

As explained above, the choice of prefetching policy is orthogonal to the choice of replacement policy. However, PDC and AC are better suited to assigning different priorities to prefetched pages rather than treating them all as "recently used."

Note that prefetching is only effective when caching prefetched pages will result in more cache hits than the pages they replace in the cache. This depends on the characteristics of the cache (its size and replacement policy) as well as on the observed interaction of the users with the specific search engine in question (how often they tend to browse deep result pages).

Key Applications

Caching and Prefetching of results in Web search engines is a key application "*per se*."

Future Directions

Implementing search result caching in live, Web-scale systems brings with it many engineering challenges that have to do with the particular architecture of the specific search engine, and the specific nature of the application. This section briefly points at several such issues, each deserving of further research attention.

Search Result Caching in Incremental Search Engines

The previous sections assumed that the results of query q are stationary over time. In the case of evolving collections cached results may become stale, and serving them defeats the purpose of an incremental engine. Thus, more advanced cache maintenance policies are required.

Caching in the Presence of Personalized Search Results

Search engines are increasingly biasing search results toward the preferences of the individual searchers. As a consequence, two searchers submitting the same query might receive different search results. In this case, cache hit ratios decrease since results cannot always be reused across different searchers.

Level of Details to Cache

A page of search results is comprised of many different components (e.g., result URLs, summary snippets, ads), with each component logically coming from different computational processes and often from different sets of physical servers. While caching is meaningless without saving the URLs of the results, the

other components can be computed separately, and whether to cache or recompute them is a matter of computational tradeoff.

Holistic View

There are many types of objects other than result pages that merit caching in a search engine, e.g., postings lists of popular query terms, user profiles, summary snippets, various dictionaries, and more [12,10]. The optimal allocation of RAM for the caching of different objects is a matter for holistic considerations that depend on the specific architecture of the search engine.

Experimental Results

The literature reports on many experiments with many query logs and caching schemes. All caching schemes have at least one tunable parameter – the cache size – whereas schemes with several cache segments (e.g., SLRU) have parameters governing the size of each segment. Search-specific caching schemes (PDC, SDC, AC) have additional tunable parameters, and when coupled with prefetching schemes, the number of reported experimental configurations grows significantly. Therefore, the summary below focuses on qualitative observations rather than quantitative ones. With respect to caching without prefetching, the literature finds that:

- Even generic caching schemes attain decent hit ratios. For example, LRU achieves on some query logs hit ratios as high as 80% of what an infinite cache would achieve. In terms of absolute values, hit ratios of 20–30% are common for LRU based schemes. In fact, recency-based caching schemes are well suited for the large topical locality of reference exhibited by power-law query distributions, where significant portions of the query stream are comprised of a small set of highly popular queries.
- Search-specific caching schemes outperform generic caching schemes. However, the relative gains in terms of hit ratio are rarely higher than 10%.

Throughout the literature, prefetching improved the achieved cache hit ratios, in many cases by 50% and more as compared with the same experimental setup without prefetching. The following are additional observations:

- The largest relative gains in hit ratio are achieved for moderate amounts of prefetching, namely fetching one or two result pages beyond what is requested in the query. More aggressive prefetching improves the hit ratio by modest amounts at best, and in small caches may even cause it to deteriorate.
- As a general rule, larger caches merit more prefetching, since much of a large cache is devoted to low-merit result pages, which are less likely to be queried than follow-up result pages of current queries.
- Not surprisingly, the adaptive prefetching scheme proposed in [14] outperforms the simplistic policy of always prefetching the same amount of result pages regardless of the identity of the page that generated the cache miss.

Cross-references

- ▶ Buffer Management
- ▶ Cache Replacement as Trade-Off Elimination
- ▶ Cache-Conscious Query Processing
- ▶ Competitive Ratio
- ▶ Indexing the Web
- ▶ Inverted Files

Recommended Reading

1. Baeza-Yates R., Gionis A., Junqueira F., Murdock V., Plachouras V., and Silvestri F. The impact of caching on search engines. In Proc. 33rd Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 2007, pp. 183–190.
2. Baeza-Yates R., Junqueira F., Plachouras V., and Witschel H.F. Admission policies for caches of search engine results. In Proc. 14th Int. Symp. String Processing and Information Retrieval, 2007, pp. 74–85.
3. Belady L.A. A study of replacement algorithms for a virtual-storage computer. IBM Syst. J., 5(2):78–101, 1966.
4. Fagni T., Perego R., Silvestri F., and Orlando S. Boosting the performance of web search engines: caching and prefetching query results by exploiting historical usage data. ACM Trans. Inf. Syst., 24(1):51–78, 2006.
5. Karedla R., Love J.S., and Wherry B.G. Caching strategies to improve disk system performance. Computer, 27(3):38–46, 1994.
6. Lempel R. and Moran S. Optimizing result prefetching in web search engines with segmented indices. In Proc. 28th Int. Conf. on Very Large Data Bases, 2002, pp. 370–381.
7. Lempel R. and Moran S. Predictive caching and prefetching of query results in search engines. In Proc. 12th Int. World Wide Web Conference, 2003, pp. 19–28.
8. Lempel R. and Moran S. Competitive caching of query results in search engines. Theor. Comput. Sci., 324(2):253–271, 2004.
9. Lempel R. and Moran S. Optimizing result prefetching in web search engines with segmented indices. ACM Trans. Internet Tech., 4:31–59, 2004.
10. Long X. and Suel T. Three-level caching for efficient query processing in large web search engines. In Proc. 14th Int. World Wide Web Conference, 2005, pp. 257–266.

11. Markatos E.P. On caching search engine query results. Comput. Commun., 24(2):137–143, 2001.
12. Saraiva P., Moura E., Ziviani N., Meira W., Fonseca R., and Ribeiro-Neto B. Rank-preserving two-level caching for scalable search engines. In Proc. 24th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 2001, pp. 51–58.
13. Silverstein C., Henzinger M., Marais H., and Moricz M. Analysis of a very large altavista query log. Technical Report 1998-014, Compaq Systems Research Center, October 1998.
14. Silvestri F., Fagni T., Orlando S., Palmerini P., and Perego R. A hybrid strategy for caching web search engine results. In Proc. 12th Int. World Wide Web Conference, 2003 (Poster).
15. Sleator D.D. and Tarjan R.E. Amortized efficiency of list update and paging rules. Commun. ACM, 28:202–208, 1985.

Web Search Result De-duplication and Clustering

Xuehua Shen[1], ChengXiang Zhai[2]
[1]Google, Inc., Mountain View, CA, USA
[2]University of lllinois at Urbana-Champaign, Urbana, IL, USA

Definition

Web search result de-duplication and clustering are both techniques for improving the organization and presentation of Web search results. De-duplication refers to the removal of duplicate or near-duplicate web pages in the search result page. Since a user is not likely interested in seeing redundant information, de-duplication can help improve search results by decreasing the redundancy and increasing the diversity among search results.

Web search result clustering means that given a set of web search results, the search engine partitions them into subsets (clusters) according to the similarity between search results and presents the results in a structured way. Clustering results helps improve the organization of search results because similar pages will be grouped together in a cluster and a user can easily navigate into the most relevant cluster to find relevant pages. Hierarchical clustering is often used to generate a hierarchical tree structure which facilitates navigation into different levels of clusters.

De-duplication can be regarded as a special way of clustering search results in which only highly similar (i.e., near-duplicate) pages would be grouped into a cluster and only one page in a cluster is presented to the user. Although standard clustering methods can also be applied to do de-duplication, de-duplication is more often done with its own techniques which are more efficient than clustering.

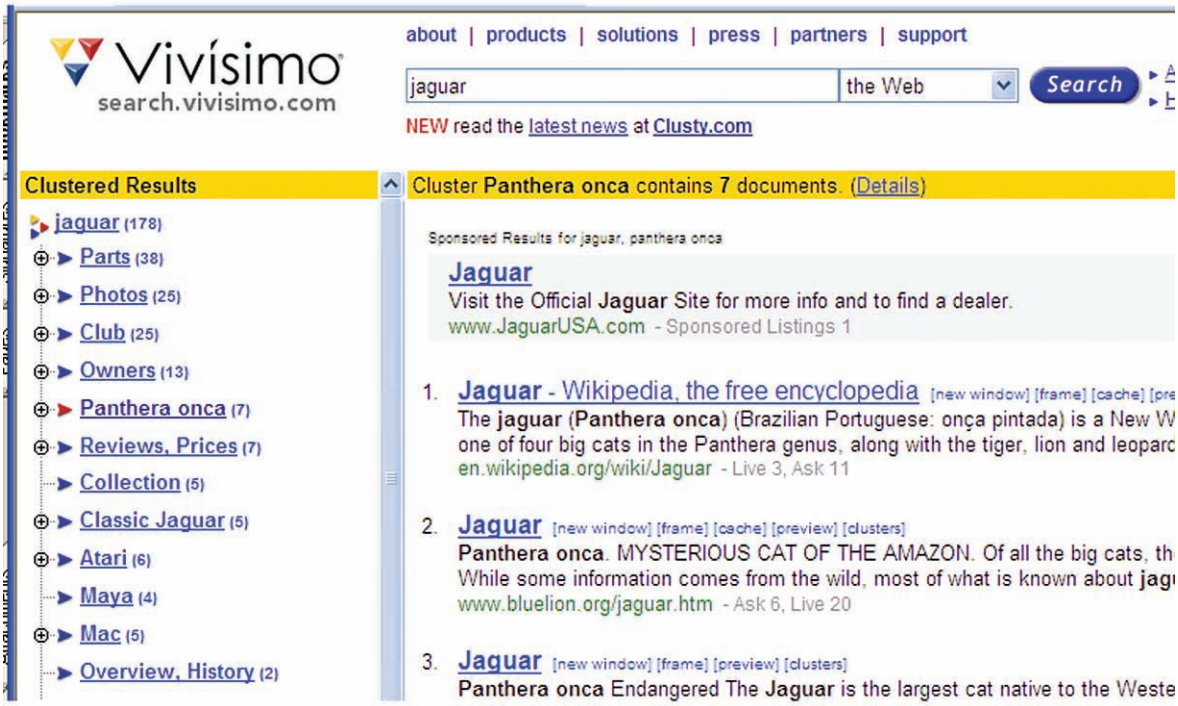
For example, Fig. 1 shows a screenshot of the clustered search results of the query "jaguar" from the vivisimo meta-search engine (http://www.vivisimo.com), which clusters web search results and presents a hierarchical tree of search results. On the left are some clusters labeled with phrases; each cluster captures some aspect of the search results, and different senses of "jaguar" (e.g., jaguar animals vs. jaguar cars) have been separated. On the right are the documents in the cluster labeled as "Panthora onca," which are displayed as the user clicks on this cluster on the left panel.

Historical Background

The idea of clustering search results appears to be first proposed in [6]. However, the application of document clustering to improve the performance of information retrieval has a much longer history [14], dating back to at least 1971 [9]. Most of this work has been motivated by the so-called cluster hypothesis, which says that closely associated documents tend to be relevant to the same requests [9]. The cluster hypothesis suggests that clustering can be potentially exploited to improve both search accuracy (e.g., similar documents can reinforce each other's relevance score) and search efficiency (e.g., scoring clusters takes less time than scoring all the documents in the whole collection). Research has been done in both directions with mixed findings.

In [3], it was proposed that clustering can be used to facilitate document browsing, and a document browsing method called Scatter/Gather was proposed to allow a user to navigate in a collection of documents. In [6], Scatter/Gather was further applied to cluster search results and was shown to be more effective in helping a user find relevant documents than simply presenting results as a ranked list.

Clustering web search results appears to be first studied in [15], which shows that it is feasible to cluster web search results on-the-fly. Vivisimo (http://www.vivisimo.com/) is among the first Web search engines to adopt search result clustering. More recent work in this line has explored presenting search results in clusters (categories) defined based on context (e.g., [4,13]), hierarchical clustering (e.g., [5]), efficiency



Web Search Result De-duplication and Clustering. Figure 1. Sample clustered results about "jaguar" from Vivisimo search engine.

of clustering, incorporating the prior knowledge or user feedback into clustering algorithms in a semi-supervised manner.

Compared with clustering, document de-duplication has a relatively short history. The issue of document duplication was not paid much attention to by information retrieval researchers in early days, probably because it was not a serious issue in the traditional applications of retrieval techniques (e.g., library systems). However, the problem of detecting duplicate or near-duplicate documents has been addressed in some other contexts before it was studied in the context of the Web. For example, the problem has naturally occurred in plagiarism detection. The problem also arose in file systems [10], where one document could be easily copied, slightly modified, saved as another file, or transformed into another format, leading to duplicate or near-duplicate documents. The SCAM system [12] is among the early systems for detecting duplicate or near-duplicate documents. The algorithms proposed in these studies are often based on signatures – representing and comparing documents based on their signatures, or values generated by a hashing function based on substrings from documents.

Web page de-duplication was first seriously examined in [1], where the authors pointed out that document duplication arises in two ways on the Web: one is that the same document is placed in several places, while the other is that the same document are in almost identical incarnations. The authors further proposed an effective de-duplication algorithm based on representing and matching documents based on shingles (i.e., contiguous subsequences of tokens in a documents). Now it has been realized that Web page de-duplication is needed at stages of indexing, ranking, and evaluation [8] to decrease the index storage, increase user satisfaction level, and improve ranking algorithm, respectively.

Foundations

Since de-duplication can be regarded as a special case of clustering, it is not surprising that de-duplication and clustering share very similar fundamental challenges, which include (i) designing an appropriate representation of a document; (ii) designing an effective measure of document similarity based on document representation; (iii) finding/grouping similar documents (duplicate or near-duplicate documents in the case of de-duplication)

quickly. Most research work attempts to solve all or some of these challenges.

How to represent a document and how to measure similarity between documents are closely related to each other in that a similarity function making sense for one representation may not work for another representation. Both highly depend on how the notion of similarity is defined. This is a non-trivial issue as two documents may be similar in many different ways (e.g., in topic coverage, in genre, or in structure), and depending on the desired perspective of similarity, one may need different representations of documents and different ways are needed to measure similarity. As a result, the optimal choice often depends on the specific applications.

For de-duplication, it often suffices to represent documents primarily based on simple tokenization of documents or even surface string representation of documents since duplicate or near-duplicate documents generally share a lot of surface features and may contain very long identical strings. For the same reason, exact matching of two representations may also be sufficient most of the time. Most proposed methods for detecting duplicate or near-duplicate documents can be roughly classified into two categories: signature matching and content matching. They mainly differ in how a document is represented. In signature matching, a document is represented by a signature of the document which is often generated using a hash function based on some (not necessarily meaningful) subsequences of tokens in a document, thus the similarity between documents is mostly measured based on syntactic features of documents. In content matching, the basic unit of document representation is often a meaning unit such as a word or phrase intended to capture the goal of content-matching. Moreover, the basic units are often weighted to indicate how well they reflect the content of a document using heuristic weighting methods such as TF-IDF weighting proposed in information retrieval. Since document representation reflects more the content of a document in the content matching approach, the similarity between documents reflects their semantic similarity better than in the fingerprint matching approach. In general, content matching captures more semantics in de-duplication but is generally more expensive than signature matching. A comprehensive evaluation of these methods can be found in [7].

The shingling algorithm proposed in [1] remains a major state-of-the-art method for Web page de-duplication. In this algorithm, every k subsequence of tokens of a web page is used as a unit of representation (called a shingle), and a page is represented by the set of unique shingles occurring in the page. The similarity between two web pages is measured by the percentage of unique shingles shared by the two pages. Sampling can be used to estimate this similarity and improve efficiency of de-duplication.

In the I-Match method [2], collection statistics is used to filter terms in a signature matching approach, thus achieving content-matching to a certain extent. In a recent work [8], it is shown that additional content-based features can be exploited to significantly improve de-duplication accuracy.

Clustering of search results is almost exclusively based on content matching rather than signature matching because the goal is to group together semantically similar pages which are not always syntactically similar. A document is often represented by a content-capturing feature vector. To construct such a vector, a set of features are selected and assigned appropriate weights to them. Many features may be adopted for clustering, including, e.g., terms from the full content of the html web page, web page snippet (title and summary), URL of the web page, and anchor text. These features need to be weighted appropriately to reflect the content of a page accurately. These challenges are essentially the same as in text retrieval where there is a need to represent a document appropriately to match a document with a query accurately. In general, document similarity can be computed in a similar way to computing the similarity between a query and a document in a standard retrieval setting. Thus many methods developed in information retrieval, especially TF-IDF term weighting and language models, can applied to clustering search results. The basic idea of TF-IDF weighting is to give a term a higher weight if the term occurs frequently in a document and occurs infrequently in the entire collection of documents. Similarity is often computed based on the dot product or its normalized form (i.e., cosine) of two TF-IDF vectors. Language models achieve a similar weighting heuristic through probabilistic models. Once similarities between documents are computed, many clustering methods can be applied, such as k-means, nearest neighbor, hierarchical agglomerative clustering, and spectral clustering. Instead of relying

on an explicit similarity function, a different strategy for clustering is to cast the clustering problem as fitting a mixture model to the text data; the estimated parameter values can then be used to easily obtain clusters. Methods of this kind are called model-based clustering methods. Finite multinomial mixture, probabilistic latent semantic indexing (PLSA), and latent Dirichlet allocation (LDA) are some examples in this category. In model-based clustering, the notion of similarity is implied by the model. Due to the lack of systematic comparisons among these different methods, it is unclear which clustering method is the best for clustering search results.

Efficiency is a critical issue for clustering search results since a search engine must do that on-the-fly after the user submits a particular query. Usually clustering is applied after the search engine retrieves a ranked list of web pages. Thus the time spent on clustering is in addition to the original latency of web search. If the latency is noticeable, the user may be discouraged from using a clustering interface. Some clustering algorithms such as k-means and some model-based algorithms are "any time" algorithms in that they are iterative and can stop at any iteration to produce (less optimal) clustering results. These algorithms would thus be more appropriate for clustering search results when efficiency is a concern. Speeding up clustering is an active research topic in data mining, and many efficient algorithms proposed in data mining can be potentially applied to clustering search results.

Another important challenge in clustering search results is to label a cluster appropriately, which clearly would directly affect the usefulness of clustering, but is yet under-addressed in research. A good cluster label should be understandable to the user, capturing the meaning of the cluster, and distinguishing the cluster from other clusters. In [11], a probabilistic method is proposed to automatically label a cluster with phrases. Another approach to improving cluster labeling and clustering of results in general is to learn from search logs to discover interesting aspects of a query and cluster the search results according to these aspects which can then be naturally labeled with some typical past queries [13].

Key Applications

The major application of search results de-duplication is to reduce the redundancy in search results, which presumably improve the utility of search results. Document de-duplication, however, has many other applications in Web search. For example, it can reduce the size of storage for index, and allow a search engine to load index of more diverse web page sets to the memory. Eliminating duplicate or near-duplicate documents can also reduce the noise in the document set for evaluation, and thus would allow an evaluation measure to reflect more accurately the real utility of a search algorithm [8]. Besides applications for search engines, de-duplication also has many other applications such as plagiarism detection and tracking dynamic changes of the same web page.

Clustering search results generally help a user navigate in the set of search results. It is expected to be most useful when the search results are poor or diverse (e.g., when the query is ambiguous) or when the user has a high-recall information need. Indeed, when the user has a high-precision information need (e.g., navigational queries) and the search results are reasonably accurate, presenting a ranked list of results may be the best and clustering may not be helpful. However, due to the inevitable mismatches between a query and web pages, the search results are more often non-optimal. When the search results are poor, it would take a user a lot of time to locate relevant documents from a ranked list of documents, so clustering is most likely helpful because it would enable a user to navigate into the relevant documents quickly. In the case of poor results due to ambiguity of query words, clustering can be expected to be most useful. When the user has a high-recall information need, clustering can also help the user quickly locate many additional relevant documents once the user identifies a relevant cluster. In addition, the overview of search results generated from clustering may be useful information itself from a user's perspective. Data visualization techniques can be exploited to visualize the clustering presentation as in Kartoo (http://www.kartoo.com). Appropriate visualization may increase the usefulness of clustering search results.

Clustering search results, when used in an iterative way, can help a user browse a web pages collection. In Scatter/Gather [3], for example, the user would interact with an information system by selecting interesting clusters, and the information system would do clustering again after the user selects some clusters. Thus clustering search results can effectively support a user to interactively access information in a collection.

URL to Code

Lemur (http://www.lemurproject.org/). Lemur is an open-source toolkit designed to facilitate research in information retrieval. Document clustering is included.

CLUTO(http://glaros.dtc.umn.edu/gkhome/views/cluto/). CLUTO is a software package for clustering low- and high-dimensional datasets and for analyzing the characteristics of clusters.

Cross-references

- ▶ Data Clustering
- ▶ Data Visualization

Recommended Reading

1. Broder A.Z., Glassman S.C., Manasse M.S., and Zweig G. Syntactic clustering of the web. Comput. Networks, 29(8–13): 1157–1166, 1997.
2. Chowdhury A., Frieder O., Grossman D.A., and McCabe M.C. Collection statistics for fast duplicate document detection. ACM Trans. Inf. Syst., 20(2):171–191, 2002.
3. Cutting D.R., Pedersen J.O., Karger D., and Tukey J.W. Scatter/gather: a cluster-based approach to browsing large document collections. In Proc. 15th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 1992, pp. 318–329.
4. Dumais S.T., Cutrell E., and Chen H. Optimizing search by showing results in context. In Proc. SIGCHI Conf. on Human Factors in Computing Systems, 2001, pp. 277–284.
5. Ferragina P. and Gulli A. A personalized search engine based on Web-snippet hierarchical clustering. In Proc. 14th Int. World Wide Web Conference, 2005, pp. 801–810.
6. Hearst M.A. and Pedersen J.O. 1Reexamining the cluster hypothesis: scatter/gather on retrieval results. In Proc. 19th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 1996, pp. 76–84.
7. Hoad T. and Zobel J. Methods for identifying versioned and plagiarised documents.. J. Am. Soc. Inf. Sci. Technol., 54(3): 203–215, 2003.
8. Huffman S., Lehman A., Stolboushkin A., Wong-Toi H., Yang F., and Roehrig H. Multiple-signal duplicate detection for search evaluation. In Proc. 33rd Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 2007, pp. 223–230.
9. Jardine N. and van Rijsbergen C. The use of hierarchic clustering in information retrieval.. Inf. Storage Retr., 7(5):217–240, 1971.
10. Manber U. Finding similar files in a large file system. In Proc. USENIX Winter 1994 Technical Conference, 1994, pp. 1–10.
11. Mei Q., Shen X., and Zhai C. Automatic labeling of multinomial topic models. In Proc. 13th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, 2007, pp. 490–499.
12. Shivakumar N. and Garcia-Molina H. SCAM: a copy detection mechanism for digital documents. In Proc. 2nd Int. Conf. in Theory and Practice of Digital Libraries, 1995.
13. Wang X. and Zhai C. Learn from Web search logs to organize search results. In Proc. 33rd Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 2007, pp. 87–94.
14. Willett P. Recent trends in hierarchic document clustering: a critical review.. Inf. Process. Manage., 24(5):577–597, 1988.
15. Zamir O. and Etzioni O. Grouper: a dynamic clustering interface to Web search results. In Proc. 8th Int. World Wide Web Conference, 1999.

Web Services

Eric Wohlstadter[1], Stefan Tai[2]
[1]University of British Columbia, Vancouver, BC, Canada
[2]University of Karlsruhe, Karlsruhe, Germany

Synonyms

e-Services

Definition

Web services provide the distributed computing middleware that enables machine-to-machine communication over standard Web protocols. Web services are defined most precisely by their intended use rather than by the specific technologies used, since different technologies are popular [1]. Web services are useful in a compositional approach to application development; where certain key features of an integrated application are provided externally through one or more remote systems. Additionally, Web service standards are a popular platform for wrapping existing legacy applications in a more convenient format for interoperability between heterogeneous systems. To provide interoperability Web services should follow standards for formatting application messages, describing service interfaces, and processing messages. Two popular technology choices discussed in this entry are the SOAP [5] based services and the REST (REpresentational State Transfer) [4] based services.

In contrast to traditional World Wide Web (WWW) resources, Web services decouple the user interface from the underlying programmatic interface, to provide an application programming interface (API) to clients. This API provides the means through which Web services expose remote computational resources or informational content over the Internet. Web services are generally designed to follow a service-oriented architecture which promotes the loose coupling useful for

interaction in a wide-area environment. This loose coupling comes at the price of giving up more powerful mechanisms such as stateful objects which introduce tighter coupling [2,7].

Historical Background

Web services address enterprise application integration (EAI) in a wide area environment. Historically, they were designed in response to the need for a simpler base of standards than provided by distributed object computing. The original WWW was also unsatisfactory due to its tight coupling with the user interface.

Distributed object computing provides the abstraction of remote stateful objects whose methods can be invoked transparently by clients. This form of remote method invocation is popular in proprietary distributed systems but has never been widely deployed on the Internet. A standardized technology known as the Common Object Request Broker Architecture (CORBA) is commonly used to develop and deploy distributed object applications. Although the CORBA platform provides powerful abstractions for application programmers, these abstractions tended to introduce a high level of complexity in the underlying infrastructure. Consensus was never reached on several important technical problems introduced by the stateful nature of objects such as distributed garbage collection and object identification [2,7]. Thus, distributed object computing is more appropriate in single enterprise deployment scenarios (e.g., telecommunications and military applications).

With the rise in acceptance of XML, it made sense to leverage this format as a means of machine-to-machine communication. In creating new standards around this format, certain problematic features such as stateful objects were not included in the newly developed standards leading to more lightweight and loosely coupled Web services.

The WWW was created as a collaborative platform for distributing information through web pages: documents with text and images including the capability to link between related content. The standard format for web pages, Hypertext Markup Language, included mixed elements of both informational content and user interface design. With the tremendous popularity of the WWW, came the opportunity to bring rich up-to-date informational data sources to a large audience. This also required complex business processes to manage requests for data. Since the retrieved data was tangled with user interface code, it was difficult to separate the two concerns for users interested in post-processing the retrieved information. "Wrappers" had to be used to parse, or "scrape" useful content from delivered pages. To simplify the distribution of information, some websites began offering remote APIs for information retrieval which has led to the interest in standardized Web services.

Foundations

Since interoperability is a key element for any Web service, some standardized approach must be taken to export service functions. The most formal approach to building Web services is built around two core standards known as SOAP and the Web Services Description Language (WSDL) [8]. The standards are being developed by organizations such as the W3C and OASIS. Other lightweight and dynamic approaches are generally categorized under the moniker of REST services. The REST approach requires few standardized technologies beyond HTTP. For both approaches, the standard way to identify each service is through means of an Internet Uniform Resource Identifier (URI). To locate an existing service, a URI is typically obtained out-of-band through informal means, although other automated approaches to service discovery are being explored.

In their most basic form, Web services provide only the basic primitives for exchanging documents between clients and servers. Standards for building robust, transactional, and secure services have been proposed but are not currently widely deployed. The remainder of this entry focuses on the core Web services fundamentals and describes how both SOAP/WSDL and REST deal with the details of service implementation: message formatting, interface description, and message processing.

Web services use a human readable message format for transport of application specific data. This practice simplifies the task of debugging and loosens a client's reliance on complex implementations for marshalling data. A popular message format is XML due to its self-described nature and abundance of third-party support for processing and storage. The use of XML is mandated by the SOAP/WSDL approach and is popular in REST services also. Conversion of XML to data structures native for a specific programming language can be done through standardized mappings. These mappings are supported by middleware which automate the mapping process. When REST services

are consumed directly from a browser agent through JavaScript, a format known as JSON (JavaScript Object Notation) [3] can be used. Unlike XML, JSON provides a direct human readable serialization of JavaScript objects. This can reduce problematic mismatches between the differing type systems of XML Schema [10] and JavaScript.

As an API, Web services often provide a number of distinct functions each with their own input requirements and output guarantees. This interface can be specified formally or informally. Formal description of Web service interfaces is provided through WSDL. WSDL is an extension of the XML Schema specification and most importantly provides description of the set of functions exposed by a service. WSDL specifications add the possibility of providing statically typed-checked service use for clients. This is done by implementing message format mapping through a code generation and compilation process.

With REST, the inputs to service functions are specified as standard web URIs, making use of URI parameters for data input. Output can be produced using any standard HTTP supported encoding and XML is often used for this purpose. The original description of the REST architecture placed considerable emphasis on the fact that services should be described as a set of resources which could be manipulated using predefined HTTP methods (i.e., GET, POST, DELETE, etc.), rather than an application specific set of functions. However in practice, the term REST more commonly refers to any service accessible with little technology beyond HTTP, whether or not the service is modeled primarily as resources or functions. REST currently provides no formal treatment for description of services, although discussion of a standard REST description language is underway as of this writing.

Although not required, many Web services middleware platforms are built around a common architectural style referred to as the Chain of Responsibility pattern [6]. On a local scale incoming and outgoing Web service messages pass through a series of components, called handlers or interceptors, before being passed to lower network layers. This architecture promotes loose-coupling within the middleware itself. Handlers addressing separate concerns such as reliability, transactions or security can easily be plugged into the middleware and configured on a per application basis. A large number of standards known as WS-* [9] are being developed to specify the formats and protocols for managing these concerns in middleware implementations.

SOAP provides a standard for encapsulating XML based Web service messages with an envelope that can be used to communicate such extra-functional information such as security tokens, time stamps, etc. Using SOAP, the Chain of Responsibility architecture can be extended to a distributed series of message processing steps. In the distributed scenario each step can be handled at a potentially different intermediary location on the network, giving rise to Web service intermediaries. SOAP also standardizes the fault handling semantics for undeliverable messages and provides a number of message exchange patterns. These patterns capture certain reliability guarantees which can potentially be of use by the underlying middleware for optimizing resource usage.

REST messages are transported according to the semantics of the Hypertext Transport Protocol resource access methods. REST architectural principles mandate the use of stateless protocols wherever possible. REST message processors should consider only point-to-point connections as opposed to an end-to-end connection. This simplifies the interposition of caches or other intermediaries for message processing. REST services should rely only on standardized HTTP message headers for communicating extra-functional processing instructions, to keep implementation infrastructure lightweight.

Key Applications

Web services expose data and application functionality using standard Internet technology for consumption by clients over the Web. For example, Web services are used to provide access to very large product data. Today, the range of Web services offerings spans from very simple services to more advanced offerings in support of specific business requirements. Simple services include format conversion services, transformation services and real-time information services such as news feeds and stock quotes. Business offerings include the full range of e-commerce, CRM, marketing, finance, and communication services, among them verification services such as credit checking, accounting services, or customized notification services. Further, entire business processes such as the shipment of goods are provided as Web services.

The modular nature of Web services supports the de-composition of application functionality and business processes and their (re-)composition into new applications. Services composition applies both

within an organization and across different organizations. Compositions can take various forms, including workflow applications that require the use of a process composition language and centralized workflow engine. Web services in this way support EAI using standard Internet technology. Another form of composition are Mashups, re-purposing Web-accessible content and functionality in new Web applications that provide an interactive (human) end-user experience. For example, a Mashup may visualize location information of social event feeds on a map.

Web services are also being used in re-architecting middleware software platforms as service-oriented architectures. Middleware services such as message queuing and data storage are provided as Web services; middleware functionality can thus be externalized and used as a remote service. This model targets small and medium sized enterprises in particular, but applies equally to larger size enterprises as well. More advanced services offerings include application hosting and execution services, which allow clients to have entire software applications hosted and executed externally.

Future Directions

Today, there exists a variety of Web services protocols, APIs, languages, and specifications. Different combinations of Web services technology are being used for different purposes. For example, SOAP and Web services specifications in support of security and reliability are used in enterprise computing, whereas REST services are popular for simple services like data access and applications in the social Web.

From a business viewpoint, Web services establish a new market for trading and contracting electronic services that complement and transform the traditional consulting services and software market. The development of this market today is accompanied by the emergence of intermediaries for buyers and sellers to exchange services. These intermediaries distinguish themselves using different pricing mechanisms, co-marketing efforts, billing and accounting management, services usage monitoring, data integration models, and infrastructure quality guarantees for services availability. New Web services technology in support of this business development is expected to emerge.

Cross-references

- ▶ AJAX
- ▶ Coupling and De-Coupling
- ▶ Discovery
- ▶ Enterprise Application Integration
- ▶ Interface
- ▶ Loose Coupling
- ▶ Mashups
- ▶ OASIS
- ▶ Request Broker
- ▶ Service Oriented Architecture
- ▶ SOAP
- ▶ Tight Coupling
- ▶ W3C
- ▶ Web 2.0 (3.0)

Recommended Reading

1. Alonso G., Casati F., Kuno H., and Machiraju V. Web Services: Concepts Architectures and Applications. Springer, Berlin, 2003.
2. Birman K. Like it or not, web services are distributed objects. Commun. ACM, 47(12):60–62, 2004.
3. Crockford D. The application/json media type for JavaScript object notation. Network Working Group, RFC 4627, 2006.
4. Fielding R. Architectural Styles and the Design of Network-based Software Architectures. Ph.D. Dissertation, University of California, 2000.
5. SOAP Version 1.2. W3C Recommendation, 2007.
6. Vinoksi S. Chain of responsibility. IEEE Internet Comput., 6(6):80–83, 2002.
7. Vogels W. Web Services are not Distributed Objects. IEEE Internet Comput., 7(6):59–66, 2003.
8. Web Services Description Language Version 2.0. W3C Recommendation, 2007.
9. Weerawarana S., Curbera F., Leymann F., Storey T., and Ferguson D. Web Services Platform Architecture. Prentice Hall, Upper Saddle River, NJ, 2005.
10. XML Schema. W3C Recommendation, 2004.

Web Services and the Semantic Web for Life Science Data

Cartik R. Kothari, Mark D. Wilkinson
University of British Columbia, Vancouver, BC, Canada

Definitions

Web services are frameworks for communication between computer applications on the World Wide Web. A general feature of "canonical" Web services is that they expose Web-based application interfaces in the form of a Web Services Description Language (WSDL) document with machine-readable content describing the input(s), output(s), and location of an application. The *Semantic*

web extends the traditional Web by applying human and machine-readable labels to the links between resources, and encouraging those resources to be "typed" into a set of well-grounded categories, defined by shared *ontologies*. Machines can thus explore and process the content of the Semantic Web in meaningful ways. Moreover, the Semantic Web moves beyond simple documents and allows linking of individual data points, analytical tools, and even conceptual entities with no physical representation. O*ntologies* are formal descriptions of a knowledge domain, and range from simple controlled vocabularies to explicit logical definitions of the defining properties for all concepts and inter-concept relationships within that knowledge domain. In the *life sciences*, with its multitude of specialized data repositories and data processing applications, the Semantic Web paradigm promises to overcome the hurdles to data discovery, integration, mining, and reuse that continues to hinder biomedical investigation in the post-genomic era.

Historical Background

The life sciences are experiencing what might best be described as a feedback-loop – the ability of life scientists to generate raw "omics" data in vast quantities is perhaps only exceeded by the quantity of data output from tools analyze this raw data, and both become the fodder for new downstream analyses. Though much of the data is stored in one of several generalized Web-based repositories – for example UniProt (http://www.pir. uniprot.org), GenBank (http://www.ncbi.nlm.nih.gov/GenBank), or EMBL (http://embl.org) – still more data is housed in specialized repositories such as Gene Expression Omnibus (http://www.ncbi.nlm.nih.gov/geo/) (GEO), for Gene Expression Data, or species-specific Web databases such as Wormbase (http://www.wormbase.org/), Flybase (http://flybase.bio.indiana.edu/), DragonDB (http://antirrhinum.net), and TAIR (http://www.arabidopsis.org/). Similarly, many common analytical tools are also made available on the Web, through traditional CGI or JavaScript interfaces. In the course of their daily experimental investigations, Biologists frequently need to use various combinations of these distributed data and/or analytical resources. For most biologists, this is achieved by manual copy/paste of the output from one resource into the input form fields of the next. Computer-savvy biologists might undertake to construct "screen scrapers" – small computer programs that automatically process the output from Web resources and pipelined data from one resource to the next – and this is further facilitated by the creation of centralized code repositories such as BioPerl (http://www.bioperl.org) and BioJava (http://biojava.org) that provide a uniform API into a standard and curated set of "scrapers." Nevertheless, automation of data extraction from traditional Web forms is fraught with difficulties; with the lack of a common data interchange format and shareable semantics being the biggest hurdle.

The introduction of Web Services provided a more standardized, and platform/language-independent method of exposing data and analytical tools on the Web, and encouraged the use of XML as the standard data exchange format between different web resources. Moreover, the interface to and functionality of a Web Service could be described in a standardized machine processable syntax – Web Services Description Language (http://www.w3.org/TR/2007/REC-wsdl20-primer-20070626/) (WSDL), and a novel transport protocol – Simple Object Access Protocol (http://www.w3.org/TR/2007/REC-soap12-part0–20070427/) (SOAP) – was introduced to hide the complexities of the data exchange process and thus facilitate design of simple object- oriented accessors to remote Web resources. A paradigm for Web Services discovery (UDDI) was also proposed; however this has not been widely adopted, and will not be discussed here.

Although Web Services greatly aided the standardization of interfaces, the problem of interoperability had not been solved – the pipelining of Web Services continued to require sophisticated knowledge of the interfaces such that appropriate data-structures could be extracted from one service and fed into the next. This is because the XML data syntax lacks semantic grounding, and thus the input and output to/from any Web Service is opaque to automated interpretation, which thwarts attempts at automated Web Service workflow composition. The advent of the Semantic Web and ontologies constructed with Semantic Web compatible knowledge representation formalisms promises to overcome this hurdle.

The Semantic Web is being implemented as a Web of machine processable data. This is in contrast to the first version of the World Wide Web where machines were primarily concerned with the presentation of data; consumption and interpretation were limited to humans. Data on the Semantic Web is rendered machine processable by annotation with syntactic constructs

whose semantics are grounded in formal mathematical logic. Knowledge representation formalisms such as the Resource Description Framework (http://www.w3.org/RDF/) (RDF) and the Web Ontology Language (http://www.w3.org/2004/OWL/) (OWL) provide syntactic constructs with logically grounded semantics that can be used to annotate web services.

Ontologies such as the OWL-Services (http://www.w3.org/Submission/2004/SUBM-OWL-S-20041122/) ontology and the Web Services Modeling Ontology (http://www.w3.org/Submission/WSMO/) (WSMO) are fresh initiatives to leverage the Semantic Web infrastructure for web service composition. A web service on the Semantic Web can be described by instances of concepts from the OWL-S or WSMO and easily invoked and composed with other compatible web services.

Foundations

Web Services

Web Services have revolutionized the field of distributed computation. Web Services enable the remote discovery and execution of software applications across the dimensions of the Web in an implementation agnostic manner. According to the World Wide Web Consortium (W3C), a Web Service is "a software system designed to support interoperable machine to machine interaction over a network." The functionality of Web Services can be easily understood from three different perspectives viz. the service provider, the service client, and the service registry.

The Service Provider A service provider is a person or organization that is responsible for the development and maintenance of the Web Service. The service provider is familiar with the implementation details of the service, such as the architecture of the underlying application and the programming language used. These details are encapsulated by the Web Service interface, which describes the function of the Web Service, its location, input parameters and output parameters and their data-types. Consider an application that performs a search for relevant publication in an online, open access repository such as PubMed (http://www.pubmed.org). This application would be encapsulated in a web interface (the Web Service), which advertises the function of the service (searching for relevant publications), the location of the service (a resolvable URI, usually a URL), the input parameters (keyword, name of the author, citation details such as journal name and/or date of publication, all of which would be string-literal data-types), and the output parameters of the service (title of the publication, abstract, and citation details; all of which are string-literals). This Web Service description is published as a WSDL document and is (optionally) registered in a registry. Note the description does not include implementation details. The underlying application could be implemented on a Linux or Microsoft platform, using any programming or scripting language desired. More importantly, can also be accessed by a client application on any platform using any language.

The Service Client The service client locates a specific Web Service from its description. The client then remotely invokes the discovered service and executes it. In the PubMed example, consider a service client that is looking for publications about the Vascular Endothelial Growth Factor (VEGF) protein. The service client discovers the book browsing service from its WSDL description. The WSDL description details the input parameters accepted by the Web Service, one of which is "keyword." The client sends the text string "VEGF" into the interface as the "keyword" parameter and thereby invokes the service. The communication between the client and the service generally utilizes the HTTP protocol, and is often supplemented by a standardized messaging format such as SOAP.

Note that "keyword" is a *human-readable* label indicating the purpose of that input parameter – the WSDL interface definition must be interpreted by a person in order to determine the intent of that parameter, emphasizing the point that, in traditional Web Services, the interfaces are opaque due to the lack of ontological grounding of XML tags.

The Service Registry The service registry acts as a "yellow pages" for Web Service discovery, and brokers the direct interaction between a client and a chosen service. Specifically, the registry holds WSDL descriptions of Web Services, including their functional characteristics such as input and output parameters, and location. In addition, human readable descriptions of the web service may also be included in the WSDL description. Details of the service provider, and classification information (e.g., the UNSPSC category of the service), and reliability estimates may also be provided. In addition, Web Service registries may perform

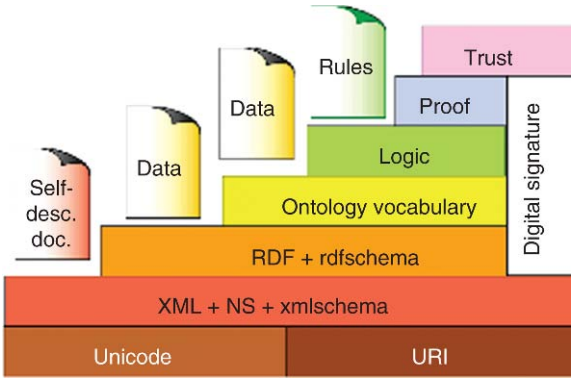
ancillary tasks such as periodic polling to identify inactive services, and collecting performance metrics and reliability estimates.

The Semantic Web

The Semantic Web initiative aims to make the vast amounts of information on the Web processable by machines. Previously, computers were involved only in the presentational aspects of this information, which was marked up with tags from the HyperText Markup Language (http://www.w3.org/html/) (HTML). HTML tags specify how the tagged information is to be presented on a browser. However, there are no semantics associated with HTML tags, nor are there semantics associated with the hyperlinks between documents, thus preventing automated interpretation of the contained information. The development of knowledge representation formalisms for the Semantic Web such as RDF, the Ontology Inference Layer (http://www.ontoknowledge.org/oil/) (OIL), the DARPA Advanced Markup Language (http //www.daml.org) (DAML), from the Defense Advanced Research Projects Agency (http://www.darpa.mil) (DARPA), and OWL have been an attempt to overcome these limitations of the Web.

OWL is the most advanced formalism for markup on the Semantic Web, providing a very expressive and decidable set of constructs for ontology development, and has been adopted by the W3C as a standard. The semantics of constructs from OWL (and OIL and DAML) are grounded in mathematical logic. Therefore, the information content is now accessible to computers and can be used by logical inference axioms to deduce implicit knowledge. Lastly, the information content can be queried by machines across the Web. Intelligent, semantic based searches become a possibility, replacing the prevalent simple keyword based searches and word and hyperlink based algorithms. Three components are critical to understanding the Semantic Web architecture; ontologies, query engines and inference engines. Figure 1 shows the Semantic Web architecture proposed by Tim Berners-Lee.

Ontologies Ontologies are machine interpretable models of knowledge domains. An ontological representation of a knowledge domain contains logically grounded definitions of concepts and inter-concept relations that are specific to that domain. Since these concepts and relations are defined in a rigorous logical framework, their semantics can be shared and processed by machines without loss of clarity. The information content of Web pages on the Semantic Web can be marked up as instances of ontologically defined concepts and relations. Knowledge representation formalisms such as OWL can be used to construct ontologies of different knowledge domains. Ontology editing tools such as Protégé (http://protege.stanford.edu/), TopBraid suite (http://www.topquadrant.com/topbraid suite.html), and Altova SemanticWorks (http://www.altova.com/products/semanticworks/semantic_web_rdf_owl_editor. html) have made the process of ontology development very straightforward for knowledge do main experts. This is attested by the proliferation of ontologies, especially in the life sciences domain.



Web Services and the Semantic Web for Life Science Data. Figure 1. Proposed semantic web architecture (Berners-Lee et al., 2001).

Query Engines Marking up the information content of Web pages with semantically rich HTML tags makes it possible for query engines and languages to retrieve this information. Two of the earliest query languages for retrieving RDF annotated information were SPARQL (http://www.w3.org/TR/rdf-sparql-query/) and the RDF Query Language (http://139.91.183.30:9090/RDF/RQL/) (RQL). SeRQL (//www.openrdf.org/doc/sesame/users/ch06.html), nRQL [9], and RDQL (http://www.w3.org/Submission/2004/SUBM-RDQL-20040109/) are other querying formalisms that have been developed to retrieve RDF annotated documents from the Semantic Web.

Inference Engines Inference is the process of deducing implicit information from the combination of explicitly stated information and inference axioms. For example,

given that any person who is the parent of a parent is a grandparent (the axiom), and given that Lisa is the parent of Charles and Charles is the parent of Harry (explicitly stated facts), it can be deduced that Lisa is a grandparent of Harry (inference). In this fashion, logical inference axioms can be used to reason with the concept (and relation) definitions and their instances. The information content of the Semantic Web can be reasoned with by specialized inference engines such as FaCT + + (http://owl.man.ac.uk/factplusplus/), Pellet (http://pellet.owldl.org), and RACERPro (http://www.racer-systems.com/products/racerpro/index.phtml).

Semantic Web Services

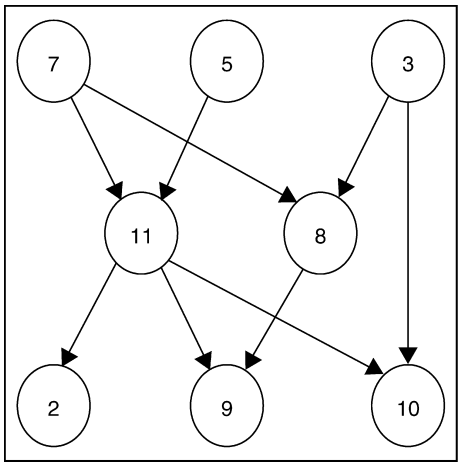
The possibility of marking up Web Services with semantically expressive tags led to the adoption of the Semantic Web Services initiative (http://www.swsi.org/). The objective of the Semantic Web Services initiative is the use and leverage of the framework of Semantic Web technologies to enable the automation and dynamism of Web Services annotation, publication, discovery, invocation, execution, monitoring, and composition. Referring again to the PubMed search example, the input and output parameters of the search web service can be described using ontologically defined concept instances. WSDL provides a means to specify the acceptable input parameters of the Web Service to be one of three string literals, viz. keyword, citation details, and author name. However, the fact that the author name, citation details, and keyword are specialized types of string literals cannot be explicitly stated with WSDL. Using an ontology to ground these Web Service parameters as specific "types" of entity, it then becomes possible to indicate, to both human and machine, the purpose of each parameter field. Moreover, the concept of a "Literature Search Tool" could be defined as a Web Service that had parameters of type "author" and "keyword." An inference engine could then be used to discover this PubMed Web Service in response to a user-request for Literature Search Tools.

Semantic Web Services in the Life Sciences

Computational methods, also known as *in silico experiments*or *workflows*, complement in vivo and in vitro methods in biological research, and provide high-throughput access to tools that can manage the large volumes of genomic, proteomic, and other types of data required by modern biomedical experiments. Since many analytical tools are now available as Web Services, it is now possible to "pipeline" these tools together to create *in silico*workflows, with the biologist manually connecting the output of one service to the input of the next based on a human readable WSDL description. This, however, is a complex task, and requires the biologist to be aware of the individual capabilities and operating principles of each Web Service. Availability of a comprehensive textual description of a Web Service's capabilities and operating principles is rare, and not always helpful to a biologist who is not computer-savvy. Therefore, the constructed workflows maybe error-prone and/or procedurally flawed. Moreover, many applications go through versions or become unavailable from time to time, making workflows somewhat fragile. The burden of "re-wiring" a workflow to accommodate a new version of an interface; replacing a dead interface with an equivalent functional one; and discovering new or improved applications as and when they become available, is also placed on the biologist and requires knowledge and awareness outside of their domains of expertise. Machine processable descriptions of Web Services are, thus, crucial to enabling their automatic discovery and invocation, and most importantly, to their automated composition into stable, self-repairing analytical workflows.

Registries of semantically annotated Web Services such as myGrid (http://www.mygrid.org.uk/) and Moby Central (http://biomoby.org/RESOURCES/MOBY-S/ServiceInstances) have been used to facilitate the discovery and use of Web Services by biologists. However, biologists still need to familiarize themselves with datatype hierarchies to discover web services that can process their data. Recent innovations capable of inferring datatypes from actual data [4] promise to relieve biologists of this requirement as well. The automated composition of Web services is more troublesome. Automated pipelining the output of one web service into the input of another is the simplest of the proposed approaches to service composition. In mathematical terms, this is analogous to finding a route in a directed graph. A web services network can be visualized as a directed graph as shown in Figure 2.

Every node in Figure 2 corresponds to a web service. Directed edges leading to a webservice correspond to input parameters, and those leading away from a



Web Services and the Semantic Web for Life Science Data. Figure 2. A directed graph representation of a web service network.

service, to output parameters. Finding a route from node 7 (which could represent a service that queries a genome database given an accession number, for instance) to node 10 (which may represent a service that performs a sequence comparison and outputs a report) in this figure is a computationally complex problem, capable of consuming large extents of computational time and memory. Even with finding the shortest possible route in the figure from node 7 to node 10, uninformed search algorithms are exponentially complex. This means that the time taken to find a solution (and the computational memory used) increases exponentially as the number of possible solutions. Heuristics are required to prune the search space in such problems and reduce the complexity to decidable and tolerable limits. In web service composition, the complexity of the problem is compounded due to several factors. Some services are asynchronous, meaning they take in input parameters without outputting anything. Other services may be notifiers, giving output parameters without (seemingly) taking in any input. Some edges many not necessarily lead to other services, being dead end points in the route finding algorithm. Lastly, the shortest combination of services may not be desirable to the biologist as well. The automated composition of web services is therefore, an open and fascinating research problem. The interested reader may refer the resources at the Web Services Choreography Working Group (http://www.w3.org/2002/ws/chor/) Web site at the W3C for more information.

Key Applications

BioMoby [10] is an open source, extensible framework that enables the representation, discovery, retrieval, and integration of biological Web Services. By registering their analysis and data access services with BioMoby, service providers agree to use and provide service specifications in a shared semantic space. The BioMoby Central registry now hosts more than a thousand services in the United States, Canada, and several other countries across the world. BioMoby uses a datatype hierarchy to facilitate automated discovery of Web Services capable of handling specific input datatypes. As a minimal Web based interface, the Gbrowse Moby (http://moby.ucalgary.ca/gbrowse_moby) service browser can be used by biologists to discover and invoke biological web services from the Moby registry and seamlessly chain these services to compose multi-step analytical workflows. The process is data centric, relying on input and output datatype specifications of the services. The Seahawk client interface [4] can infer the datatype of the input data files directly and immediately present the biologist with a list of BioMoby Web Services that can process that input file. Users of the Seahawk interface are relieved of the necessity to familiarize themselves with datatype hierarchies, and instead are free to concentrate on the analytical aspects of their work. Other clients for BioMoby services include Remora (http://lipm-bioinfo.toulouse.inra.fr/remora/cgi/remora.cgi), Ahab (http://bioinfo.icapture.ubc.ca/bgood/Ahab.html), and MOWserv (http://www.inab.org/MOWServ/).

myGrid [3] is an example of a computational grid architecture that brings together various computational resources to support *in silico* biological experiments. The computational resources include analysis and data retrieval services. As such, myGrid is a service grid and the myGrid philosophy regards *in silico* experiments to be distributed queries and workflows. Taverna [5], a module of myGrid, provides a means to integrate the diverse resources on the myGrid framework into reusable *in silico* workflows. Grimoires [1] is a semantics enabled service registry that enables discovery of myGrid services. Taverna also provides an interface to access BioMoby services [6], a promising development towards the integration of research initiatives in the area of biomedical informatics.

The PathPort framework [11] developed at The Virginia Bioinformatics Institute presents a Web based interface that makes it possible for end users to invoke local and distributed biological Web Services that are

described in WSDL, in a location and platform independent manner. Sembowser [8] is an ontology based implementation of a service registry that facilitates Web Service annotation, search and discovery based upon domain specific keywords whose semantics are grounded in mathematical logic. Sembowser associates every published Web Service with concepts defined in an ontology and further, classifies the published service on the basis of the task it performs and the domain with which it is associated.

The Simple Semantic Web Architecture and Protocol (http://semanticmoby.org) (SSWAP), is a Semantic Web Services architecture that uses ontology based descriptions of biological Web Services from the Virtual Plant Information Network (http://vpin.ncgr.org/) (VPIN) to discover, and remotely execute Web services. Formerly known as Semantic Moby, this architecture is the closest realization of the conventional definition of a Semantic Web Services framework, and utilizes the knowledge embedded in any Web-based ontology to assist in discovery and pipelining of Web Services.

The need for semantic descriptions to enable automated Web Service discovery, invocation and composition has led to the development of prototypical ontology based frameworks. The OWL-S framework, the METEOR-S framework (http://lsdis.cs.uga.edu/projects/meteor-s/), and the Web Services Modeling Framework (WSMF) [2] are examples of research initiatives in this area. Ontology based Web Service architectures are in very early stages of development and their evolution into industrial strength technologies will be eagerly anticipated.

Web services with machine processable descriptions are of great value in the development of mashups. A mashup is a composite application that brings together data and functionality from diverse sources using technologies such as AJAX and RSS. Mashup development has been of specific interest to the Web 2.0 community, with its emphasis on mass collaborative information gathering techniques, as exemplified by Wikipedia (http://www.wikipedia.org), Flickr (http://www.flickr.com), and Delicious (http://del.icio.us). Very recently, diverse applications have been embedded with Semantic Web ontologies, query engines, and inference engines to create mashups. Mashups have been used to answer specific questions about genes responsible for Alzheimer's disease and also visualize these genes [7]. In this demo, results from SPARQL queries have been integrated with data extracted from the Allen Brain Atlas site (http://www.brainatlas.org/aba/) using screen scraping techniques, and finally visualized using Google Maps (http://maps.google.com) and the Exhibit (http://simile.mit.edu/exhibit/)visualization toolkit.

Future Directions

The Semantic Web, with its promise of information sharing, reuse and integration, has seen widespread adoption by the life sciences community. Currently, the curators of many of the major genome, proteome, transcriptome, and interactome repositories are adopting Semantic Web standards, while more and more service providers are gravitating towards the Semantic Web Services paradigm. Semantic Web standards are crucial to a synergistic approach to a full understanding of, and representation of, complexity in Life Sciences.

Cross-references

- ▶ Controlled Medical Vocabulary
- ▶ Data Structures and Models for Biological Data Management
- ▶ Data Types in Scientific Data Management Systems
- ▶ Grid and Workflows
- ▶ Mashups
- ▶ Ontologies
- ▶ Ontologies and Life Science Data Management
- ▶ Ontologies in Scientific Data Integration
- ▶ Ontology
- ▶ Ontology-Based Data Models
- ▶ OWL: Web Ontology Language
- ▶ Query Languages for Ontological Data
- ▶ Query Languages for the Life Sciences
- ▶ RDF
- ▶ Scientific Workflows
- ▶ Screen Scraper
- ▶ Semantic Web
- ▶ Semantic Web Query Languages
- ▶ Semantic Web Services
- ▶ SOAP
- ▶ W3C
- ▶ Web 2.0 (3.0)
- ▶ Web Services
- ▶ Workflow Management and Workflow Management Systems
- ▶ Workflow Model
- ▶ Workflow Modeling
- ▶ XML

Recommended Reading

1. Fang W, et al. Performance analysis of a semantics enabled service registry. In Proc. 4th All Hands Meeting. Nottingham, UK, 2005. Available at: http://users.ecs.soton.ac.uk/lavm/papers/Fang-AHM05.pdf
2. Fensel D. and Bussler C. The Web Services Modeling Framework (WSMF), Electron. Commerce Res. Appl., 1(2):113–137, 2002.
3. Goble C., Pettifer S., Stevens R., and Greenhalgh C. Knowledge integration: in silico experiments in bioinformatics. In the Grid: Blueprint for a New Computing Infrastructure (2nd edn.), I. Foster, C. Kesselman (eds.). Morgan Kaufmann, Los Altos, CA, 2003.
4. Gordon P. and Sensen C. Seahawk: Moving beyond HTML in Web-based Bioinformatics Analyses. BMC Bioinformatics, 8(June):208, 2007.
5. Hull D., Wolstencroft K., Stevens R., Goble C., Pocock M., Li P., and Oinn T. Taverna: a tool for building and running workflows of services. Nucleic Acids Res., 34: W729–W732, 2006.
6. Kawas E., Senger M., and Wilkinson M. BioMoby extensions to the taverna workflow management and enactment software. BMC Bioinformatics, 7:523, 2006.
7. Ruttenberg A, et al. Advancing translational research with the semantic web. BMC Bioinformatics, 8(Suppl. 3):S2, May 2007.
8. Sahoo S., Sheth A., Hunter B., and York W. Sembowser: adding semantics to a biological web services registry. In Semantic Web: Revolutionizing Knowledge Discovery in the Life Sciences, C. Baker, K. Cheung (eds.). Springer 2007.
9. Wessel M. and Möller R. A high performance semantic web query answering engine. In Proc. Int. Workshop on Description Logics, 2005.
10. Wilkinson M. and Links M. BioMOBY: an open-source biological web services proposal. Brief. Bioinform., 3(4): 331–341, 2002.
11. Xue T., Yang B., Will R., Sharp B., Kenyon R., Crasta O., and Sobral B. A generalized framework for pathosystems informatics and bioinformatics web services. In Proc. 2007 Int. Conf. on Bioinformatics and Computational Biology, 2007.

Web Services Business Process Execution Language

► Composed services and WS-BPEL

Web Site Wrappers

► Languages for Web Data Extraction

Web Spam Detection

Marc Najork
Microsoft Research, Mountain View, CA, USA

Synonyms

Spamdexing; Google bombing; Adversarial information retrieval

Definition

Web spam refers to a host of techniques to subvert the ranking algorithms of web search engines and cause them to rank search results higher than they would otherwise. Examples of such techniques include content spam (populating web pages with popular and often highly monetizable search terms), link spam (creating links to a page in order to increase its link-based score), and cloaking (serving different versions of a page to search engine crawlers than to human users). Web spam is annoying to search engine users and disruptive to search engines; therefore, most commercial search engines try to combat web spam. Combating web spam consists of identifying spam content with high probability and – depending on policy – downgrading it during ranking, eliminating it from the index, no longer crawling it, and tainting affiliated content. The first step – identifying likely spam pages – is a classification problem amenable to machine learning techniques. Spam classifiers take a large set of diverse features as input, including content-based features, link-based features, DNS and domain-registration features, and implicit user feedback. Commercial search engines treat their precise set of spam-prediction features as extremely proprietary, and features (as well as spamming techniques) evolve continuously as search engines and web spammers are engaged in a continuing "arms race."

Historical Background

Web spam is almost as old as commercial search engines. The first commercial search engine, Lycos, was incorporated in 1995 (after having been incubated for a year at CMU); and the first known reference to "spamdexing" (a combination of "spam" and "indexing") dates back to 1996. Commercial search engines began to combat spam shortly thereafter, increasing their efforts as it became more prevalent.

Spam detection became a topic of academic discourse with Davison's paper on using machine learning techniques to identify "nepotistic links," i.e., link spam [4], and was further validated as one of the great challenges to commercial search engines by Henzinger et al. [9]. Since 2005, the workshop series on *Adversarial Information Retrieval on the Web* (AIRWeb) provides a venue for researchers interested in web spam.

Foundations

Given that the objective of web spam is to improve the ranking of select search results, web spamming techniques are tightly coupled to the ranking algorithms employed (or believed to be employed) by the major search engines. As ranking algorithms evolve, so will spamming techniques. For example, if web spammers were under the impression that a search engine would use click-through information of its search result pages as a feature in their ranking algorithms, then they would have an incentive to issue queries that bring up their target pages, and generate large numbers of clicks on these target pages. Furthermore, web spamming techniques evolve in response to countermeasures deployed by the search engines. For example, in the above scenario, a search engine might respond to facetious clicks by mining their query logs for many instances of identical queries from the same IP address and discounting these queries and their result click-throughs in their ranking computation. The spammer in turn might respond by varying the query (while still recalling the desired target result), and by using a "bot-net" (a network of third-party computers under the spammer's control) to issue the queries and the click-throughs on the target results.

Given that web spamming techniques are constantly evolving, any taxonomy of these techniques must necessarily be ephemeral, as will be any enumeration of spam detection heuristics. However, there are a few constants:

- Any successful web spamming technique targets one or more of the features used by the search engine's ranking algorithms.
- Web spam detection is a classification problem, and search engines use machine learning algorithms to decide whether or not a page is spam.
- In general, spam detection heuristics look for statistical anomalies in some of the features visible to the search engines.

Web Spam Detection as a Classification Problem

Web spam detection can be viewed as a binary classification problem, where a classifier is used to predict whether a given web page or entire web site is spam or not. The machine learning community has produced a large number of classification algorithms, several of which have been used in published research on web spam detection, including decision-tree based classifiers (e.g., C4.5), SVM-based classifiers, Bayesian classifiers, and logistic regression classifiers. While some classifiers perform better than others (and the spam detection community seems to favor decision-tree-based ones), most of the research focuses not on the classification algorithms, but rather on the features that are provided to them.

Taxonomy of Web Spam Techniques

Content spam refers to any web spam technique that tries to improve the likelihood that a page is returned as a search result and to improve its ranking by populating the page with salient keywords. Populating a page with words that are popular query terms will cause that page to be part of the result set for those queries; choosing good combinations of query terms will increase the portion of the relevance score that is based on textual features. Naïve spammers might perform content spam by stringing together a wide array of popular query terms. Search engines can counter this by employing language modeling techniques, since web pages that contain many topically unrelated keywords or that are grammatically ill-formed will exhibit statistical differences from normal web pages [11]. More sophisticated spammers might generate not a few, but rather millions of target web pages, each page augmented with just one or a few popular query terms. The remainder of the page may be entirely machine-generated (which might exhibit statistical anomalies that can be detected by the search engine), entirely copied from a human-authored web site such as Wikipedia (which can be detected by using near-duplicate detection algorithms), or stitched together from fragments of several human-authored web sites (which is much harder, but not impossible to detect).

Link spam refers to any web spam technique that tries to increase the link-based score of a target web page by creating lots of hyperlinks pointing to it. The hyperlinks may originate from web pages owned and

controlled by the spammer (generically called a *link farm*), they may originate from partner web sites (a technique known as *link exchange*), or they may originate from unaffiliated (and sometimes unknowing) third parties, for example web-based discussion forums or in blogs that allow comments to be posted (a phenomenon called *blog spam*). Search engines can respond to link spam by mining the web graph for anomalous components, by propagating distrust from spam pages backwards through the web graph, and by using content-based features to identify spam postings to a blog [10]. Many link spam techniques specifically target Google's PageRank algorithm, which not only counts the number of hyperlinks referring to a web page, but also takes the PageRank of the referring page into account. In order to increase the PageRank of a target page, spammers should create links on sites that have high PageRanks, and for this reason, there is a marketplace for expired domains with high PageRank, and numerous brokerages reselling them. Search engines can respond by temporarily dampening the endorsement power of domains that underwent a change in ownership.

Click spam refers to the technique of submitting queries to search engines that retrieve target result pages and then to "click" on these pages in order to simulate user interest in the result. The result pages returned by the leading search engines contain client-side scripts that report clicks on result URLs to the engine, which can then use this implicit relevance feedback in subsequent rankings. Click spam is similar in method to *click fraud*, but different in objective. The goal of click spam is to boost the ranking of a page, while the goal of click fraud (generating a large number of clicks on search engine advertisements) is to spend the budget associated with a particular advertisement (to hurt the competitor who has placed the ad or simply to lower the auction price of said ad, which will drop once the budget of the winning bidder has been exhausted). In a variant of click fraud, the spammer targets ads delivered to his own web by an ad-network such as Google AdSense and obtains a revenue share from the ad-network. Both click fraud and click spam are trivial to detect if launched from a single machine, and hard to detect if launched from a bot-net consisting of tens of thousands of machines [3]. Search engines tackle the problem by mining their click logs for statistical anomalies, but very little is known about their algorithms.

Cloaking refers to a host of techniques aimed at delivering (apparently) different content to search engines than to human users. Cloaking is typically used in conjunction with content spam, by serving a page containing popular query terms to the search engine (thereby increasing the likelihood that the page will be returned as the result of a search), and presenting the human user with a different page. Cloaking can be achieved using many different techniques: by literally serving different content to search engines than to ordinary users (based for example on the well-known IP addresses of the major search engine crawlers), by rendering certain parts of the page invisible (say by setting the font to the same color as the background), by using client-side scripting to rewrite the page after it has been delivered (relying on the observation that search engine crawlers typically do not execute scripts), and finally by serving a page that immediately redirects the user's browser to a different page (either via client-side scripting or the HTML "meta-redirect" tag). Each variant of cloaking calls for a different defense. Search engines can guard against different versions of the same page by probing the page from unaffiliated IP addresses [13]; they can detect invisible content by rendering the page; and they can detect page modifications and script-driven redirections by executing client-side scripts [12].

Key Applications

Web spam detection is used primarily by advertisement-financed general-purpose consumer search engines. Web spam is not an issue for enterprise search engines, where the content providers, the search engine operator and the users are all part of the same organization and have shared goals. However, web spam is bound to become a problem in any setting where these three parties – content providers, searchers, and search engines – have different objectives. Examples of such settings include vertical search services, such as product search engines, company search engines, people search engines, or even scholarly literature search engines. Many of the basic concepts described above are applicable to these domains as well; the precise set of features useful for spam detection will depend on the ranking algorithms used by these vertical search engines.

Future Directions

Search engines are increasingly leveraging human intelligence, namely the observable actions of their

user base, in their relevance assessments; examples include click-stream analysis, toolbar data analysis, and analysis of traffic on affiliate networks (such as the Google AdSense network). It is likely that many of the future spam detection features will also be based on the behavior of the user base. In many respects, the distinction between computing features for ranking (promoting relevant documents) and spam detection (demoting facetious documents) is artificial, and the boundary between ranking and spam suppression is likely to blur as search engines evolve.

Experimental Results

Several studies have assessed the incidence of spam in large-scale web crawls at between 8% and 13% [11,5]; the percentage increases as more pages are crawled, since many spam sites serve a literally unbounded number of pages, and web crawlers tend to crawl high-quality human-authored content early on. Ntoulas et al. describe a set of content-based features for spam detection; these features, when combined using a decision-tree-based classifier, resulted in an overall spam prediction accuracy of 97% [11].

Data Sets

Castillo et al. have compiled the WEBSPAM-UK2006 data set [2], a collection of web pages annotated by human judges as to whether or not they are spam. This data set has become a reference collection to the field, and has been used to evaluate many of the more recent web spam detection techniques.

Cross-references

- ▶ Indexing the Web
- ▶ Web Page Quality Metrics
- ▶ Web Search Relevance Feedback
- ▶ Web Search Relevance Ranking

Recommended Reading

1. Becchetti L., Castillo C., Donato D., Leonardi S., and Baeza-Yates R. Using rank propagation and probabilistic counting for link-based spam detection. In Proc. KDD Workshop on Web Mining and Web Usage Analysis, 2006.
2. Castillo C., Donato D., Becchetti L., Boldi P., Leonardi S., Santini M., and Vigna S. A reference collection for Web spam. ACM SIGIR Forum, 40(2):11–24, 2006.
3. Daswani N. and Stoppelman M. and the Google Click Quality and Security Teams. The anatomy of clickbot.A. In Proc. 1st Workshop on Hot Topics in Understanding Botnets, 2007.
4. Davison B.D. Recognizing nepotistic links on the web. In Proc. AAAI Workshop on Artificial Intelligence for Web Search, 2000.
5. Fetterly D., Manasse M., and Najork M. Spam, damn spam and statistics. In Proc. 7th Int. Workshop on the Web and Databases, 2004, pp. 1–6.
6. Gyöngyi Z., and Garcia-Molina H. Spam: its not just for Inboxes anymore. IEEE Comput., 38(10):28–34, 2005.
7. Gyöngyi Z. and Garcia-Molina H. Web Spam Taxonomy. In Proc. 1st Int. Workshop on Adversarial Information Retrieval on the Web, 2005, pp. 39–47.
8. Gyöngyi Z., Garcia-Molina H., and Pedersen J. Combating Web spam with TrustRank. In Proc. 30th Int. Conf. on Very Large Data Bases, 2004, pp. 576–587.
9. Henzinger M., Motwani R., and Silverstein C. Challenges in web search engines. ACM SIGIR Forum 36(2):11–22, 2002.
10. Mishne G., Carmel D., and Lempel R. Blocking blog spam with language model disagreement. In Proc. 1st Int. Workshop on Adversarial Information Retrieval on the Web, 2005, pp. 1–6.
11. Ntoulas A., Najork M., Manasse M., and Fetterly D. Detecting spam web pages through content analysis. In Proc. 15th Int. World Wide Web Conference, 2006, pp. 83–92.
12. Wang Y.M., Ma M., Niu Y., and Chen H. Spam double-funnel: connecting Web spammers with advertisers. In Proc. 16th Int. World Wide Web Conference, 2007, pp. 291–300.
13. Wu B. and Davison B. Detecting semantic cloaking on the web. In Proc. 15th Int. World Wide Web Conference, 2006, pp. 819–828.

Web Structure Mining

▶ Data, Text, and Web Mining in Healthcare

Web Transactions

Heiko Schuldt
University of Basel, Basel, Switzerland

Synonyms

Internet transactions

Definition

A *Web Transaction* is a transactional interaction between a client, usually a web browser, and one or several databases as backend of a multi-tier architecture. The middle tier of the architecture includes a web server which accepts client requests via HTTP. It forwards these requests either directly to the underlying database or to an application server which, in turn, interacts with the database.

Key Points

The main application of Web transactions is in eCommerce applications. In a minimal configuration, the architecture for Web transactions consists of a client, a web server and a database server. Database access is provided at the web server level, i.e., by embedding database access into Java servlets or JavaServer Pages (JSP). More sophisticated architectures consider a dedicated application server layer (i.e., an implementation of the Java Enterprise Edition specification Java EE) and support distributed databases. Communication between application and database layer is usually implemented on the basis of JDBC (Java Database Connectivity), ODBC (Open Database Connectivity), or native database protocols. Transaction support at application server level is provided by services like JTS (Java Transaction Service) via the Java Transaction API (JTA) in the Java world, or OTS (Object Transaction Service) in CORBA. Essentially, these services allow to associate several application server calls and their interactions with resource managers with the same transaction context and to coordinate them by using a two phase commit (2PC) protocol. Thus, Web transactions mostly focus on atomic commit processing. Similarly, protocols on the Web service stack exploit 2PC to atomically execute several Web services.

Multi-tier web applications require special support for failure handling at application level (application recovery). In order to increase the degree of scalability of multi-tier architectures for Web transactions, application servers can be replicated. To avoid that the database backend then becomes a bottleneck, dedicated caching strategies at the middle tier are applied. Web transactions can be part of Transactional processes.

Cross-references

- ▶ Application Server
- ▶ Caching
- ▶ Transactional Processes
- ▶ Web Service

Recommended Reading

1. Barga R., Lomet D., Shegalov G., and Weikum G. Recovery Guarantees for Internet Applications. ACM Trans. Internet Technol., 4(3):289–328, 2004.
2. Burke R. and Monson-Haefel R. Enterprise JavaBeans 3.0. O'Reilly, 2006.
3. Luo Q., Krishnamurthy S., Mohan C., Pirahesh H., Woo H., Lindsay B., and Naughton J. Middle-tier database caching for e-business. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 2002, pp. 600 –611.

Web Usage Mining

▶ Data, Text, and Web Mining in Healthcare

Web Views

Alexandros Labrinidis
Department of Computer Science, University of Pittsburgh, Pittsburgh, PA, USA

Synonyms

Web Views; HTML fragment

Definition

Web Views are web pages or web page fragments that are automatically created from base data, which are typically stored in a database management system (DBMS).

Key Points

Although caching of HTML pages (and fragments) has been proposed in the literature as early as the mid-1990s, the term Web View was introduced in 1999 [2] to denote that these fragments are generated through queries made to a back-end DBMS.

The concept of a *Web View* facilitates the materialization of dynamically generated HTML fragments outside the DBMS. The main advantage of materializing Web Views (outside the DBMS, e.g., at the web server) is that the web server need not query the DBMS for every user request, thus greatly improving query response time for the user [3]. On the other hand, for the quality of the data returned to the user to be high, the system must keep materialized Web Views fresh (in the background). This essentially decouples the processing of queries at the web server from the processing of updates at the DBMS (which are also propagated to materialized Web Views).

Materializing a Web View presents a trade-off. On the one hand, it can greatly improve response time for user-submitted queries. On the other hand, it generates a background overhead for the Web View to be kept fresh (essentially a materialization decision can be

viewed as a "*contract*" for the Web View to be kept fresh). As such, selecting which Web View to materialize is an important problem that has received attention [5,6]. This problem is essentially similar to the view selection problem in traditional DBMSs [5], with added complexity due to the online nature of the Web and the need for any solution to be highly dynamic, constantly adapting to changing conditions and workloads.

Having identified the set of Web Views to materialize, there is the issue of determining the order by which to propagate updates to them. Since one update to a base relation can trigger the refresh of multiple different materialized Web Views, the order by which these are updated can make an impact on the overall quality of data returned to the user. This is essentially a special-case online scheduling problem, which has been addressed in [4].

Web Views have been used successfully to decouple the processing of queries (to generate dynamic, database-driven web pages) from that of updates (to the content stored inside the DBMS used to driven the web site). This enables much better performance (in terms of response to user queries) without sacrificing the freshness of the data served back to the user.

Cross-references

► View Maintenance

Recommended Reading

1. Gupta H. and Mumick I.S. Selection of views to materialize under a maintenance cost constraint. In Proc. 7th Int. Conf. on Database Theory, 1999, pp. 453–470.
2. Labrinidis A. and Roussopoulos N. Alexandros On the materialization of Web views. In Proc. ACM SIGMOD Workshop on The Web and Databases, 1999, pp. 79–84.
3. Labrinidis A. and Roussopoulo N. Web View materialization. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 2000, pp. 367–378.
4. Labrinidis A. and Roussopoulos N. Update propagation strategies for improving the quality of data on the Web. In Proc. 27th Int. Conf. on Very Large Data Bases, 2001, pp. 391–400.
5. Labrinidis A. and Roussopoulos N. Balancing performance and data freshness in Web database servers. In Proc. 29th Int. Conf. on Very Large Data Bases, 2003, pp. 393–404.
6. Labrinidis A. and Roussopoulos N. Exploring the tradeoff between performance and data freshness in database-driven Web servers. VLDB J., 13(3):240–255, 2004.

Web Widget

► Snippet

What-If Analysis

Stefano Rizzi
University of Bologna, Bologna, Italy

Definition

In order to be able to evaluate beforehand the impact of a strategic or tactical move so as to plan optimal strategies to reach their goals, decision makers need reliable predictive systems. What-if analysis is a data-intensive simulation whose goal is to inspect the behavior of a complex system, such as the corporate business or a part of it, under some given hypotheses called scenarios. In particular, what-if analysis measures how changes in a set of independent variables impact a set of dependent variables with reference to a given simulation model such a model is a simplified representation of the business, tuned according to the historical corporate data. In practice, formulating a scenario enables the building of a hypothetical world that the analyst can then query and navigate.

Historical Background

Though what-if analysis can be considered as a relatively recent discipline, its background is rooted at the confluence of different research areas, some of which date back decades ago.

First of all, what-if analysis lends some of the techniques developed within the simulation community, to contextualize them for *business intelligence*. Simulations are used in a wide variety of practical contexts, including physics, chemistry, biology, engineering, economics, and psychology. Much literature has been written in this field over the years, mainly regarding the design of simulation experiments and the validation of simulation models [5,8,9].

Another relevant field for what-if analysis is economics that provide the insights into business processes necessary to build and test simulation models. For instance, in [1] a set of alternative approaches to forecasting are surveyed, and useful guidelines for selecting the best ones according to the availability and reliability of knowledge are given.

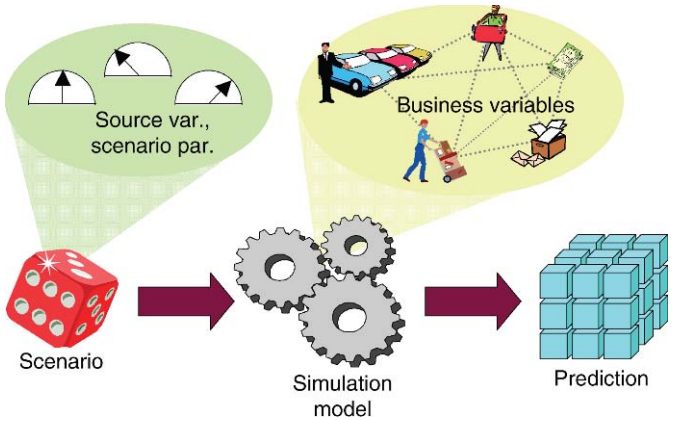
Finally, what-if analysis relies heavily on database and *data warehouse* technology. Though data warehousing systems have been playing a leading role in supporting the decision process over the last decade, they are aimed at supporting analysis of past data

("what-was") rather than giving conditional anticipations of future trends ("what-if"). Nevertheless, the historical data used to reliably build what-if predictions are normally taken from the enterprise data warehouse. Besides, there is a tight relationship between what-if analysis and *multidimensional modeling* since input and output data for what-if analysis are typically stored within *cubes* [7]. In particular, in [2] the SESAME system for formulating and efficiently evaluating what-if queries on data warehouses is presented. Here, scenarios are defined as ordered sets of hypothetical modifications on multidimensional data. Finally, there are relevant similarities between simulation modeling for what-if analysis and the modeling of *Extraction, Transformation and Loading* applications; in fact, both ETL and what-if analysis can both be seen as a combination of elementary processes each transforming an input data flow into an output.

Foundations

As sketched in Fig. 1, a what-if application is centered on a *simulation model*, that establishes a set of complex relationships between some *business variables* corresponding to significant entities in the business domain (e.g., products, branches, customers, costs, revenues, etc.). A simulation model supports one or more *scenarios*, each describing one or more alternative ways to construct a *prediction* of interest for the user. The prediction typically takes the form of a multidimensional *cube*, whose *dimensions* and *measures* correspond to business variables, to be interactively explored by the user by means of any On-Line Analytical Processing (OLAP) front-end. A scenario is characterized by a subset of business variables, called *source variables*, and by a set of additional parameters, called *scenario parameters*, that the user has to value in order to execute the model and obtain the prediction. While business variables are related to the business domain, scenario parameters convey information technically related to the simulation, such as the type of regression adopted for forecasting and the number of past years to be considered for regression. Distinguishing source variables among business variables is important since it enables the user to understand which are the "levers" that she can independently adjust to drive the simulation. Each scenario may give rise to different simulations, one for each assignment of the source variables and of the scenario parameters.

A simple example of a what-if query in the marketing domain is: *How would my profits change if I run a 3 × 2 (pay 2 and take 3) promotion for one week on all audio products on sale?* Answering this query requires building a simulation model capable of expressing the complex relationships between the business variables that determine the impact of promotions on product sales, and to run it against the historical sale data in order to determine a reliable forecast for future sales. In particular, the source variables for this scenario are the type of promotion, its duration, and the product category it is applied to. Possible scenario parameters could be the type of regression used for forecasting and the number of past years to be considered for regression. The specific simulation expressed by the what-if query reported in the text is determined by giving values "3 × 2," "one week" and "audio," respectively,



What-If Analysis. Figure 1. Functional sketch for what-if analysis.

to the three source variables. The prediction could be a *cube* with dimensions week and product and measures revenue, cost and profit.

Importantly, what-if analysis should not be confused with *sensitivity analysis*, aimed at evaluating how sensitive the behavior of the system is to a small change of one or more parameters. Besides, there is an important difference between what-if analysis and simple *forecasting*, widely used especially in the banking and insurance fields. In fact, while forecasting is normally carried out by extrapolating trends out of the historical series stored in information systems, what-if analysis requires simulating complex phenomena whose effects cannot be simply determined as a projection of past data.

On the other hand, applying forecasting techniques is often required during what-if analysis. In [4] the authors report a useful classification of forecasting methods into *judgmental*, such as those based on expert opinions and role-playing, and *statistical*, such as extrapolation methods, expert systems and rule-based forecasting. The applicability of these methods to different domains is discussed, and an algorithm for selecting the best method depending on the specific characteristics of the problem at hand is reported.

A separate mention is in order for *system dynamics* [4,11], an approach to modeling the behavior of nonlinear systems, in which cause-effect relationships between abstract events are captured as dependencies among numerical variables; in general, such dependencies can give rise to retroactive interaction cycles, i.e., feedback loops. From a mathematical standpoint, systems of differential equations are the proper tool for modeling such systems. In the general case, however, a solution cannot always be found analytically, so numerical techniques are often used to predict the behavior of the system. A system dynamics model consists of a set of variables linked together, classified as *stock* and *flow* variables; flow variables represent the rate at which the level of cumulation in stock variables changes. By running simulations on such a model, the user can understand how the system will evolve over time as a consequence of a hypothetical action she takes. She can also observe, at each time step, the values assumed by the model variables and (possibly) modify them. Thus, it appears that system dynamics can effectively support what-if applications in which the current state of any part of the system could influence its own future state through a closed chain of dependency links.

Designing a what-if application requires a methodological framework; the one presented in [6] relies on seven phases:

1. *Goal analysis*, aimed at determining which business phenomena are to be simulated, and how they will be characterized. The goals are expressed by (i) identifying the set of business variables the user wants to monitor and their granularity; and (ii) defining the relevant scenarios in terms of source variables the user wants to control.
2. *Business modeling*, which builds a simplified model of the application domain in order to help the designer to understand the business phenomenon as well as give her some preliminary indications about which aspects can be either neglected or simplified for simulation.
3. *Data source analysis*, aimed at understanding what information is available to drive the simulation and how it is structured.
4. *Multidimensional modeling*, which defines the multidimensional schema describing the prediction by taking into account the static part of the business model produced at phase 2 and respecting the requirements expressed at phase 1.
5. *Simulation modeling*, whose aim is to define, based on the business model, the simulation model allowing the prediction to be constructed, for each given scenario, from the source data available.
6. *Data design and implementation*, during which the multidimensional schema of the prediction and the simulation model are implemented on the chosen platform, to create a prototype for testing.
7. *Validation*, aimed at evaluating, together with the users, how faithful the simulation model is to the real business model and how reliable the prediction is. If the approximation introduced by the simulation model is considered to be unacceptable, phases 4–7 should be iterated to produce a new prototype.

The three modeling phases require a supporting formalism. Standard UML can be used for phase 2 (e.g., a use case diagram and a class diagram coupled with activity diagrams) and any formalism for conceptual modeling of multidimensional databases can be effectively adopted for phase 4. Finding a suitable formalism to give broad conceptual support to phase 5 is much harder, though some examples based on the use of colored Petri nets, event graphs and flow charts can be found in the simulation literature [10].

Key Applications

Among the killer applications for what-if analysis, it is worth mentioning profitability analysis in commerce, hazard analysis in finance, promotion analysis in marketing, and effectiveness analysis in production planning. Less traditional, yet interesting applications described in the literature are urban and regional planning supported by *spatial databases*, index selection in relational databases, and ETL maintenance in data warehousing systems.

Either spreadsheets or OLAP tools are often used to support what-if analysis. Spreadsheets offer an interactive and flexible environment for specifying scenarios, but lack seamless integration with the bulk of historical data. Conversely, OLAP tools lack the analytical capabilities of spreadsheets and are not optimized for scenario evaluation [2]. Recently, what-if analysis has been gaining wide attention from vendors of business intelligence tools. For instance, both SAP SEM (Strategic Enterprise Management) and SAS Forecast Server already enable users to make assumptions on the enterprise state or future behavior, as well as to analyze the effects of such assumptions by relying on a wide set of forecasting models. Also Microsoft Analysis Services provides some limited support for what-if analysis. This is now encouraging companies to integrate and finalize their business intelligence platforms by developing what-if applications for building reliable business predictions.

Future Directions

Surprisingly, though a few commercial tools are already capable of performing forecasting and what-if analysis, and some papers describe relevant applications in different fields, very few attempts have been made so far to address methodological and modeling issues in this field (e.g., [6]). On the other hand, facing a what-if project without the support of a design methodology is very time-consuming, and does not adequately protect the designer and his customers against the risk of failure. The main problem related to the design of what-if applications is to find an adequate formalism to conceptually express the simulation model, so that it can be discussed and agreed upon with the users. Unfortunately, no suggestion to this end is given in the literature, and commercial tools do not offer any general modeling support. Another relevant problem is to establish a general framework for estimating the loss of precision that is introduced when modeling low-level phenomena with higher-level dependencies. This could allow designers to assess the reliability of the prediction as a function of the quality of the historical data sources and of the precision of the simulation model.

Decision makers are used to navigating multidimensional data within OLAP sessions, that consist in the sequential application of simple and intuitive OLAP operators, each transforming a cube into another one. Consequently, it is natural for them to ask for extending this paradigm for accessing information also to what-if analysis. This would allow users to mix together navigation of historical data and simulation of future data into a single session of analysis. In the same direction, an approach has recently been proposed for integrating OLAP with data mining [3]. This raises an interesting research issue. In fact, OLAP should be extended with a set of new, well-formed operators specifically devised for what-if analysis. An example of such operator could be *apportion*, which disaggregates a quantitative information down a hierarchy according to some given criterion (*driver*). For instance, a transportation cost forecasted by branch and month could be apportioned by product type proportionally to the quantity shipped for each product type. In addition, efficient techniques for supporting the execution of such operators should be investigated.

Cross-references

- ▶ Business Intelligence
- ▶ Cube
- ▶ Data Warehousing Systems: Foundations and Architectures
- ▶ On-Line Analytical Processing

Recommended Reading

1. Armstrong S. and Brodie R. Forecasting for marketing. In Quantitative methods in marketing. G. Hooley and M. Hussey (eds.). Int. Thompson Business Press, London, 1999, pp. 92–119.
2. Balmin A., Papadimitriou T., and Papakonstantinou Y. Hypothetical Queries in an OLAP Environment. In Proc. 26th Int. Conf. on Very Large Data Bases, 2000, pp. 220–231.
3. Chen B., Chen L., Lin Y., and Ramakrishnan, R. Prediction cubes. In Proc. 31st Int. Conf. on Very Large Data Bases, 2005, pp. 982–993.
4. Coyle R.G. System Dynamics Modeling: A Practical Approach. Chapman and Hall, London, 1996.
5. Fossett C., Harrison D., and Weintrob H. An assessment procedure for simulation models: a case study. Oper. Res., 39(5): 710–723, 1991.

6. Golfarelli M., Rizzi S., and Proli A. Designing what-if analysis: towards a methodology. In Proc. ACM 9th Int. Workshop on Data Warehousing and OLAP, 2006, pp. 51–58.
7. Koutsoukis N.S., Mitra G., and Lucas C. Adapting on-line analytical processing for decision modeling: the interaction of information and decision technologies. Decis. Support Syst., 26(1):1–30, 1999.
8. Kreutzer W. System Simulation – Programming Styles and Languages. Addison Wesley, Reading, MA, 1986.
9. Law A.M. and Kelton W.D. Simulation Modeling and Analysis. McGraw-Hill Higher Education, Boston, MA, 1999.
10. Lee C., Huang H.C., Liu B., and Xu Z. Development of timed colour petri net simulation models for air cargo terminal operations. Comput. Ind. Eng., 51(1):102–110, 2006.
11. Roberts E.B. Managerial applications of system dynamics. Pegasus Communications, 1999.

While Loop

► Loop

Wide-Area Data Replication

► WAN Data Replication

Wide-Area Storage Systems

► Peer-to-Peer Storage

WIMP Interfaces

Stephen Kimani
CSIRO Tasmanian ICT Centre, Hobart, TAS, Australia

Definition

There exist many types of interaction styles. They include but are not limited to: command line interface, natural language, question/answer and query dialog, form-fills and spreadsheets, WIMP, and three-dimensional interfaces. The most common of the foregoing interaction styles is the WIMP. WIMP is an acronym for Windows, Icons, Menus and Pointers. Alternatively, it is an acronym for Windows, Icons, Mice and Pull-down menus. Examples of user interfaces that are based on the WIMP interaction style include: Microsoft Windows for PCs, MacOs for Apple Macintosh, various X Windows-based systems for UNIX, etc.

Historical Background

WIMP interfaces were invented at the SRI laboratory in California. The development of WIMP interfaces continued at Xerox PARC. The 1981 Xerox Star workstation is considered to be the first production computer to use the desktop metaphor, productivity applications and a three-button mouse. WIMP was popularized by the Apple Macintosh in the early 1980s. The interaction style/paradigm has now been copied by the Microsoft Windows operating system, Motif, the X Window System, etc. The rapid rise of Microsoft Windows has made WIMP interfaces become the dominant interface paradigm. WIMP interfaces have been improved with a set of new user interface widgets during the years. However, the basic structure of a WIMP interface usually does not change. WIMP interfaces typically present the work space using a desktop metaphor [5]. Everything is presented in a two dimensional space which has windows. The functionality of the application is made available through interface widgets such as: menus, dialog boxes, toolbars, palettes, buttons, etc.

Foundations

In this section is a description of the elements of the WIMP interface.

Windows

Windows are areas of the display that behave as if they were independent terminals. They are typically rectangular areas of the display that can be manipulated independently on the display screen. In most cases, the view of the contents of a window can be changed by scrolling or editing. Windows can contain text and/or graphics. They can be moved, resized, closed, maximized, or minimized (reduced to an icon). Many windows can be displayed on the screen simultaneously thereby allowing multiple separated tasks to be visible at the same time. The user can switch from one from one task to another by moving from one window to another.

The components of windows usually include:

- Scrollbars: They enable users to move the contents of the window up and down (vertical scrollbar), or from side to side (horizontal scrollbar).

- Title bars: They describe the name of the window.
- Status bar: It displays the status of the task/process in the window.
- Boxes/widgets for resizing and closing the window.

Oftentimes, all windows in use will not fit on the screen space at once. Several strategies exist for managing multiple windows:

1. Iconification/minimizing: This strategy allows screen space to be saved by reducing windows to window icons. The user can re-expand the icons at will. A window icon therefore serves as a visual reminder of the window.
2. Tiling: In this case, the system uses all the available screen space to display the windows. The windows do not overlap. Tiling can take many forms. For instance: some systems use a fixed number of tiles while others allow variable numbers of tiles.
3. Overlapping: This strategy is probably now the most popular. In the strategy, windows are allowed to partially obscure each other like overlapping papers arranged on a desk. Cascading is a form of overlapping where the windows are automatically arranged fanned out, usually in a diagonal line so that the title and one other border of each window can be seen. With cascading, many windows can therefore be displayed in a limited screen space at the same time (Figs. 1 and 2).

When multiple windows are displayed on the screen at the same time, the active window is usually distinguished by a shaded/bold title bar. The active window is said to have focus. Windows have three size states: maximized, minimized/iconified and normal/restored. Some systems support windows within windows e.g., in Microsoft PowerPoint (MS Office 2000).

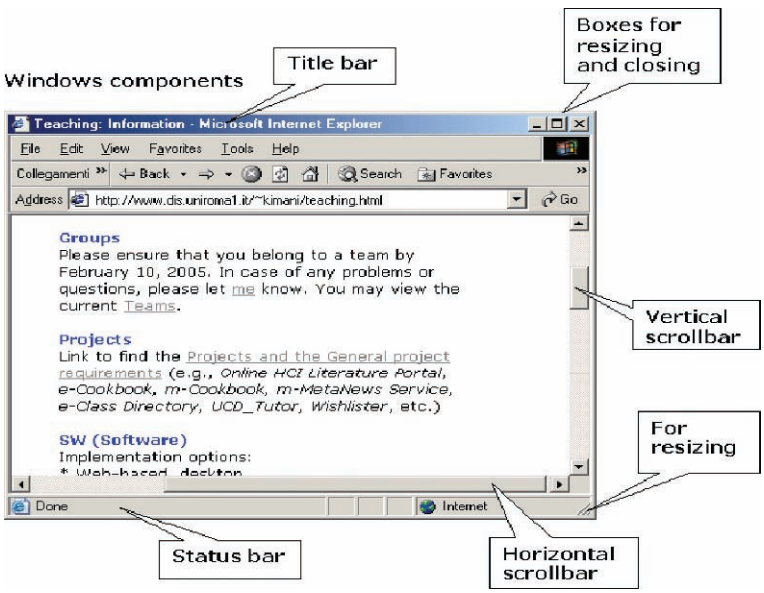
Icons

In the context of WIMP, an icon is a small picture or image, used to represent some aspect of the system (such as a printer icon to represent the printing action) or to represent some entity/object (such as a window) (Fig. 3).

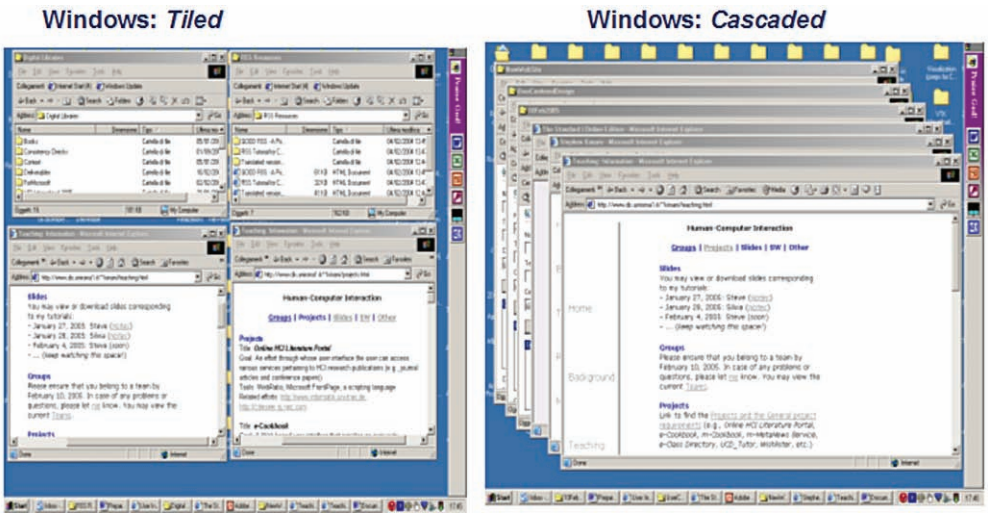
As it was indicated earlier, a window may among other things be closed completely. Alternatively it may be reduced to this small representation (iconified/minimized). An icon therefore can save screen space and therefore many windows can be available on the screen simultaneously. Moreover, it can serve as a reminder to the user that s/he can resume dialog (or interaction with the represented application) by simply opening up the window.

Pointers

A pointer is a cursor on the display screen. It is worth noting that interaction in WIMP interfaces relies a lot on pointing and selecting interaction objects such as



WIMP Interfaces. Figure 1. Common components of a window.



WIMP Interfaces. Figure 2. Tiling versus cascading.

WIMP Interfaces. Figure 3. Examples of icons.

icons and menu items. Pointers are therefore important in WIMP interfaces. There are at least two types of cursors: mouse cursor and text cursor. A mouse cursor shows the user where the current position of the mouse is considered to be with respect to the windows on screen. Usually, the shape and behavior of the mouse cursor can be changed. A text cursor shows where input will be directed from the keyboard. The mouse is the most common device for pointing and clicking. However, other input devices such as joystick, trackball, cursor keys or keyboard shortcuts too can be used for such pointing and selecting tasks. In a particular system, different cursors are used to represent different modes/states e.g., normal cursor as an arrow, double-arrowed cursor for resizing windows, hour-glass when system is busy, etc. A hot-spot is the location to which the cursor points. Selection occurs at the coordinate of the hot-spot. The hot-spot of the cursor should be obvious to the user (Fig. 4).

Menus

A menu is an interaction feature that presents a set of options displayed on the screen where the selection and execution of one (or more) of the options results in a change in the state of the interface [8]. The human beings' ability to recognize information on being given a visual cue is superior to their ability to recall the information. Menus therefore can serve as cues for the operations or services that the system can perform. Therefore, the labels/names used for the commands in menus should be informative and meaningful. A potential candidate for selection can be indicated by using a pointing device (e.g., mouse, arrow keys, etc) to move the pointer accordingly. As the pointer moves, visual feedback is typically given by some kind of visual emphasis such as highlighting the menu item. Selection of a menu item can be realized by some additional user action (such as pressing a button on the pointing devices, pressing some key on the keyboard, etc). Keyboard accelerators, which are key combinations that have the same effect as selecting the menu item, are sometimes offered. When there are too many items, the menu items are often grouped and layered. The main menu, usually represented as a menu bar, should be conspicuous or readily available/accessible. Menu bars are often placed at one of the sides of the screen or window. For instance: at the top of the screen (e.g., MacOS), at the top of each window (e.g., Microsoft Windows). Most systems show the currently selected state of any group of menu items by displaying them in bold or by a tick. Entries that are disallowed in the current context are often shown in a dimmed font.

Types of menus:

- Pull-down menus: They are attached to a main menu under the title bar or to buttons. Pull-down menus are dragged from the main menu by

WIMP Interfaces. Figure 4. Examples of pointers.

moving the pointer into the menu bar and pressing the button.
- Fall-down menus: They automatically appear from the main menu when the pointer enters the menu bar, without having to press the button.
- Pop-up menus: They appear when a particular region of the screen or window, maybe designated by an icon, is selected, but they remain in position until the user instructs it to disappear again e.g., by clicking on a "close box" in the border of the menu's window, by releasing the mouse button, etc.
- Pin-up menus: They can be "pinned" or attached to the screen, staying in place until explicitly asked to go away.
- Pie menus: The menu options in a pie menu have a circular arrangement with the pointer at the center.

There are two main challenges with menus: deciding which items to include and how to group those items. Here are some guidelines on menu design:

- A menu label in pull-down menus should reflect the functions of the underlying menu items.
- The items in pull-down menus should be grouped by function.
- Menu groupings in pull-down menus should be consistent (to facilitate the transfer of learning and confidence to the user).
- Menu items should be ordered in the menu according to importance and frequency of use.
- Opposite functionalities (e.g., "save" and "delete") should be kept apart to prevent accidental selection of the wrong function.

Additional Interaction Elements

Many other interaction elements may be present in a WIMP interface. For instance: buttons, sliders, toolbars, palettes, dialog boxes, etc. Buttons are individual and isolated regions within a display that can be selected by the user to invoke a specific operation or state. Sliders are used to set quantities that vary continuously within given limits or even to enable the user to choose between large numbers of options [9]. A toolbar is a collection of small buttons, each with icons, that provides convenient access to commonly used functions. It should be pointed out that although the function of a toolbar is similar to that of a menu bar, the icons in a toolbar are smaller than the equivalent text and therefore more functions can be simultaneously displayed. A palette is a mechanism for making the set of possible modes and the active/current mode visible to the user. A dialog box is an information window used by the system to provide contextual information.

Key Applications

WIMP interfaces have made computer usage more accessible to users who previously could not use the earlier types of user interfaces. Such users include: young children (who cannot yet read or write), managers, and non-professional home users [11]. WIMP interfaces can be credited for the increased emphasis on the incorporation of user interface design and usability evaluation in the software development process in the late 1980s and early 1990s. On the whole, WIMP interfaces can be said to be instrumental in the realization of applications that are characterized by relative ease of learning, ease of use, and ease of transfer of knowledge due to the consistency in WIMP-based designs [11].

Cross-references

- ▶ Human-Computer Interaction
- ▶ Icon
- ▶ Visual Interaction
- ▶ Visual Interfaces
- ▶ Visual Metaphor

Recommended Reading

1. Alistair E. The design of auditory interfaces for visually disabled users. In Proc. SIGCHI Conf. on Human Factors in Computing Systems, 1988, pp. 83–88.
2. Balakrishnan R. and Kurtenbach G. Exploring bimanual camera control and object manipulation in 3D graphics interfaces.

In Proc. SIGCHI Conf. on Human Factors in Computing Systems, 1999, pp. 56–62.
3. Beaudouin-Lafon M. Designing interaction, not interfaces. In Proc. Working Conf. on Advanced Visual Interfaces, 2004, pp. 15–22.
4. Beaudouin-Lafon M. and Lassen H.M. The architecture and implementation of CPN2000, a post-WIMP graphical application. In Proc. 13th Annual ACM Symp. on User Interface Software and Technology, 2000, pp. 181–190.
5. Cesar P. Tools for adaptive and post-WIMP user interfaces. New Directions on Human Computer Interaction, 2005.
6. Dix A., Finlay J., Abowd G., and Beale R. Human-Computer Interaction. Prentice Hall, Englewood Cliffs, NJ, 2003.
7. Green M. and Jacob R. SIGGRAPH: '90 Workshop report: software architectures and metaphors for non-WIMP user interfaces. Comput. Graph., 25(3): 229–235, 1991.
8. Paap K.R. and Roske-Hofstrand R.J. Design of menus. In Handbook of Human-Computer Interaction, M. Helander (ed.). Amsterdam, North-Holland, 1998.
9. Preece J., Rogers Y., Sharp H., Benyon D., Holland S., and Carey T. Human-Computer Interaction. Addison-Wesley, Reading, MA, 1994.
10. Odell D.L., Davis R.C., Smith A., and Wright P.K. Toolglasses, marking menus, and hotkeys: a comparison of one and two-handed command selection techniques. In Proc. 2004 Conf. on Graphics Interface, 2004, pp. 17–24.
11. van Dam A. Post-WIMP user interfaces. Commun. ACM, 40(2):63–67, 1997.

Window-based Query Processing

Walid G. Aref
Purdue University, West Lafayette, IN, USA

Synonyms

Stream query processing

Definition

Data Streams are infinite in nature. As a result, a query that executes over data streams specifies a "window" of focus or the part of the data stream that is of interest to the query. When new data items arrive into the data stream, the window may either expand or slide to allow the query to process these new data items. Hence, queries over data streams are continuous in nature, i.e., the query is continuously re-evaluated each time the query window slides. Window-based query processing on data streams refers to the various ways and techniques for processing and evaluating continuous queries over windows of data stream items.

Historical Background

Windows over relational tables have already been introduced into Standard SQL (SQL:1999) in order to support data analysis, decision support, and more generally, OLAP-type operations.

However, the motivation for having windows in data stream management systems is quite different. Since data streams are infinite, it is vital to limit and focus the scope of a query to a manageable and finite portion of the data stream. Earlier works on window query processing have focused on processing tuple-count and time-sliding windows. Ever since, window query processing techniques have been developed to deal with out-of-order tuple arrivals, e.g., [14], revision tuple processing, e.g., [1,13], incremental evaluation techniques, e.g., [6], stream punctuation techniques, e.g., [17], multi-query optimization techniques, e.g., shared execution of window queries [4,7,9,10], and adaptive stream query processing techniques, e.g., [12].

Foundations

Queries over data streams are continuous in nature. A continuous query progressively produces results as new data items arrive into the data stream. Since data streams are infinite, queries that execute over a data stream need to define a region of interest (termed a window). There are several ways by which a query can define its window(s) of interest. These include stream-based versus operation-based windows, tuple-count versus time-sliding windows, and sliding versus predicate windows. These ways for specifying windows are orthogonal and can hence be combined. For example, a window can be time-sliding and at the same time, operation-based. Similarly, a stream-based window can also be predicate-based, etc.

Query Processing Techniques: Incremental Evaluation Versus Reevaluation

Whenever the data items within a window change, e.g., when the window slides with the arrival of new tuples or with the expiration of some old tuples from the window, a continuous query's answer needs to be reproduced. There are two mechanisms for reproducing the answer to a continuous query, namely (i) query reevaluation and (ii) incremental evaluation, which are explained in the following sections.

Query Reevaluation

In query reevaluation, a window over a data stream is viewed as an instantiation of a table that contains the data tuples in the window. When the window slides, a new table is formed (or opened). So, from the point of view of a continuous query, a data stream is a sequence of tables. With the arrival of a new table (window of data stream items), the query is reevaluated to produce a new output result. As a consequence, an important feature of query reevaluation is that the semantics of the traditional query processing operators, e.g., selects and joins, do not change. When the slide of a window is less than its range, multiple overlapping windows will be open concurrently over the same stream. Hence, it is possible that a newly arriving tuple contribute to multiple windows.

Query Processing using Revision Tuples At times, data stream items may be noisy or erroneous due to the nature of the data streaming applications. As a result, stream data sources may need to "revise" previously issued tuples in the form of "revision tuples" [1,13]. So, in a sense, processing a revised tuple may involve reprocessing of the input tuples that were in the same window or computation as the tuple being revised to produce revised output results. Consequently, some query operators will need to preserve "state" in order to reprocess the revised input tuples. For example, aggregates may need to store the individual values (or some summary of these values) that were used to produce the aggregate result so that the aggregate operator may be recomputed given the revised input tuple. This approach is referred to as "upstream processing." It is possible to go downstream, i.e., from the output tuples backwards. The idea is to correct the previously generated output tuples using the new and original values of the revised tuple as well as some state information depending on the nature of the participating query operators. More detail about processing revision tuples can be found in [1,13].

Query Processing using Punctuations Some stream query operators are stateful while others are blocking. Since data streams are infinite, the state of these query operators may be unbounded in size. Similarly, blocking operators cannot wait indefinitely as the data streams never end. Knowing more a priori knowledge about the stream semantics, it is possible to embed, within the data stream, special annotations, termed "Punctuations" that break the infinite data stream into finite sub-streams. Then, stateful and blocking query operators can be applied successfully to each of the substreams.

Stream punctuations can take multiple forms. For example, one stream can send a "last-reading-within-this-hour" punctuation to reflect that no more readings for this hour are expected from that source. In this case, a blocking operator can produce results related to this source once the operator receives this punctuation. Similarly, in a bidding application, sending a "no-more-bids-for-item:%itemid" punctuation would allow the bid to be finalized for that item. In general, a punctuation can be viewed as a predicate that evaluates to "false" for every tuple in the stream that follows the punctuation. This helps query operators generate output tuples following a punctuation as well as help reduce the size of the state kept per operator. Detailed discussion about query processing using punctuations can be found in [17].

Query Processing Using Heartbeats Stream data sources often assign a timestamp to each data element they produce. Due to the distributed nature of the stream data sources, data elements may arrive to the data stream management system out of order. System buffers need to store the out-of-order tuples to present them later to the query processor in increasing timestamp order. Before processing a tuple t with timestamp ts, it is important to guarantee that no more tuples with timestamp less than ts will arrive to the system. Stream heartbeats are a mechanism that provide such assurance. Each data source is responsible for generating its own heartbeats periodically and embeds them within its own data stream. In cases when a data source does not have this capability, the data stream management system should be able to synthesize and generate heartbeats for each data stream based on parameterized knowledge of the environment, e.g., the maximum network delay.

When processing a query that accesses multiple streams, say s_1, s_2,...,s_n, the query processor computes the minimum timestamp, say ts_{min}, of the heartbeats of all the n streams. All the tuples in the system buffer with timestamps less than ts_{min} are forwarded to the query for processing. Such query-level heartbeats are simple and easy to implement. However, they can block the processing of a query unnecessarily, e.g., when one of the streams is temporarily blocked while the others are available. Alternatively, in contrast to a

query-level heartbeat, in an operator-level heartbeat, the query processor computes the minimum timestamps of heartbeats for streams input to each query operator in the plan. In this case, each operator will have its own heartbeat.

Input tuples with timestamps less than an operator's heartbeat are forwarded to that operator. Latency and memory requirements of both query-level and operator-level heartbeats are further detailed in [14].

Incremental Query Evaluation

In incremental query evaluation, whenever the window contents change, the query does not reprocess all the tuples inside the window. Instead, the query only processes the changed tuples. Moreover, only the changes to the query answer are reported. The answer to the query is considered more as a materialized view in the sense that only the changes in the base tables (the deltas) need to be processed by the view expression. As a result, some new tuples get inserted into the view while others get deleted from it.

Two types of events need to be handled in the case of incremental query evaluation. The first event type is when a new tuple arrives into the data stream and hence becomes inside of the stream's window. The second event type is when a tuple exits the window. For example, when a time-sliding window slides, a tuple's timestamp may become outside of the time interval covered by the window, and hence the tuple expires and exits the window.

Systems vary on how they handle both types of events. STREAM [16] generates two types of streams: an Insert (I) stream that contains the newly arriving tuples, and a Delete (D) stream that contains the expiring tuples. In contrast to STREAM that has two types of streams, Nile [8] maintains only one stream but with two types of tuples: Positive and Negative Tuples. Positive tuples correspond to the new tuples that arrive into the window while negative tuples correspond to the expiring tuples. Positive tuples are the regular tuples that a traditional query processor handles. In contrast, negative tuples are processed differently by each query operator.

Incremental Query Processing Using Negative Tuples

In a traditional relational query evaluation plan (QEP), a table-scan operator is usually at the bottom (leaf) level of the QEP and serves as the interface operator that pipelines the tuples of a relation to the rest of the QEP. In incremental query evaluation, the equivalent to a table-scan operator is the window-expire operator (W-expire, for short). A W-expire operator is assigned to each stream in the query and is added at the bottom (leaf) level of a QEP. The inputs to the W-expire operator are the raw stream tuples as well as the definition of the window (e.g., say a window of range 10 min and slide 2 min). The function of the W-expire operator is two-fold: (i) introduce the new stream tuples to the QEP as they arrive into the stream, and (ii) generate (or synthesize) negative tuples that correspond to stream tuples that expire from the window and introduce them to the QEP.

Conceptually, a negative tuple, say t^-, needs to trace the same path in the QEP that its corresponding positive tuple, say t^+, took to produce the same results that t^+ produced (but with negative sign). Once it receives a negative tuple, each query operator, in turn, processes the negative tuple and produces 0 or more negative tuples as output that get processed by the operators next in the pipeline. The semantics of all query operators are extended to handle positive as well as negative tuples. Additionally, each query operator needs to store additional state information to be able to process negative tuples successfully. This is similar to the case of processing revision tuples in Borealis. Details about the extended semantics and state information of query operators for incremental query processing as well as optimizations to reduce the cost of handling negative tuples can be found in [6].

Query Processing for Predicate Windows

In contrast to a sliding window, in a predicate window, a window is specified by a predicate that defines the stream data tuples of interest to a query. For example, a query may define its window of interest to be the room identifiers of the rooms with temperatures greater than 110 F. When a room's temperature is below 110 F, that room identifier exits the window, otherwise it remains inside the window of interest.

Predicates that define a window may greatly vary in complexity. In general, a predicate window can be expressed using a regular select-from-where query. Consider the predicate window that is defined by following Select query over Stream roomTemperatures: Select room-id, temperature from roomTemperatures where temperature >120. The room-ids and temperature values of the high-temperature rooms are maintained inside this predicate window.

Three types of tuples need to be maintained in this scenario (insert, delete, and update tuples). Insert and delete tuples are the same as the positive and negative tuples used for incremental evaluation, while update tuples are similar to revision tuples. When the temperature of Room 5 increases from 110 to 130 F (i.e., the room did not have a high temperature before), the corresponding tuple (room-id, temperature): (5, 130) enters into the predicate window via an insert tuple. Similarly, when the temperature of the same room changes from 130 to 140 F, the corresponding tuple (5, 140) replaces the old tuple (5, 130) via an update tuple. Finally, when the temperature of Room 5 goes below 120 F, e.g., 70 F, the corresponding tuple (5, 140) is eliminated from the predicate window via a delete tuple. Because of the generality of predicate windows and the use of insert, delete, and update tuples, predicate windows can be used to support views over data streams [5].

Shared Execution of Multiple Window-based Queries

In continuous stream query processing, it is imperative to exploit resources and commodity among query expressions to achieve improved efficiency. Shared execution is one important means of achieving this goal. The role of joins in particular is further enhanced in stream query processing due to the use of "selection pull up." In the NiagaraCQ system [9,10], it was shown that the traditional heuristic of pushing selection predicates below joins is often inappropriate for continuous query systems because early selection destroys the ability to share subsequent join processing. Since join processing is expensive relative to selections, it is almost always beneficial to process the join once in shared mode and then subject the produced join output tuples to the appropriate selection operations. A similar argument holds for aggregation operations and Group-by.

Multiple windowed query operators, e.g., window joins, can share their execution as the windows of interest over one data stream overlap. For example, consider the two streams A and B and the two join operators J and K that use the same join predicate, e.g., A.a = B.b. Assume an operation-based time-sliding windows of sizes w_J for J and w_K for K, e.g., w_J = 1 h or w_K = 1 day. Since J and K have the same join predicate and $w_J < w_K$, the join output tuples of J are a subset of the join output tuples. Notice that it is possible that two tuples, say A_1 and B_1, join and their timestamps are more than 1 h apart but less than 1 day. Hence, the join output pair $<A_1, B_1>$ is part of K's join output tuples but is not part of J's. Executing both operations separately wastes system resources. Sharing the execution of the join operator would save on CPU and memory resources. One feasible policy is to perform only the join with the largest window (K in the example, since w_K is the larger of the two windows) [4,12]. The output of this join is then routed to multiple output streams (two streams in the example), where the output tuples for the joins with smaller windows get filtered out based on their timestamps. One advantage of this policy is its simplicity since arriving tuples are completely processed before considering the next incoming tuple. However, this policy has a disadvantage in that it delays the processing of small-window joins until the largest-window join is completely processed, hence negatively affecting the response time of small-window joins.

An alternative policy for shared execution of window joins is to perform the join with the smallest window first by all new tuples, then the next larger window joins, and so on until the largest window join is processed [7]. As long as there are tuples to be processed by the smallest window, they are processed first before any tuples are processed by the larger windows. Notice that the join output pairs produced for the smallest windows are also directly output tuples for all the larger windows, but the converse is not true, as explained above. The response time for the small-window joins is significantly enhanced. However, there is significant state information that needs to be maintained for each tuple to know at what point inside the windows from which that tuple resumes execution with one of the window joins. More detail and tradeoffs of each of the policies as well as better optimized policies for shared execution of joins can be found in [7].

Out-of-order Tuple Processing

In processing queries over data streams, it is often the case that the window operators are order sensitive. For example, when continuously computing the running average over the most-recent ten tuples in a stream, the output result depends on the order of the tuples in the stream. With the ordered arrival of a new tuple, the 11th most-recent tuple gets dropped and the new average is computed. The same is true for time-sliding windows. When tuples arrive in order, the timestamp of the new tuple is used to slide the window. This is not

the case when tuples arrive out of order. In this case, none of the windows can get closed and any produced answer pertaining to a given window cannot be considered final since at any future point in time, a tuple can arrive into that window and hence the output needs to be recalculated.

There are several ways to deal with out-of-order tuples. As explained in Section "Query Processing using Revision Tuples," revision tuples are one way to deal with out-of-order tuples. The timestamp, say TS, of the most-recent tuple of a stream is maintained. Whenever an out-of-order tuple arrives (this tuple's timestamp is less than TS), a revision tuple is issued to revise the output of the affected window(s). When the maximum possible delay of a tuple is known in advance, the stream query processor can leave open all the potential windows that are within this delay window. Once the maximum delay is reached, the corresponding windows get closed and the tuples inside the window are processed to produce the final answers for that window. The input stream manager can synthesize and issue a punctuation or a heartbeat to that effect so that the window results can be computed and finalized.

Key Applications

Key applications for window-based stream query processing include computer network traffic analysis, network performance monitoring, intrusion detection. stock market data analytics, web browsing clickstream analytics, algorithmic stock trading, sensor network data summarization and processing, moving object tracking, traffic monitoring, and surveillance applications.

Cross-references

- ▶ Adaptive Stream Processing
- ▶ Continuous Queries in Sensor Networks
- ▶ Continuous Query
- ▶ Data Stream
- ▶ Distributed Streams
- ▶ Event Stream
- ▶ Fault-Tolerance and High Availability in a Data Stream Management Systems
- ▶ Histograms on Streams
- ▶ Publish/Subscribe over Streams
- ▶ Stream Models
- ▶ Stream Processing
- ▶ Stream-Oriented Query Languages and Operators
- ▶ Windows
- ▶ XML-Stream Processing

Recommended Reading

1. Abadi D., Ahmad Y., Balazinska M., Cetintemel U., Cherniack M., Hwang J-H., Lindner W., Maskey A.S., Rasin A., Ryvkina E., Tatbul N., Xing Y., and Zdonik S. The design of the Borealis stream processing engine. In Proc. 2nd Biennial Conf. on Innovative Data Systems Research, 2005, pp. 277–289.
2. Abadi D., Carney D., Cetintemel U., Cherniack M., Convey C., Lee S., Stonebraker M., Tatbul N., and Zdonik S. Aurora: a new model and architecture for data stream management. VLDB J., 12(2):120–139, 2003.
3. Bai Y., Thakkar H., Luo C., Wang H., and Zaniolo C. A data stream language and system designed for power and extensibility. In Proc. Int. Conf. on Information and Knowledge Management, 2006, pp. 337–346.
4. Chandrasekaran S. and Franklin M.J. Streaming queries over streaming data. In Proc. 28th Int. Conf. on Very Large Data Bases, 2002, pp. 203–214.
5. Ghanem T.M. Supporting Views in Data Stream Management Systems. Ph.D. Dissertation. Department of Computer Science, Purdue University, 2007.
6. Ghanem T.M., Hammad M.A., Mokbel M.F., Aref W.G., and Elmagarmid A.K. Incremental evaluation of sliding-window queries over data streams. IEEE Trans. Knowl. Data Eng., 19(1): 57–72, 2007.
7. Hammad M.A., Franklin M.J., Aref W.G., and Elmagarmid A.K. Scheduling for shared window joins over data streams. In Proc. 29th Int. Conf. on Very Large Data Bases, 2003, pp. 297–308.
8. Hammad M.A., Mokbel M.F., Ali M.H., Aref W.G., Catlin A.C., Elmagarmid A.K., Eltabakh M., Elfeky M.G., Ghanem T., Gwadera R., Ilyas I.F., Marzouk M., and Xiong X. Nile: a query processing engine for data streams. In Proc. 20th Int. Conf. on Data Engineering, 2004, p. 851.
9. Jianjun C., DeWitt D.J., Feng T., and Yuan W. NiagaraCQ: a scalable continuous query system for internet databases. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 2000, pp. 379–390.
10. Jianjun C., DeWitt D.J., and Naughton J.F. Design and evaluation of alternative selection placement strategies in optimizing continuous queries. In Proc. 18th Int. Conf. on Data Engineering, 2002, pp. 345–356.
11. Johnson T., Muthukrishnan S., Shkapenyuk V., and Spatscheck O. A heartbeat mechanism and its application in gigascope. In Proc. 31st Int. Conf. on Very Large Data Bases, 2005, pp. 1079–1088.
12. Madden S., Shah M.A., Hellerstein J.M., and Raman V. Continuously adaptive continuous queries over streams. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 2002, pp. 49–60.
13. Ryvkina E., Maskey A.S., Cherniack M., and Zdonik S. Revision processing in a stream processing engine: a high-level design. In Proc. 22nd Int. Conf. on Data Engineering, 2006.
14. Srivastava U. and Widom J. Flexible time management in data stream systems. In Proc. 23rd ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems, 2004, pp. 263–274.

15. Stonebraker M., Cetintemel U., and Zdonik S. The 8 requirements of real-time stream processing. ACM SIGMOD Rec., 34 (4):42–47, 2005.
16. The STREAM Group. STREAM: the Stanford stream data manager. IEEE Data Eng. Bull., 26(1):19–26, 2003.
17. Tucker P.A., Maier D., Sheard T., and Fegaras L. Exploiting punctuation semantics in continuous data streams. IEEE Trans. Knowl. Data Eng., 15(3):555–568, 2003.

Windows

Carlo Zaniolo
University of California-Los Angeles, Los Angeles, CA, USA

Synonyms

Logical window = Time-based window; Physical window = Tuple-based windows

Definition

Windows were introduced as part of SQL:1999 OLAP Functions. For instance, given a sequence of bids it is possible to use the following SQL:2003 statement to find the last 40 offers (the current offer and the previous 39) for item 0021:

```
SELECT itemID, avg(Offer)
OVER(ROWS 39 PRECEDING ORDER BY TIME)
FROM BIDS
WHERE ItemID=0021
```

When BIDS is instead a data stream, the "ORDER BY TIME" clause becomes redundant, and clauses such as "FOLLOWING" are often not supported in continuous query languages. However, these languages still provide a "PARTITION BY" clause (or the more traditional "GROUP BY" clause), whereby a user can specify that the average of the last 40 offers must be computed for all items, not just item 0021. In addition to entailing powerful and flexible analytical queries on ordered sequences and time-series, as in databases, windows on data streams play the key role of *synopses*, and are widely employed in this capacity. In particular, since it is not feasible to memorize unbounded data streams, window joins are used instead. In window joins, the newly arriving tuples in one data stream are joined with the recent tuples in the window of the other data streams (and symmetrically). The design and implementation of window-based extensions for aggregates and joins operators for data streams has generated significant research work [1,2,3].

Key Points

Traditional SQL aggregates are blocking (and thus not suitable for continuous queries on data streams) but their window versions are not. Therefore, the window concept is the cornerstone of many continuous query languages, where its functionality has also been extended with new constructs, such as *slides*, *tumbles*, and *landmark* windows, that are not in SQL:2003. In a sliding window of w tuples (seconds) the aggregate computed over w is returned for each new incoming tuple: when a slide of size s is also specified, then results are only returned every s tuples (seconds). With $s = w$ we have a tumbling window in which results are only returned at the end of each window. When $w/s = k$, then the window, and the computation of the aggregate, is partition into k panes [2]. A landmark window is one where an occurrence of some event of semantic significance, e.g., a punctuation mark [3], defines one or both endpoints. Efficient implementation requires delta computations that exploit the algebraic properties of aggregates – e.g., by increasing (decreasing) the current sum with the value from the tuple entering (leaving) the window – and architectures that consolidate the vast assortment of windows and aggregates at hand [1,2]. Windows provide the basic synopsis needed to support joins with limited memory. Special techniques are used to optimize multi-way joins, and response time for all joins. Further optimization issues occur when load-shedding is performed by either (i) using secondary store to manage overflowing window buffers, or (ii) dropping tuples from the windows in such a way that either a max-subset or a random sample of the original window join is produced. Aged-based policies, where older tuples are dropped first, is preferable in certain applications. Sketching techniques are often used to estimate the productivity of tuples, whereby the least productive tuples are dropped first [3]. The same techniques can also be used to estimate duplicates in DISTINCT aggregates; reservoir-sampling inspired techniques have instead been proposed for aggregates where duplicates are not ignored.

Cross-references

▶ Data Sketch/Synopsis
▶ Join

- ▶ One-Pass Query Processing
- ▶ Punctuations
- ▶ SQL
- ▶ Stream Processing

Recommended Reading

1. Bai Y. et al. A data stream language and system designed for power and extensibility. In Proc. Int. Conf. on Information and Knowledge Management, 2006, pp. 337–346.
2. Li J. et al. Semantics and evaluation techniques for window aggregates in data streams. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 2005, pp. 311–322.
3. Maier D., Tucker P.A., and Garofalakis M. Filtering, punctuation, windows and synopses. In Stream Data Management, Vol. 30, N. Chaudhry, K. Shaw, M. Abdelguerfi (eds.). Kluwer, Dordecht, 2005, pp. 35–56.

Wireless Sensor Networks

- ▶ Sensor Networks

Within-Element Term Frequency

- ▶ Term Statistics for Structured Text Retrieval

Word Conflation

- ▶ Stemming

Word of Mouth

- ▶ Reputation and Trust
- ▶ Trust and Reputation in Peer-to-Peer Systems

Work Element

- ▶ Activity

Work Performer

- ▶ Actors/Agents/Roles

Workflow

- ▶ Workflow Management

Workflow Constructs

Nathaniel Palmer
Workflow Management Coalition, Hingham, MA, USA

Synonyms

Process semantics

Definition

The elements of a Workflow Model defining how the process is executed.

Key Points

Workflow Constructs represent the elements of a Workflow Model, such as Control Data, Splits, Joins, Activities and Actors, which combined define the parameters of how a process instances is executed. The "Semantics" of the process refer to how it is executed in the context of workflow rules, such the steps, sequencing and dependencies; whereas the "Syntax" of the process refers to its definition within the context of a specific language. Workflow Constructs is directly related to semantics, however, does not define specific syntax.

Cross-references

- ▶ Activity
- ▶ Actors/Agents/Roles
- ▶ Control Data
- ▶ Join
- ▶ Split
- ▶ Workflow Model

Workflow Control Data

- ▶ Control Data

Workflow Enactment Service State Data

- ▶ Control Data

Workflow Engine State Data

► Control Data

Workflow Evolution

Peter Dadam, Stefanie Rinderle
University of Ulm, Ulm, Germany

Synonyms

Process evolution; Schema evolution in workflow management systems; Schema evolution in process management systems; Adaptive workflow/process management; Workflow/process instance changes

Definition

The term evolution has been originally used in biology and means the progressive development of a species over time, i.e., the adaptation to changing environmental requirements. Business processes (which are often called workflows when implemented and thus automated within a workflow management system) also "live" within an environment (e.g., the enterprise or the market). This environment is typically highly dynamic and thus the running workflows have to adapt to these changing requirements – i.e., to *evolve* – in order to keep up with the ever-changing business environment and provide their users with the competitive edge.

Workflow evolution implies two basic challenges: *change realization* and *change discovery*. Change realization means that it must be *technically* possible to adapt workflows. Change discovery refers to the question which workflow modifications are required when the environmental requirements change. So far, research has focused on the first topic whereas the second one is subject to future research.

Regarding the technical realization, workflow changes can take place at two levels – at the workflow instance and the workflow type level. Instance-specific changes are often applied in an ad-hoc manner and become necessary in conjunction with real-world exceptions. They usually affect only single workflow instances. As opposed to this, in conjunction with workflow schema changes at the workflow type level, a collection of related instances may have to be adapted, i.e., workflow schema changes are *propagated* to running workflow instances or running workflow instances are *migrated* to the changed workflow schema.

Historical Background

In research workflow evolution has been mainly addressed from a technical point of view so far. More precisely, different approaches have been developed for the realization of workflow change during the last 10 years [8]. These approaches can be divided into different categories, i.e., pre-planned workflow changes, flexibility by design, and flexible workflow management technology.

Pre-planned workflow changes: In this category workflow changes are planned in advance [3,6]. Then, based on different strategies, one or several of these changes are executed during runtime if necessary (e.g., if an exception occurs). Strategies to initiate a workflow change can be rule-based (e.g., using ECA rules), goal-based (e.g., using AI planning techniques), or process-driven (e.g., based on graph grammars).

Flexibility by design: Workflow adaptations can be also foreseen in the workflow description [1,12] (e.g., in the workflow graph). The approaches range from simply modeling alternative branches with associated selection codes to approaches which specify workflows only as a rough "skeleton" in advance and leave the detailed specification of the workflow to the user at runtime. Then, for example, the user can select which activities are to be executed in which order. Some approaches also integrate the support of planning methods from Artificial Intelligence for the runtime workflow specification.

Flexible workflow management technology: Approaches in this field either deal with the modification of workflow instances [7] or with workflow schema evolution (i.e., changes at workflow type level) [2,11,13,14]. Basic questions, however, are very similar for workflow schema and instance changes, mainly concerning the correctness of the workflow management systems after changes at either level have been carried out. Consequently, different change frameworks on workflows have been defined [15] and several correctness criteria have been developed which partly depend on the used workflow meta model (e.g., Petri Nets) [8]. Whereas the correctness question is more or less sufficient to deal with workflow instance changes, other approaches have discovered that, for workflow schema changes, additional challenges arise. The reason is that if a workflow schema is modified, a possibly large number of workflow instances running according to the workflow schema is affected as well. Thus, approaches arose to deal with

efficiency and usability aspects. Furthermore, some proof-of-concept prototypes were implemented.

The last development stage of technical workflow evolution research has been made when considering workflow schema and instance changes in their interplay and not in an isolated manner [9].

Currently, more and more approaches start to integrate semantic knowledge into workflow management systems. This can be used as a basis for developing intelligent adaptive systems which are able to learn from previous changes and therefore automatically adapt their workflows when the environment changes (cf. Future Directions).

Foundations

As already mentioned, workflow evolution comprises two basic challenges – *change discovery* and *change realization*. Change discovery refers to the detection of necessary business process optimizations which should be brought into the running workflows. For change discovery different approaches have emerged recently which are, for example, based on process mining or case-based reasoning techniques. Since these trends are currently "in the flow" the reader is referred to the "Future Directions" section for more details.

Contrary, the topic of workflow change (i.e., how to bring the discovered changes into the running system) has been investigated in great detail. Note that the following discussion of scientific challenges in the context of workflow evolution refers to dynamic workflow changes (i.e., changing a workflow instance or a workflow schema during runtime) and not to pre-planned workflow changes. Fig. 1a shows an example of an order workflow based on which the problem spectrum is motivated in the following.

At workflow type level, changes are handled by modifying the affected *workflow schema S* based on which a collection of workflow instances *I*1,..., *In* is running. Intuitively, it is necessary that the modification of a *correct* workflow schema *S* again results in a *correct* workflow schema S' (a workflow schema is called correct if it satisfies correctness constraints set out by the underlying workflow meta model, i.e., the formalism used to describe the business processes, e.g., Petri Nets). This so called *static* schema correctness can be preserved, for example, by the applied change operations.

After changing workflow schema *S* it is also necessary to deal with process instances I1,..., *In* running on S (cf. Fig. 1b). Different strategies for this so-called *dynamic process schema evolution* have been proposed in literature. One possibility is to *cancel* all running instances and to restart them according to the new workflow schema. However, this strategy may cause an immense loss of work and would usually not be accepted by users. Therefore, at minimum, it is claimed that workflow instances I1,..., In started according to workflow schema *S* can be finished on *S* without being interrupted. This strategy can be implemented by providing adequate versioning concepts and may be sufficient for workflows of short duration. However, doing so raises severe problems in conjunction with long-running workflows as often found in, for example, clinical or engineering environments. Due to the resulting mix of workflow instances running on old and new schema versions a chaos within the production or the offered services may occur. Furthermore, often, it is not acceptable to run instances on the old schema version if laws or business rules (e.g., clinical guidelines) are violated.

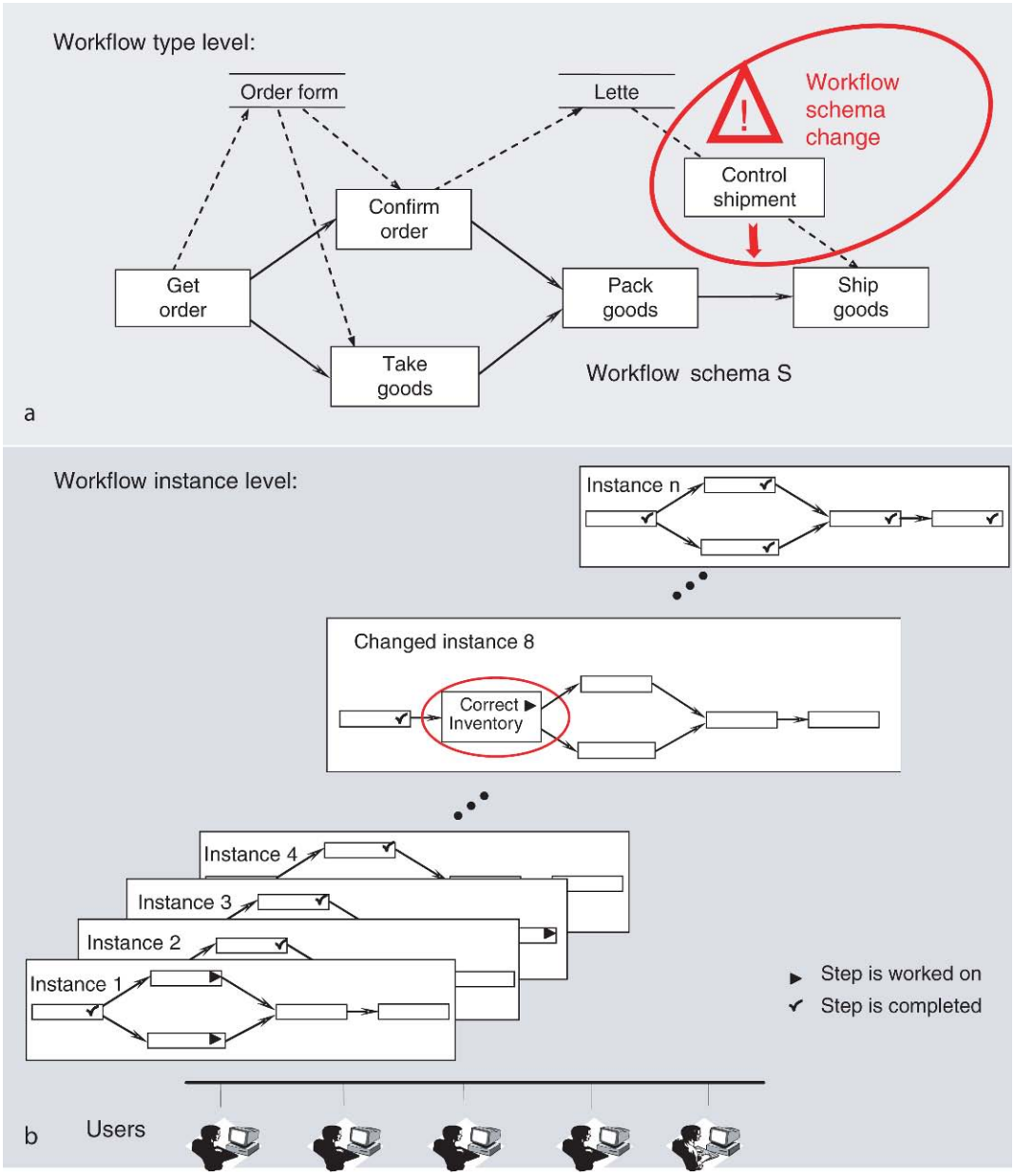
For these reasons, it is quite important to be able to apply workflow schema changes to running instances as well. This scenario is called the *propagation* of workflow schema changes to running workflow instances or, in other words, the *migration* of the ("old") running instances to the changed workflow schema. Note that the number of running instances *I*1,..., *In* may be very large. In environments like hospitals or telecommunication companies, for example, $n > 10{,}000$ may easily hold.

To better understand the challenge of propagating a workflow schema change to running workflow instances, one can compare a running workflow instance with an application program. This program is currently executed and consists of several procedures. These steps can be successively executed (sequential execution), can be situated within if-then-else conditions (alternative branches), or can be even executed in parallel threads. Workflow schema evolution can then be compared to manipulating the running program by inserting one or more procedures, deleting procedures, or changing the order of procedures in the midst of program execution. In particular, the changes have to be carried out by maintaining a correct program execution (e.g., correct parameter provision) for all possible execution alternatives.

Consequently, it is a non-trivial but crucial task for workflow management systems to enable workflow

schema evolution. The following main requirements have been identified:

1. *Completeness*: Workflow designers must not be restricted, neither by the used workflow meta model nor by the offered change operations.
2. *Correctness*: The ultimate ambition of any adaptive/flexible workflow management system must be the correctness of workflow changes; i.e., introducing changes to the runtime system without causing inconsistencies or errors (like deadlocks or improperly invoked activity programs). Therefore, adequate *correctness criteria* are needed. These criteria must not be too restrictive, i.e., no workflow instance should be needlessly excluded from applying a dynamic change.
3. *Efficient Correctness Checks*: Assume that an appropriate correctness criterion for workflow changes has been found. Then the challenge is to check this criterion *efficiently*. This is especially important for



Workflow Evolution. Figure 1. Example order workflow.

large-scale environments with hundreds (up to thousands) of running workflow instances.

4. *Usability*: The migration of workflow instances to a changed workflow schema must not require expensive user interactions. First of all, such interactions may lead to delays within the instance executions. Secondly, users (e.g., designers or administrators) must not be burdened with the job to adapt workflow instances to changed workflow schemes. Complex process structures and extensive state adaptations might overstrain them quickly. Therefore, it is crucial to offer methods for the automatic adaptation of workflow instances.

However, supporting workflow schema changes is not sufficient to enable workflow evolution. In fact, it must be possible to modify single workflow instances as well. Such workflow instance changes are often carried out in an ad-hoc manner in order to deal with an exceptional situation, e.g., an unforeseen correction of the inventory information in an order workflow as depicted for instance 8 in Fig. 1b.

In the literature, workflow schema and instance changes have been an important research topic for several years. So far, there are only a few adaptive systems supporting both kinds of changes in one system [4,14], however, only in a separated manner (i.e., already individually modified workflow instances are excluded from migration to the changed workflow schema). As discussed, this approach is not sufficient in many cases, particularly in connection with long-running workflows. Assume, for example, a medical treatment workflow which is normally executed in a standardized way for every patient. Assume further that due to an unforeseen reaction, an additional drug is given to a certain patient. However, this deviation from the standard procedure must not imply that the affected workflow instance (and therefore the patient) is excluded from further workflow optimizations. Therefore, it must be possible to propagate workflow schema changes at the type level to individually modified instances as well.

This *interplay* between *concurrently* applied workflow schema and instance changes raises many challenges, for example, the question to which degree the changes *overlap*. Overlapping means that workflow schema and instance change (partly) have the same effects on the underlying workflow schema. Such an overlap occurs, for example, if a workflow instance has (partly) anticipated a future workflow schema change in conjunction with a flawed process design. In this situation conflicts may arise between the overlapping workflow schema and instance changes (e.g., if the same activities are deleted).

Different approaches have been developed to deal with concurrent workflow schema and instance changes (e.g., based on workflow schema and change comparisons in order to determine overlapping degrees between changes) [9]. Thus, the current state-of-the art comprises sufficient solutions for the technical realization of workflow changes. However, there are advanced challenges regarding the semantic realization of workflow changes and the acquisition of the workflow changes. These challenges are described within the "Future Directions" section.

Key Applications

Production workflow (process) management systems

Clinical workflows
Office automation
Development workflows (e.g., in the automotive domain)
Enterprise application integration (?)

Future Directions

Currently, there are comprehensive solutions available regarding the technical support of workflow changes. However, this is not sufficient for a complete workflow evolution solution. First of all, in addition to the technical requirements as discussed in the "Foundations" section, also semantic aspects must be taken into account. In order to make this technology broadly applicable, it is not sufficient to check only whether the planned workflow changes violate any structural or state-related correctness properties. In addition, it must also be possible to verify if semantic constraints (e.g., business rules, four eyes principle) imposed on the affected workflow are violated [5].

In some cases the necessity to change a workflow schema has reasons like changes in legislation, in global business rules, or in the structural organization of the enterprise etc. In many cases, however, workflow schema changes are motivated by the observation that the existing workflow does not meet some real-world requirements. For example, the order of steps is not optimal, some steps are missing or not necessary, or important cases are not reflected in the workflow

schema. It should not be left to users to detect necessary changes. If users find "work-arounds" to locally solve their problem, this information may never reach the person in charge. More promising is to let the system analyze execution logs in order to discover repetitive deviations (e.g., using process mining), to compute an alternative workflow schema covering these cases, and to support the process designer to (semi-)automatically perform the necessary workflow schema evolution [10].

Cross-references

- ▶ Business Intelligence
- ▶ Business Process Execution Language
- ▶ Business Process Management
- ▶ Business Process Reengineering
- ▶ Composed Services and WS-BPEL
- ▶ Petri Nets
- ▶ Process Mining
- ▶ Process Optimization
- ▶ Workflow Constructs
- ▶ Workflow Management and Workflow Management System
- ▶ Workflow Management Coalition
- ▶ Workflow Model
- ▶ Workflow Patterns
- ▶ Workflow Schema
- ▶ Workflow Transactions

Recommended Reading

1. Adams M., ter Hofstede A.H.M., Edmond D., and van der Aalst W.M.P. A service-oriented implementation of dynamic flexibility in workflows. In Proc. Int. Conf. on Cooperative Inf. Syst., 2006.
2. Casati F., Ceri S., Pernici B., and Pozzi G. Workflow evolution. Data Knowl. Eng., 24(3):211–238, 1998.
3. Heimann P., Joeris G., Krapp C., and Westfechtel B. DYNAMITE: dynamic task nets for software process management. In Proc. 18th Int. Conf. on Software Eng., 1996, pp. 331–341.
4. Kochut K., Arnold J., Sheth A., Miller J., Kraemer E., Arpinar B., and Cardoso J. IntelliGEN: a distributed workflow system for discovering protein-protein interactions. Distr. Parallel Databases, 13(1):43–72, 2003.
5. Ly L.T., Rinderle S., and Dadam P. Semantic correctness in adaptive process management systems. In Proc. Int. Conf. Business Process Management, 2006, pp. 193–208.
6. Müller R., Greiner U., and Rahm E. AgentWork: a workflow system supporting rule-based workflow adaptation. Data Knowl. Eng., 51(2):223–256, 2004.
7. Reichert M. and Dadam P. ADEPTflex – supporting dynamic changes of workflows with-out losing control. J. Intell. Inform. Syst., 10(2):93–129, 1998.
8. Rinderle S., Reichert M., and Dadam P. Correctness criteria for dynamic changes in workflow systems – a survey. Data Knowl. Eng., 50(1):9–34, 2004.
9. Rinderle S., Reichert M., and Dadam P. Disjoint and overlapping process changes: challenges, solutions, applications. In Proc. Int. Conf. on Cooperative Inf. Syst., 2004, pp. 101–120.
10. Rinderle S., Weber B., Reichert M., and Wild W. Integrating process learning and process evolution – a semantics based approach. In Int. Conf. Business Process Management, 2005, pp. 252–267.
11. Sadiq S., Marjanovic O., and Orlowska M. Managing change and time in dynamic workflow processes. Int. J. Cooper. Inform. Syst., 9(1, 2):93–116, 2000.
12. Sadiq S., Sadiq W., and Orlowska M. Pockets of flexibility in workflow specifications. In Proc. 20th Int. Conf. on Conceptual Modeling, 2001, pp. 513–526.
13. van der Aalst W.M.P. Exterminating the dynamic change bug: a concrete approach to support workflow change. Inform. Syst. Front., 3(3):297–317, 2001.
14. Weske M. Formal foundation and conceptual design of dynamic adaptations in a workflow management system. In Proc. Hawaii Int. Conf. System Sciences, 2001.
15. Weber B., Rinderle S., and Reichert M. Change patterns and change support features in process-aware information systems. In Proc. Int. Conf. Advanced Information Systems Engineering, 2007, pp. 574–588.

Workflow Join

Nathaniel Palmer
Workflow Management Coalition, Hingham, MA, USA

Synonyms

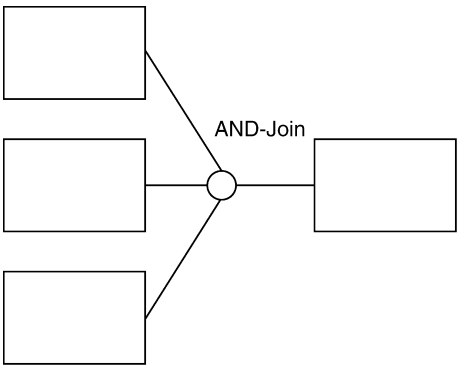
AND-join; Rendezvous; Synchronization join

Definition

A point in the workflow where two or more parallel executing activities converge into a single common thread of control.

Key Points

The execution of parallel activities commences with an AND-Split and concludes with an AND-Join. For example, in a credit application process there may be a split in the workflow at which point multiple activities are completed separately (in parallel, if not simultaneously.) Each parallel executing thread is held until the set of all thread transitions to the next activity is completed at which point the threads converge and the next activity is initiated.



Cross-references

- ▶ AND-Split
- ▶ OR-Join
- ▶ Process Life Cycle
- ▶ Workflow Management and Workflow Management System

Workflow Lifecycle

- ▶ Process Life Cycle

Workflow Loop

- ▶ Loop

Workflow Management

Nathaniel Palmer
Workflow Management Coalition, Hingham, MA, USA

Synonyms

Workflow; Business process management; Case management

Definition

The automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules.

Key Points

The automation of a business process is defined within a Process Definition, which identifies the various process activities, procedural rules and associated control data used to manage the workflow during process enactment.

Many individual process instances may be operational during process enactment, each associated with a specific set of data relevant to that individual process instance (or workflow "Case"). A loose distinction is sometimes drawn between production workflow, in which most of the procedural rules are defined in advance, and ad-hoc workflow, in which the procedural rules may be modified or created during the operation of the process.

Cross-references

- ▶ Business Process Management
- ▶ Event-Driven Business Process Management
- ▶ Workflow Management and Workflow Management System
- ▶ Workflow Model
- ▶ XML Process Definition Language

Workflow Management and Workflow Management System

Johann Eder
University of Vienna, Vienna, Austria

Synonyms

Business process management

Definition

Business Process: A set of one or more linked procedures or activities that collectively realize a business objective or policy goal, normally within the context of an organizational structure defining functional roles and relationships [10].

Workflow: The automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules [10].

Workflow Management System (WFMS): A system that defines, creates and manages the execution of workflows through the use of software, running on one or more workflow engine, which is able to interpret

the process definition, interact with workflow participants and, where required, invoke the use of IT tools and applications [10].

Historical Background

In the 1980s, office automation was a major research focus with several topics, the support for the including movement of documents through offices. Some proposals were made to model document or form flows and to support the "paperless" office.

Another driver for the development of workflows came from extended transaction models. The problem was to solve integrity problems in heterogenous information systems where, within one business process, several databases are updated. Integrity was compromised if some of the updates failed or were not carried out. Extended transaction models aimed to provide assurance that all steps were performed properly.

A third development stream came from computer supported cooperative work, where the need arose to better support repeating processes.

In the 1980s revolutionary ideas for improving business by business process reengineering also emerged. Enterprises were urged to organize themselves around their core business processes and not according to functional considerations as before. This movement in business administration called for software to support process driven organizations.

Since the 1990s, hundreds of workflow systems have been developed, both academic prototypes and commercial products with quite different functionalities.

Foundations

Workflows automate and/or support the execution of business processes. The core idea of workflow management is to separate the process logic from the functional application logic. This means that applications provide functionality, and the workflow system controls, when to involve which functionality. So the workflow is responsible for *who* (which actor) does *what* (which activity), with *which data* and resources, *when* (control logic) under *which constraints*. And it is the job of the application system or workflow participant to know *how* an activity is performed. So the workflow system is responsible for the logistics of a process but not for the execution of the individual steps. This separation of concerns, to separate the doing of things from the organization of the process is supposed to make software more flexible and to integrate heterogeneous applications.

Aspects of Workflows

A workflow consists of a set of activities and the dependencies between them, the so-called business logic. These dependencies can be expressed in different forms. The most frequent way is to express the dependencies by control flow definitions, often supported by graphical notations. Other ways are data flow or rules. However, the control flow is only one aspect of workflow management, although it has received most attention.

The following aspects of workflows can be distinguished:

Activities: Activities are units of work to be performed, either by a specific program or by an user, typically interacting with some application programs. Activities can be elementary, i.e from the point of the workflow systems they are the smallest units of work. Elementary activities are frequently called tasks. Activities can be composed to complex activities.

Process: The process defines the control flow, i.e., the admissible relative orders of activity executions. Typical elements of control flow are sequence, and-parallelism (concurrent execution of activities), or-parallelism (selection of different activities, inclusive or exclusive). Workflow models provide the means for defining the process. Many process models rely on (variants of) Petri-nets or (simpler) variants of Petri-nets, in particular, so called workflow nets. Workflow nets are graphs, where the nodes are activities and the directed edges represent a precedence relation between activities. Additionally, there are control nodes: and-split for invoking several successor activities concurrently, and-join for synchronizing parallel threads, xor-splits for selecting one of several successor activities and xor-join for joining alternate paths again.

Actors: Actors perform the tasks defined in the workflow specification. Actors can be humans or systems. Systems perform so-called automatic tasks, i.e., execute some application programs. Human actors can be identified directly (by a user name), or indirectly by reference to a position in the organization, by roles, representing a set of users competent for fulfilling a tasks, by organizational unit, by specification of necessary skills and competencies, or by reference to some data item containing the actor. For indirect representation of the individual actor, the workflow system has to resolve the role at runtime and assign a user to

a particular task. For the resolution of actors specified by organizational units, the workflow system maintains a model of the organization.

Data: Data are needed for the processing of a workflow. Three kinds of data comprise workflows. *Workflow control data* are managed by a WFMS and describe workflow execution (e.g., control and data flow among activities), relevant internally for a WFMS and without a long-term impact beyond the scope of the current workflow (e.g., a termination state of an activity). *Application data* are managed by the applications supporting the process instance and generally are never seen by the WFMS. *Workflow relevant data* are used by the WFMS to determine the state transitions of a workflow (e.g., transition conditions) and may be accessed both by the WFMS and the applications. Further, a workflow system manages *organizational data* which it uses at runtime to resolve actors, and *audit data* kept in the workflow log documenting the execution details of workflow instances.

Applications: Applications are pieces of software that can be invoked within a workflow to fulfill a certain activity. A WFMS has to administer applications and the details of their invocation.

Other aspects: Other aspects of workflows include constraints that have to be satisfied (e.g., pre- and postcondition of activities, quality-of-service constraints), in particular temporal constraints specifying deadlines and bounds for the duration between activities. Business rules may restrict the process.

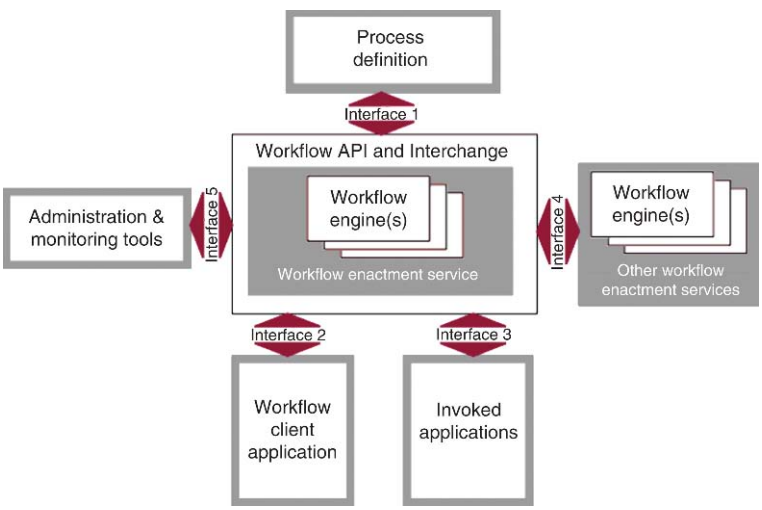
Workflow Management Systems

Workflow Management Systems are generic software systems that are driven by workflow models and control the execution of workflows.

The Workflow Management Coalition (WfMC) developed a reference architecture for WFMS that has been widely adopted. Figure 1 shows the model with its components and interfaces.

The Workflow Enactment Service, consisting of one or several workflow engines, is the core of a workflow management system. It accepts a process description, interprets it, creates instances (workflows), and controls the execution of the process instances. This means that the activities of the workflows are invoked according to the logic expressed in the process model, activities are dispatched to actors, and, if necessary, external services and external applications are invoked or process instances are forwarded to other workflow enactment services.

Interface 1 accepts the process definition that contains all information necessary for the execution of process instances. Process definition tools are employed to describe processes in a proper format. Some process definition languages have been proposed as standards (e.g., XPDL or BPEL) to have uniform interfaces between process definition tools and workflow enactment services. Nevertheless, many workflow management systems feature proprietary languages, partly because of the different types of workflows they intend to support.



Workflow Management and Workflow Management System. Figure 1. Workflow Management Systems: WfMC reference architecture [2].

Interface 2 links the workflow enactment service with workflow client applications, the piece of software the workflow participants (or actors) work with. The main feature of this component is to maintain the worklist, a collection of all activities assigned to a particular user. Users can typically accept an activity, delegate it, refuse it, work on it, or complete it.

Interface 3 describes how other applications are invoked. This interface is frequently used to invoke legacy applications. Since applications (in particular interactive programs) can be invoked from the workflow client application as well, the interfaces 2 and 3 are typically combined permitting the invocation of applications both directly from the workflow enactment service and on user request from the workflow client tool.

Interface 4 addresses workflow interoperability. It defines the interaction between workflow enactment services, the invocation of subworkflows, the synchronization between cooperating workflows, and the forwarding of workflows to other workflow applications.

Interface 5 specifies the interaction with administration and monitoring tools. Administration tools provide the general administration of the workflow system: the registration of user, of application programs, the security management, in particular the administration of rights. The monitoring tools allow for the inspection of the progress of workflow instances and typically provide sophisticated statistics about the execution of workflows to support process managers and to provide data for process improvement.

Key Applications

Workflow systems are mainly employed to support the management and execution of business processes. The typical architecture is that a workflow management system is used for managing the business processes and for integrating the different application areas. Such an architecture can be found in various enterprises and organizations. A few example applications include insurance claim processing, loan processing, product development, bug reporting, order processing, media production, etc.

However, there are additional types of workflow applications:

- Groupware like systems with high flexibility to support workflow that cannot be fully modeled beforehand but develop as the process is executed. These systems primarily support the cooperation between individuals and care for highly dynamic processes.
- There are workflow systems as top layers on large enterprise software systems, adding processes to the rich integrated functionality of these systems.
- Workflow systems are used to integrate heterogeneous application systems and provide an integration platform for these systems. Workflow systems are a layer in the architecture of software systems caring for the process aspects just as data management systems are responsible for storing and retrieving data (process brokers).

Workflows are frequently classified into three different groups according to the criteria of how much they can be automated, how frequent they are executed, and how much they can be specified:

- *Ad-hoc workflows* are highly dynamic workflows that are not fully specified at build time and consist mainly of manual activities. Examples are document routing, or review processes.
- *Administrative workflows* are well structured and more repetitive. They are the typical office paper shuffling type of processes. Examples are budgeting, purchase order processing, etc.
- *Production workflows* are highly repetitive, well structured and fully automated. They represent the core business processes of an organization. Examples are insurance claim handling, bug reporting, order processing, etc.

The employment of workflow technology is supposed to bring the following benefits:

- Efficiency: The automation of process control, the increased possibilities for concurrent execution of activities of the same workflow instance, and the automation of the forwarding of work to the next actor increase the efficiency of business processes as measured in throughput and in turn-around time.
- Process modeling: The business processes are explicitly specified and do no longer solely exist in the heads of the participants or in manuals. This is a necessary prerequisite to reflect on processes and to improve processes.
- Automated process control: the centralized automatic control of the process execution guarantees that business rules are respected. Deviations from the specified process, which are of course possible, are documented and can be used for auditing and

process improvement. Users are informed of work to do, and the data used for performing activities are automatically delivered to the respective workers.

- Integration: Workflow systems allow to bridge heterogeneous applications. Functionalities in application systems are called external applications. The workflow describes and the WFMS enforces the proper sequences of invocation of functionalities in different application systems and transports data between different systems. In general, the integrity of the whole system consisting of heterogeneous applications is achieved.
- Transparency: Transparency is increased through explicit process specification and documentation. All steps and decisions in the execution of a process are centrally documented (workflow log or workflow history). Therefore, it is easy to report the actual state of a process instance.
- Monitoring: The progress of workflows can be easily monitored. For example, delays or exceptional situations are immediately recognized allowing process managers to react timely to assure that deadlines are met.
- Documentation: The execution of workflows is automatically documented. Each process can be traced, e.g., for the purpose of revision. The process documentation is also a valuable source for statistics about the processes to identify weaknesses of the process definition and for process improvements.
- Quality assurance: Quality assurance procedures like the ISO 9000 Suite or TQM (Total Quality Management) require in particular that processes are specified, that processes are executed according to their specification, and that all processes are documented. All that is supported by workflow systems.
- Process oriented management: Workflow systems support the management of process driven organization.

Workflow systems are a flourishing market. Applications can be found in a large number of application domains: e.g., insurances, banks, government agencies, hospitals, etc. Scientific workflows are successfully applied in life science, physics, and chemistry, etc. There are many applications with embedded workflow technology, e.g., document management systems, content management systems, or ERP-systems.

Future Directions

An important development of workflow systems is their combination with web service technology. Web service composition relies on the same principles as workflow, the main difference is that the activities are web services.

Another important topic is the support of interorganizational business processes. The benefits of workflow technology should be exploited also for business processes that span several organizations (enterprizes). While this is already achieved for some types of virtual enterprizes, where the partners (suppliers, costumers, consultants, service providers, etc.) involved and the interfaces are fairly stable, it is still a challenge to support processes with changing partners or multitudes of partners. Here a lot of research is still needed, in particular, integration with semantic web services.

Cross-references

- ▶ Database Techniques to Improve Scientific Simulations
- ▶ Extended Transaction Models
- ▶ Workflow Management Coalition
- ▶ Workflow Model
- ▶ Workflow Transactions

Recommended Reading

1. van der Aalst W. et al. Workflow Management: models, methods, and systems. MIT, Cambridge, MA, 2002.
2. Dumas M., van der Aalst W.M., and ter Hofstede A.H. Process-aware information systems: bridging people and software through process technology. Wiley, 2005.
3. Eder J., Groiss H., and Liebhart W. The Workflow Management System Panta Rhei. In Workflow Management Systems and Interoperability. Kalinichenko, A. Dogac, M.T. Özsu, A. Sheth (eds). 1998. pp. 129–144.
4. Ellis C.A. and Nutt G.J. Office Information Systems and Computer Science. ACM Comput. Surv., 12(1): 27–60, 1980.
5. Ellis C.A. and Nutt G.J. Modeling and Enactment of Workflow Systems. In Proc. 14th Int. Conf. on Application and Theory of Petri Nets., 1993, pp. 1–16.
6. Georgakopoulos D., Hornick M., and Sheth A. An overview of workflow management: From process modeling to workflow automation infrastructure. Distrib. Parallel Databases, 3(2):119–153, 1995.
7. Jablonski S. and Bussler C. Workflow management: modeling concepts, architecture and implementation. Int. Thomson Computer, 1996.
8. Leymann F. and Roller D. Production workflow. Prentice-Hall, Englewood, Cliffs, NJ, 2000.
9. The Workflow Reference Model. Workflow Management Coalition, 1995.
10. Workflow management coalition terminology and glossary [R]. Workflow Management Coalition, 1999.

Workflow Management Coalition

Nathaniel Palmer
Workflow Management Coalition, Hingham, MA, USA

Synonyms
XPDL

Definition
The Workflow Management Coalition, founded in August 1993, is a non-profit, international organization of workflow vendors, users, analysts and university/research groups. The Coalition's mission is to promote and develop the use of workflow through the establishment of standards for software terminology, interoperability and connectivity among BPM and workflow products. Comprising more than 250 members worldwide, the Coalition is the primary standards body for this software market.

Key Points
The WfMC creates and contributes to process related standards, educates the market on related issues, and is the only standards organization that concentrates purely on process. The WfMC created Wf-XML and XPDL, the leading process definition used today in over 70 solutions to store and exchange process models. XPDL is not an executable programming language but a process design format for storing the visual diagram and process syntax of business process models, as well as extended product attributes.

The Coalition has developed a framework for the establishment of workflow standards. This framework includes five categories of interoperability and communication standards that will allow multiple workflow products to coexist and interoperate within a user's environment. These interface points are structured around the workflow reference model which provides the framework for the Coalition's standards program. The reference model identifies the common characteristics of workflow systems and defines five discrete functional interfaces through which a workflow management system interacts with its environment – users, computer tools and applications, other software services, etc. Working groups meet individually, and also under the umbrella of the Technical Committee, which is responsible for overall technical direction and coordination.

Cross-references
▶ Workflow Management and Workflow Management System
▶ Workflow Model
▶ XML Process Definition Language

Workflow Meta-Model

▶ Workflow Schema

Workflow Mining

▶ Process Mining

Workflow Model

Nathaniel Palmer
Workflow Management Coalition, Hingham, MA, USA

Synonyms
Business process model; Process definition

Definition
A set of one or more linked procedures or activities which collectively realize a business objective or policy goal, normally within the context of an organizational structure defining functional roles and relationships.

Key Points
- A business process is typically associated with operational objectives and business relationships, for example an Insurance Claims Process, or Engineering
- Development Process. A process may be wholly contained within a single organizational unit or may span several different organizations, such as in a customer-supplier relationship.
- A business process has defined conditions triggering its initiation in each new instance (e.g., the arrival of a claim) and defined outputs at its completion.
- A business process may involve formal or relatively informal interactions between participants; its duration may also vary widely.

- A business process may consist of automated activities, capable of workflow management, and/or manual activities, which lie outside the scope of workflow management.

Cross-references

► Workflow Management and Workflow Management System
► Workflow Modeling

Workflow Model Analysis

W. M. P. van der Aalst
Eindhoven University of Technology, Eindhoven, The Netherlands

Definition

The correctness, effectiveness, and efficiency of the business processes supported by a workflow management systems are vital to the organization. A workflow process definition which contains errors may lead to angry customers, back-log, damage claims, and loss of goodwill. Flaws in the design of a workflow definition may also lead to high throughput times, low service levels, and a need for excess capacity. This is why it is important to *analyze* a workflow process definition before it is put into production.

In order to analyse a workflow definition, it is necessary to translate the definition into a model suitable for analysis, e.g., a simulation model or a model that allows for verification techniques (e.g., a Petri net).

Key Points

Process-aware information systems are driven by process models. A typical example of a process-aware information system is a system generated using workflow management software. In such cases there is an explicit process model that can be used as a starting point for analysis [3,4].

There are three types of workflow model analysis:

1. *Validation*, i.e., testing whether the workflow behaves as expected.
2. *Verification*, i.e., establishing the correctness of a workflow.
3. *Performance analysis*, i.e., evaluating the ability to meet requirements with respect to throughput times, service levels, and resource utilization.

Validation can be done by interactive simulation: a number of fictitious cases are fed to the system to see whether they are handled well. For verification and performance analysis more advanced analysis techniques are needed. Fortunately, many powerful analysis techniques have been developed for formal methods such as Petri nets [2,3]. Linear algebraic techniques can be used to verify many properties, e.g., place invariants, transition invariants, and (non-)reachability. Coverability graph analysis, model checking, and reduction techniques can be used to analyze the dynamic behavior of the workflow process. Simulation, queueing networks, and Markov-chain analysis can be used for performance evaluation (cf. [1,2]).

Note that all three types of analysis assume that there is some initial workflow model. This implies that analysis may require some modeling or some translation from one language to another. In some cases, *process mining* can be used to discover the underlying workflow model.

Cross-references

► Business Process Management
► Petri Nets
► Process Mining
► Workflow Management

Recommended Reading

1. Marsan M.A., Balbo G., and Conte G. et al. Modelling with generalized stochastic Petri nets. Wiley Series in Parallel Computing. Wiley, New York, NY, USA, 1995.
2. Reisig W. and Rozenberg G. (eds.) Lectures on Petri Nets I: Basic Models. LNCS. 1491. Springer, Berlin Heidelberg New York, 1998.
3. van der Aalst W.M.P. The application of Petri nets to workflow management. J. Circ. Syst. Comput., 8(1):21–66, 1998.
4. van der Aalst W.M.P. and van Hee K.M. Workflow Management: Models, Methods, and Systems. MIT, Cambridge, MA, 2004.

Workflow Modeling

Marlon Dumas
University of Tartu, Tartu, Estonia

Synonyms

Business process modeling

Definition

A workflow model is a representation of the way an organization operates to achieve a goal, such as

delivering a product or a service. Workflow models may be given as input to a workflow management system to automatically coordinate the tasks composing the workflow model. However, workflow modeling may be conducted purely for documentation purposes or to analyze and improve the operations of an organization, without this improvement effort implying automation by means of a workflow system.

A typical workflow model is a graph consisting of at least two types of nodes: task nodes and control nodes. Task nodes describe units of work that may be performed by humans or software applications, or a combination thereof. Control nodes capture the flow of execution between tasks, therefore establishing which tasks should be enabled or performed after completion of a given task. Workflow models, especially when they are intended for automation, may also include object nodes denoting inputs and outputs of tasks. Object nodes may correspond to data required or produced by a task. But in some notations, they may correspond to physical documents or other artifacts that need to be made available or that are produced or modified by a task. Additionally, workflow models may include elements for capturing resources that are involved in the performance of tasks. For example, in two notations for workflow modeling, the Unified Modeling Language (UML) Activity Diagrams, and the Business Process Modeling Notation (BPMN), resource types are captured as *swimlanes*, with each task belonging to one swimlane (or multiple in the case of UML). In other notations, such as Event-driven Process Chains (EPC), this is represented by attaching resource types to each task.

Historical Background

The idea of documenting business processes to improve customer satisfaction and business operations dates back to at least the 1960s when it was raised, among others, by Levitt [8]. Levitt contended that organizations should avoid focusing exclusively on goods manufacturing, and should view their "entire business process as consisting of a tightly integrated effort to discover, create, arouse, and satisfy customer needs." However, it is not until the 1970s that the idea of using computer systems to coordinate business operations based on process models emerged. Early scientific work on business process modeling was undertaken in the context of *office information systems* by Zisman [13] and Ellis [2] among others. During the 1970s and 1980s there was optimism in the IT community regarding the applicability of process-oriented office information systems. Unfortunately, few deployments of this concept succeeded, partly due to the lack of maturity of the technology, but also due to the fact that the structure and operations of organizations were centered around the fulfillment of individual functions rather than end-to-end processes. Following these early negative experiences, the idea of process modeling lost ground during the next decade.

Towards the early 1990s, however, there was a renewed interest in business process modeling. Instrumental in this revival was the popularity gained in the management community by the concept of *Business Process Reengineering* (BPR) advocated by Hammer [3] and Davenport [1] among others. BPR contended that overspecialized tasks carried across multiple organizational units should be unified into coherent and globally visible processes. This management trend fed another wave of business process technology known as workflow management. Workflow management, in its original form, focused on capturing business processes composed of tasks performed by human actors and requiring data, documents and forms to be transferred between them. Workflow management was successfully applied to the automation of routine and high-volume administrative processes such as order-to-cash, procure-to-pay and insurance claim handling. However it was less successful in application scenarios requiring the integration of heterogeneous information systems, and those involving high levels of evolution and change. Also, standardization efforts in the field of workflow modeling and management, led by the Workflow Management Coalition (WfMC), failed to gain wide adoption.

The BPR trend also led to the emergence of business process modeling tools that support the analysis and simulation of business processes, without targeting their automation. The ARIS platform, based on the EPC notation, is an example of this family of tools. Other competing tools supported alternative notations such as IDEF3 and several variants of flowcharts.

By the late 1990s, the management community had moved from the concept of BPR to that of *Business Process Improvement* (BPI), which advocates an incremental and continuous approach to adopting process-orientation. This change of trend combined with the limited success of workflow management and its failure to reach standardization, led to the

term *workflow* acquiring a negative connotation. Nonetheless, the key concepts underpinning workflow management reemerged at the wake of the twenty-first century under the umbrella of *Business Process Management* (BPM). BPM adopts a more holistic view than workflow management, covering not only the automated coordination of processes, but also their monitoring and continuous analysis and improvement. Also, BPM does not only deal with the coordination of human tasks, but also the integration of heterogeneous information and software systems.

With BPM came the realization that business processes should be analyzed and designed both from a business viewpoint and from a technical viewpoint, and that methods are required to bridge these viewpoints. Also, the BPM trend led to a convergence of modeling languages. BPMN has emerged as a standard for process modeling at the analysis level, while at the implementation level, the place is taken by WS-BPEL. Other notations include UML Activity Diagrams and EPCs at the analysis level, and YAWL (Yet Another Workflow Language) [6] at the execution level.

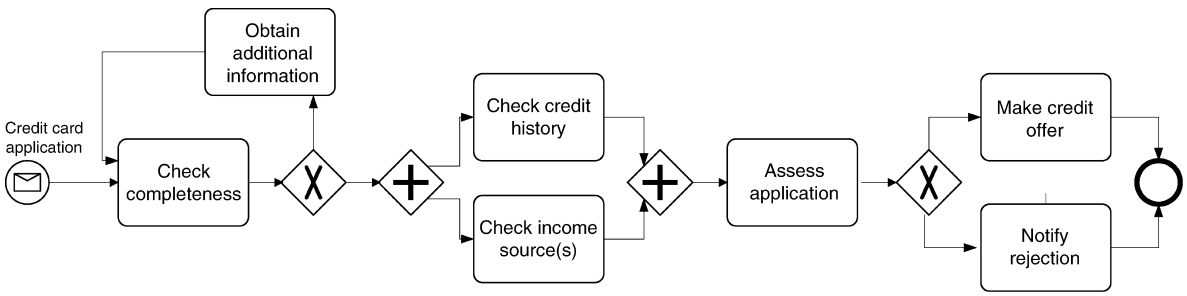
Foundations

Workflow modeling can be approached from a number of perspectives [4]. The *control-flow perspective* describes the ordering and causality relationships between tasks. The *data perspective* (or information perspective) describes the data that are taken as input and produced by the activities in the process. The *resource perspective* describes the structure of the organization and identifies resources, roles, and groups. The *task perspective* describes individual steps in the processes and thus connects the other three perspectives.

Many workflow modeling notations emphasize the control-flow perspective. This is the case for example of process modeling notation based on *flowcharts*. Flowcharts consist of actions (also called activities) and decision nodes connected by arcs that denote sequential execution. In their basic form, flowcharts do not make a separation between actions performed by different actors or resources. However, an extension of flowcharts, known as cross-functional flowcharts, allow one to perform this partitioning. Flowcharts have inspired many other contemporary process modeling notations, including UML Activity Diagrams and BPMN.

A simplified business process model for credit approval in BPMN is given in Fig. 1. An execution of this process model is triggered by the receipt of a credit application, denoted by the leftmost element in the model. Following this, the application is checked for completeness. Two outcomes are possible: either the application is marked as complete or as incomplete. This choice is represented by the *decision gateway* depicted by the diamond labeled by an "X" sign. If the application is incomplete, additional information is sought from the applicant. Otherwise, two checks are performed in parallel: a credit history check and an income check. This is denoted by a *parallel split gateway* (the first diamond labeled by a "+" sign). The two parallel checks then converge into a *synchronization gateway* (the second diamond labeled by a "+"). The credit history check may be performed automatically while the income source check may require an officer to contact the applicant's employer by phone. After these checks, the application is assessed and it is either accepted or rejected.

A second family of notations for business process modeling are those based on state machines. State machines provide a simple approach to modeling behavior. In their basic form, they consists of states connected by transitions. When used for business process



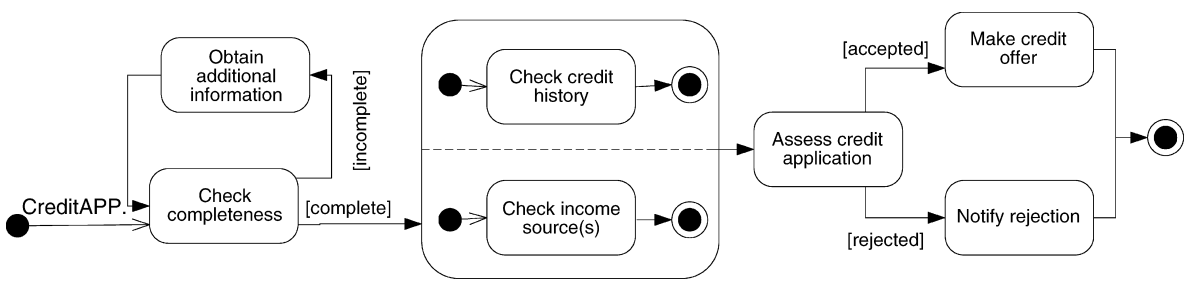
Workflow Modeling. Figure 1. Example of a business process model in BPMN.

modeling, it is common for the transitions in a state machine to be labeled by event-condition-action (ECA) rules. The semantics of a transition labeled by an action is the following: if the execution of the state machine is in the source state, an occurrence of the event (type) has been observed, and the condition holds, then the transition may be taken. If the transition is taken, the action associated with the transition is performed and the execution moves to the target state. A transition may contain any combination of these three elements: e.g., only an event, only a condition, only an action, an event and a condition but no action, etc. In some variants of state machines, it is possible to attach activities to any given state. The semantics is that when the state is entered, the activity in question may be started, and the state can only be exited once this activity has completed. Several commercial business process management systems support the execution of process models defined as state machines. Examples include the "Business State Machine" models supported by IBM Websphere and the "Workflow State Machine" models supported by Microsoft Windows Workflow Foundation.

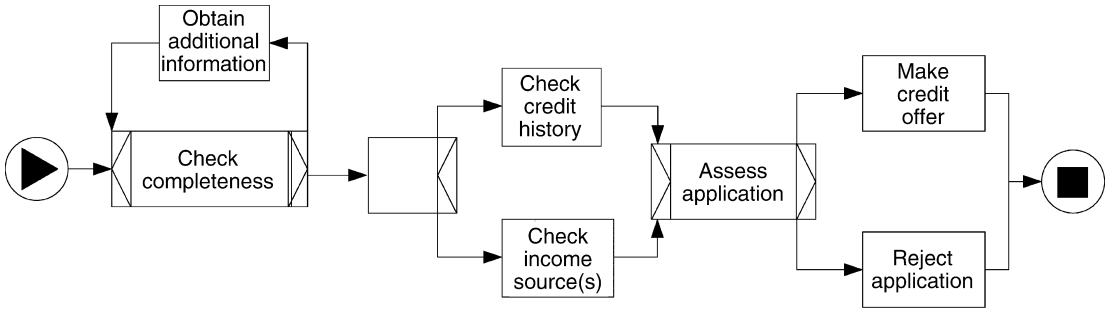
A disadvantage of using basic state machines for process modeling is their tendency to lead to state explosion. Basic state machines represent sequential behavior: the state machine can only be in one state at a time. This, when representing a business process model with parallel execution threads, one essentially needs to enumerate all possible permutations of the activities that may be executed concurrently. Another source of state explosion is exception handling. Statecharts have been proposed as a way to address this state explosion problem. Statecharts include a notion of concurrent components as well as transitions that may interrupt the execution of an entire component of the model. The potential use of statecharts for workflow modeling has been studied in the research literature, for example in the context of the Mentor research prototype [10], although it has not been adopted in commercial products.

Figure 2 shows a statechart process model intended to be equivalent to the BPMN "Credit Approval" process model of Fig. 1. It can be noted that decision points are represented by multiple transitions sharing the same source state and labeled with different conditions (conditions are written between square brackets). Parallel execution is represented by means of a composite state divided into two concurrent components. Each of these components contains one single state. In the general case, each concurrent component may contain an arbitrarily complex state machine.

Finally, a third family of notations for control-flow modeling of business processes are those based on *Petri nets*. In the research literature, workflow modeling notations based on Petri nets have been used mainly for studying expressiveness and verification problems (e.g., detecting deadlocks in workflow models). Also, certain Petri net-based notations are supported by business process modeling tools and workflow management systems. A notable example is YAWL. YAWL is an extension of Petri nets designed on the basis of the *Workflow Patterns*. A key design goal of YAWL is to support as many workflow patterns as possible in a direct manner, while retaining the theoretical foundation of Petri nets. Like Petri nets, YAWL is based on the concepts of place and transition (Transitions are called *tasks* in YAWL while places are called *conditions*.). YAWL also incorporates a concept of *decorator*, which is akin to the concept of gateway in BPMN. In addition, it supports the concepts of sub-process (called *composite task*) and cancelation region. The latter is used to capture the fact that the execution of a set of tasks may be interrupted by the execution of another task, and can be used for example for fault handling. Figure 3 shows a YAWL process model intended to be equivalent to the BPMN "Credit Approval" process



Workflow Modeling. Figure 2. Example of a business process model captured as a statechart.



Workflow Modeling. Figure 3. Example of a business process model captured in YAWL.

model of Fig. 1 and the statechart process model of Fig. 2. The symbols attached to the left and the right of each task are the decorators. The decorator on the left of task "Check Completeness" is an XOR-merge decorator, meaning that the task can be reached from either of its incoming flows. The decorator of the right of this same task is an XOR-split, which corresponds to the decision gateway in BPMN. The decorator just before "Check Credit History" is an AND-split (akin to a parallel split gateway) and the one attached to the left of "Assess Application" is an AND-join and it is used to denote a full synchronization.

The data perspective of process modeling can be captured using notations derived from dataflow diagrams. Dataflow diagrams are directed graphs composed of three types of nodes: *processes*, *entities* and *data stores*. Entities denote actors, whether internal or external to an organization. Processes denote units of work that transform data. Data stores denote units of data storage. A data store is typically used to store data objects of a designated type. The arcs in a dataflow diagram are called *data flows*. Data flows denote the transfer of data between entities, processes and data stores. They may be labeled with the type of data being transferred.

Data flow diagrams are often used very liberally, with almost no universally accepted syntactic constraints. However, best practice dictates that each process should have at least one incoming data flow and one outgoing data flow. Data flows may connect entities to processes and vice-versa, but can not be used to connect entities to data stores or to connect one entity to another directly. It is possible to connect two processes directly by means of a data flow, in which case it means that the output of the first process is the input of the second. It is also possible to assign ordinal numbers to processes in order to capture their execution order. Dataflow diagrams also support decomposition: Any process can be decomposed into an entire dataflow diagram consisting of several subprocesses.

Data flows are quite restricted insofar as they do not capture decision points and they are not suitable for capturing parallel execution or synchronization. Because of this, dataflow diagrams are not used to capture processes in details, instead, they are used as a high-level notation.

On the other hand, some elements from data flow diagrams have found their way into other process modeling notations such as EPCs, which incorporate constructs for capturing input and output data of functions. Similarly, Activity Diagrams have a notion of object nodes, used to represent inputs (outputs) consumed (produced) by activities. Also, more sophisticated variants of dataflow languages have been adopted in the field of *scientific workflow*, which is concerned by the automation of processes involving large datasets and complex computational steps in fields such as biology, astrophysics and environmental sciences.

When modeling business processes for execution (e.g., by a workflow system), it is important to capture not only the flow of data into and out of tasks, but also data transformation, extraction and aggregation steps. At this level of detail, it may become cumbersome to use simple data flows. Instead, executable process modeling languages tend to rely on *scoped variables*. Variables may be defined for example at the level of the process model. Tasks may take as input several input and output parameters, and data mappings are used to connect the data available in the process model variables, and the input and output parameters. Data mappings may be inbound (i.e., expressions that extract data from variables in the task's encompassing

scope and assign values to the task's input parameter) or outbound (from the output parameters to the variables in the encompassing scope). This is the approach used for example in YAWL, Mentor and AdeptFlex [9]. It is also the approach adopted by many commercial workflow products and by WS-BPEL.

One area of research in the area of workflow modeling is the analysis of their expressiveness [6]. Several classes of workflow notations have been identified from this perspective: structured workflow models, synchronizing workflow models, and standard workflow models. Structured workflow models are those that can be composed out of four basic control-flow constructs, one for choosing one among multiple blocks of activities, another for executing multiple blocks of activities in parallel, a third one for executing multiple blocks of activities in sequence, and finally a fourth one for repeating a block of activities multiple times. An activity by itself can be a block, and more complex blocks can be formed by composing smaller blocks using one of the four constructs. It has been shown that structured workflow models are well-behaved, meaning in particular that their execution can not result in deadlocks [5]. On the other hand, this class of models has low expressive power. It is not difficult to construct examples of unstructured workflow models that can not be translated to equivalent structured ones under reasonable notions of equivalence. Synchronizing workflow models are acyclic workflow models composed of activities that are connected by transitions. These transitions correspond to control links in the WS-BPEL terminology. They impose precedence relations and they can be labeled with conditions. An activity can be executed if all its predecessors have been either completed or skipped. Like structured workflow models, this class of models are well-behaved. However, they are not very expressive. Finally, standard workflow models correspond to the subset of BPMN composed of tasks, decision gateways, merge gateways (where multiple alternative branches converge), parallel split gateways and synchronization gateways. This class of models is more expressive than the former two, but on the other hand, some of these models may contain deadlocks. Other classes of workflow models are those corresponding to Workflow Nets, which are more expressive than standard workflow models as they can capture different types of choices. YAWL models are even more expressive due to the presence of the cancelation region construct, and another construct (called the "OR-join") for synchronization of parallel branches, where some of these branches may never complete. A significant amount of literature in the area of workflow verification has dealt with studying properties of modeling languages incorporating a notion of OR-join [7].

Key Applications

Business process modeling can be applied both for documentation, business analysis, business process improvement and automation. In some cases, organizations undertake business process modeling projects for the sake of documenting key activities and using this documentation for employee induction and knowledge transfer. business process models are sometimes required for compliance reasons, for example as a result of legislation in the style of the Sarbannes-Oxley act in the United States which requires public companies to have internal control processes in place to detect and prevent frauds. A process model that captures how an organization currently works, is called an "as is" process model. Such "as is" models are useful in understanding bottlenecks and generating ideas for improvement using process analysis techniques, including simulation and activity-based costing. The integration of improvement ideas in a business process leads to a "to be" process model. Some of these improvement ideas may include automating (parts of) the business process. At this stage, business process models need to be made more detailed. Once they have been made executable, business process models may be deployed in a business process execution engine (also called a workflow engine).

Future Directions

Standardization efforts in the field of business process modeling have led to the current co-existence of two standard process modeling languages: BPMN at the business analysis level and BPEL at the implementation and execution level. One question that arises from the co-existence of these two standards is whether or not models in BPMN should be synchronized with models defined in BPEL and how. The assumption behind this question is that business analysts will define models in BPMN, while developers are likely to work at the level of BPEL. For facilitate the collaboration between these stakeholders, it would be desirable to relate models defined in these two languages. An alternative would be for the two languages to converge into a single one.

Another related area where open problems exist is that of scientific workflow modeling. Researchers have argued for some time that workflow-related problems in scientific fields such as genetics and ecological sciences, do not have the same requirements as those found in the automation of business processes. These differences in requirements call for new paradigms for workflow modeling. While several proposals have been put in place [9], e.g., workflow modeling languages based on variants of dataflow modeling, consolidation still needs to occur and consensus in the area is yet to be reached.

Cross-references

- ▶ Activity Diagrams
- ▶ Business Process Execution Language
- ▶ Business Process Management
- ▶ Business Process Modeling Notation
- ▶ Business Process Reengineering
- ▶ Petri Nets
- ▶ Workflow Model Analysis
- ▶ Workflow Patterns

Recommended Reading

1. Davenport T.H. Process Innovation: Reengineering Work through Information Technology. Harvard Business School, Boston, 1992.
2. Ellis C.A. Information control nets: a mathematical model of office information flow. In Proc. Conf. on Simulation, Measurement and Modeling of Computer Systems. 1979, pp. 225–240.
3. Hammer M. Reengineering work: don't automate, obliterate. Harvard Bus. Rev., July/August 1990, pp. 104–112.
4. Jablonski S. and Bussler C. Workflow Management: Modeling Concepts, Architecture, and Implementation. Int. Thomson Computer, London, UK, 1996.
5. Kiepuszewski B., ter Hofstede A.H.M., and Bussler C. On structured workflow modelling. In Proc. 12th Int. Conf. on Advanced Information Systems Engineering, 2000, pp. 431–445.
6. Kiepuszewski B., ter Hofstede A.H.M., and van der Aalst W.M.P. Fundamentals of control flow in workflows. Acta Inform., 39 (3):143–209, 2003.
7. Kindler E. On the semantics of epcs: resolving the vicious circle. Data Knowl. Eng., 56(1):23–40, 2006.
8. Levitt T. Marketing myopia. Harvard Bus. Rev., July/August 1960, pp. 45–56.
9. Ludäscher B., Altintas I., Berkley C., Higgins D., Jaeger E., Jones M., Lee E.A., Tao J., and Zhao Y. Scientific workflow management and the Kepler system. Concurrency Comput.–Pract. Exp., 18(10):1039–1065, 2006.
10. Muth P., Wodtke D., Weissenfels J., Dittrich A., and Weikum G. From centralized workflow specification to distributed workflow execution. J. Intell. Inform. Syst., 10(2), 1998.
11. Reichert M. and Dadam P. ADEPTflex: supporting dynamic changes of workflow without loosing control. J. Intell. Inform. Syst., 10(2):93–129, 1998.
12. van der Aalst W.M.P. and ter Hofstede A.H.M. YAWL: yet another workflow language. Inform. Syst., 30(4):245–275, 2004.
13. Zisman M.D. Representation, Specification and Automation of Office Procedures. PhD thesis, University of Pennsylvania, Warton School of Business, 1977.

Workflow on Grid

▶ Grid and Workflows

Workflow Participant

▶ Actors/Agents/Roles

Workflow Patterns

W. M. P. van der Aalst
Eindhoven University of Technology, Eindhoven, The Netherlands

Definition

Differences in features supported by the various contemporary commercial workflow management systems point to different insights of *suitability* and different levels of *expressive power*. One way to characterize these differences and to support modelers in designing non-trivial workflows, is to use a patterns-based approach. Requirements for workflow languages can be indicated through workflow *patterns*, i.e., frequently recurring structures in processes that need support from some process-aware information system. The Workflow Patterns Initiative [6] aims at the systematic identification of workflow requirements in the form of patterns.

The Workflow Patterns Initiative resulted in various collections of patterns. The initial twenty control-flow patterns have been extended with additional control-flow patterns and patterns for other perspectives such as the resource, data, and exception handling perspectives. All of these patterns have been used to evaluate different systems in a truly objective manner.

Key Points

The Workflow Patterns Initiative [6] was established with the aim of delineating the fundamental requirements

that arise during business process modeling on a recurring basis and describe them in an imperative way. Based on an analysis of contemporary workflow products and modeling problems encountered in various workflow projects, a set of twenty patterns describing the control-flow perspective of workflow systems was identified [1]. Since their release, these patterns have been widely used by practitioners, vendors and academics alike in the selection, design and development of workflow systems.

An example of one of the initial workflow patterns is the *Discriminator* pattern which refers to the ability "to depict the convergence of two or more branches such that the first activation of an incoming branch results in the subsequent activity being triggered and subsequent activations of remaining incoming branches are ignored." It can be seen as a special case of the more general *N-out-of-M* pattern, where N is equal to one. While the more recent languages and systems support this pattern, many of the traditional workflow management systems have severe problems supporting this pattern. As a result, the designer is forced to change the process, to work around the system, or to create "spaghetti diagrams."

Based on many practical applications the original set of twenty control-flow patterns was extended in two directions. First of all, the set of control-flow patterns was extended and refined. There are now more than forty control-flow patterns, all formally described in terms of Petri nets. Second, patterns for other perspectives were added. The data perspective [5] deals with the passing of information, scoping of variables, etc, while the resource perspective [4] deals with resource to task allocation, delegation, etc. More than eighty patterns have been defined for both perspectives. The patterns for the exception handling perspective deal with the various causes of exceptions and the various actions that need to be taken as a result of exceptions occurring.

The resulting sets of patterns have been used for: (i) improving the understanding of workflow management and clarifying the semantics of different constructs, (ii) analyzing the requirements of workflow projects, (iii) evaluating products, and (iv) the training of workflow designers.

Cross-references

- ▶ Workflow Constructs
- ▶ Workflow Management
- ▶ Workflow Schema

Recommended Reading

1. van der Aalst W.M.P., ter Hofstede A.H.M., Kiepuszewski B., and Barros A.P. Workflow patterns. Distrib. Parallel Database, 14(1):5–51, 2003.
2. Casati F., Castano S., Fugini M.G., Mirbel I., and Pernici B. Using patterns to design rules in workflows. IEEE Trans. Softw. Eng., 26(8):760–785, 2000.
3. Hohpe G. and Woolf B. Enterprise Integration Patterns. Addison-Wesley, Reading, MA, 2003.
4. Russell N., van der Aalst W.M.P., ter Hofstede A.H.M., and Edmond D. Workflow resource patterns: identification, representation and tool support. In Proc. 17th Int. Conf. on Advanced Information Systems Eng., 2005, pp. 216–232.
5. Russell N., ter Hofstede A.H.M., Edmond D., and van der Aalst W.M.P. Workflow data patterns: identification, representation and tool support. In Proc. 24th Int. Conf. on Conceptual Modeling, 2005, pp. 353–368.
6. Workflow Patterns Home Page. http://www.workflowpatterns.com.

Workflow Scheduler

▶ Scheduler

Workflow Schema

Nathaniel Palmer
Workflow Management Coalition, Hingham, MA, USA

Synonyms

Workflow Meta-Model

Definition

The meta structure of a business process defined in terms of a data model.

Key Points

In contemporary workflow the Workflow Schema is represented by a standard process definition such as XPDL (XML Process Definition Language) based on XML Schema. In non-standard environments or workflows defined with in specific application (not a BPMS or workflow management system) the Workflow Schema is built as a proprietary data model.

Cross-references

- ▶ Workflow Model
- ▶ Workflow Modeling
- ▶ XML Process Definition Language

Workflow Transactions

Vladimir Zadorozhny
University of Pittsburgh, Pittsburgh, PA, USA

Synonyms

Transactional workflows; Transactional processes; Transactional business processes

Definition

Workflow Transaction is a unit of execution within a Workflow Management System (WFMS) that combines process-centric approach to modeling, executing and monitoring of a complex process in an enterprise with data-centric Transaction Management ensuring the correctness and reliability of the workflow applications in the presence of concurrency and failures. Workflow Transactions accommodate application-specific Extended Transaction Models (ETMs) that relax basic ACID properties of a classical database transaction models.

Key Points

Consider a Workflow Management System (WFMS) that models complex processes in an enterprise (e.g., processing insurance claim) using a workflow structured as a set of business tasks that should be performed in a special order. Some of the business tasks can be performed by humans while some of them are implemented as processes that typically create, process and manage information using loosely coupled distributed, autonomous and heterogeneous information systems. This *process-centric* approach to modeling of a complex enterprise process can be contrasted with *data-centric* approach accepted in database Transaction Management (TM) that keeps track of data dependencies, support concurrent data accesses, and crash recovery. Traditional TM approach is too strict for workflow applications, which requires selective use of transactional properties to allow flexible inter-task collaboration within a workflow. For example, classical ACID transaction model enforcing task isolation makes impossible task cooperation. Another example is large amount of task blocking and abortions caused by transaction synchronization, which is not acceptable for large-scale workflows. In order to address those issues Workflow Transactions accommodate application-specific Extended Transaction Models (ETMs) that relax some of the ACID properties.

There are several variations of the workflow transaction concept that depends on the way how the Workflow and Transaction Managements are merged. In general, workflows are more abstract than transactions and transactions provide specific semantics to workflow models. Alternatively, there are approaches where transactions are considered on a higher level of abstraction, while workflows define process structure for the transaction models. Workflows and transaction can also interact as peers at the same abstraction level within a loosely coupled process model. While workflow transaction commonly supports business applications, the workflow technology has also been adopted to support workflows for scientific explorations and discovery.

Cross-references

- ▶ Transaction Management
- ▶ Workflow Management

Recommended Reading

1. Georgakopoulos D., Hornick M., and Sheth A. An overview of workflow management: from process modeling to workflow automation infrastructure. Distrib. Parallel Databases, 3(2):119–153, 1995.
2. Grefen P. and Vonk J. A taxonomy of transactional workflow support. Int. J. Coop. Inf. Syst., 15(1):87–118, 2006.

Workflow/Process Instance Changes

- ▶ Workflow Evolution

World Wide Web Consortium

- ▶ W3C

WORM

- ▶ Storage Devices
- ▶ Storage Security
- ▶ Write Once Read Many

Wrapper

- ▶ Temporal Strata

Wrapper Adaptability

► Wrapper Stability

Wrapper Induction

Max Goebel[1], Michal Ceresna[2]
[1]Vienna University of Technology, Vienna, Austria
[2]Lixto Software GmbH, Vienna, Austria

Synonyms

Wrapper Generation; Information Extraction

Definition

Wrapper induction (or query induction) is a subfield of wrapper generation, which itself belongs to the broader field of information extraction (IE). In IE, wrappers transform unstructured input into structured output formats, and a wrapper generation systems describes the transformation rules involved in such transformations. Wrapper induction is a solution to wrapper generation where transformation rules are learned from examples and counterexamples (inductive learning). The induced wrapper subsequently is applied to unseen input documents to collect further label relations of interest. To ease annotation of examples by the user, the learning framework is often implemented within a visual annotation environment, where the user selects and deselects elements visually.

The term "wrapper induction" was first conceptualized by Nicholas Kushmerick in his influential PhD thesis in 1997 in the context of semi-structured Web documents [10], where the author also laid the theoretic foundations of wrapper induction in the PAC learning framework.

Historical Background

IE is a very mature research field that has spun off the database community in the need to integrate a constantly increasing number of unstructured or semi-structured documents into structured database systems.

The World Wide Web (WWW) as a means of an information sink has been accepted very quickly since its mainstream adoption in the early 1990s. Its low cost and wide spread has resulted in an enormous surge of data to be available on the Web. Particularly in the past decade, the number of documents of high informational value on the Web has exploded. The lack of rigid structure in Web documents together with the Web's vast value has initiated an independent research field for wrapper generation, which studies IE approaches particularly for semi-structured Web documents.

First IE approaches in the Web domain have seen systems where wrappers were constructed manually [13,16], yet it soon showed that manual wrapper systems did not scale well and were inflexible to change in document structure. The urge for more principled, and automated wrapper generation systems sparked the beginning of a whole new research sub-field of wrapper generation.

The beginnings of wrapper induction can be found in [10]. In this original work of Kushmerick, the author identified six wrapper classes and evaluated them in terms of learning theoretical concepts. The conclusion of his work was that quite expressive wrapper systems can be learned from surprisingly simple wrappers. His implementation prototype was called WIEN (1998), and it presents the first wrapper induction system.

Shortly after, SoftMealy and STALKER have been published. The SoftMealy system [7] (1998) relaxed the finite state transducer (Mealy machine) for Web documents, which brought great performance improvements over the WIEN system. The STALKER system [9,14] introduced hierarchical wrappers generated from landmark automata. In [17] (1999), the state machine approach was augmented with observation probabilities to form a Hidden Markov model for header extraction from research papers. BWI [6] (boosted wrapper induction, 2000) extended wrapper induction from its initial application on Web documents back to free text documents. It uses a probabilistic model for tuple length classification together with a boosting algorithm for delimiter learning.

In more recent work, [8] generate node predicates on the DOM tree representation of a HTML document (such as tagName, tagAttribute, #children, #siblings). These are constructed for each example node in the training set together with those nodes along the path towards the root from each such node – until a common ancestor is reached. Verification rules are generated from special verification predicates to be checked later. Finally, patterns with equal extraction results are combined.

In [3], the authors also use the DOM tree representation of web documents to detect repetitive

occurrences of pre/post subtree constellations. They use boolean logic to combine all rules into a minimal disjunctive normal form over all training examples.

Applying supervised learning to Web wrapper generation bears a crucial drawback: sufficiently large sets of documents with annotated examples are hard to obtain and require expertise in the making. Labor and time investments renders the labeling of examples costly, creating a bottleneck in IE from Web documents. As a result, part of the research focused on tools that allow non-expert users to create wrappers interactively via visual user interaction. Starting with a set of non-annotated documents, the algorithm tries to infer the correct wrapper by asking the user to annotate a minimal number of examples.

Lixto [2], NoDoSe [1], Olera [4] are examples of interactive tools that allow users to build wrappers through a GUI. Finally, DEByE [12] (2002) is similar to STALKER and WIEN as it also used landmark automata for rule generation. Unlike all previously mentioned wrapper induction systems that generate extraction rules top-down (most general first), DEByE approaches the problem bottom-up, starting with the most specific extraction rule and generating from there.

Foundations

From Free Text to Web Documents

Wrapper generation systems nowadays focus on the Web domain, where documents are of a semi-structured format. Originally, wrapper generation systems were developed for free text documents. Free text exhibits no implicit structure at all, and extraction requires natural language processing techniques such as *Part-of-Speech tagging*. With the advent of the Web as a content platform, the importance of free text has dwindled in IE with respect to the importance of Web documents. Web documents are an intermediate format, lying between free text (unstructured) and structured text (e.g., XML). The use of HTML tags allows to apply relations between individual text segments in the sense of specifying structural information (see Fig. 1). One genuine problem for Web IE is that the lack of strict rules on these relations yields room for ambiguities of a tag's semantic meaning in different contexts. Web wrapper induction is mainly concerned with the exploitation of tag tree (DOM) structures.

Wrapper Induction

Kushmerick defined the wrapper induction problem as a simple model of information extraction. Input collection is referred to as *set of documents S*. An *extraction rule* (also query) allows to get a subset of the document according to some user-defined filter arguments. The result of a query is a set of information fields from *S*.

Information fields are made up of vectors of unique *attributes* which are referred to as *tuples*. The concept of attributes is borrowed from relational databases, where attributes correspond to table columns in the relational model. In some systems the order of the attributes is relevant, others allow permutations and missing attributes. The collection of all tuples describes the *content* of *S*. Given this definition, a wrapper can be formalized as the function mapping a page to its content.

$$page \Rightarrow wrapper \Rightarrow content$$

Different wrappers define different mappings. In turn, this means that the output of two different wrappers are two different subsets of all tuples defined in the page's content.

Patterns and Output Structure

Wrapping semi-structured resources is equivalent to extracting a hierarchy of elements from a document and returning as result a structured, relational

Price: $39.95

Description:
Title: *Communication sequential processes*
Author: *C.A.R. Hoare*
Year: 1985
Publisher: Prentice-Hall

```
<html><body>
<b>Price:</b> $39.95 <p>

<b> Description: </b> <br>
Title: <i> Communication sequential processes </i></br>
Author: <i> C.A.R. Hoare </i> <br>
Year : 1985 <br>
Publisher: Prentice-Hall <p>
</body> </html>
```

Wrapper Induction. Figure 1. Relational data in free text (*left*) and formatted in semi-structured HTML (*right*).

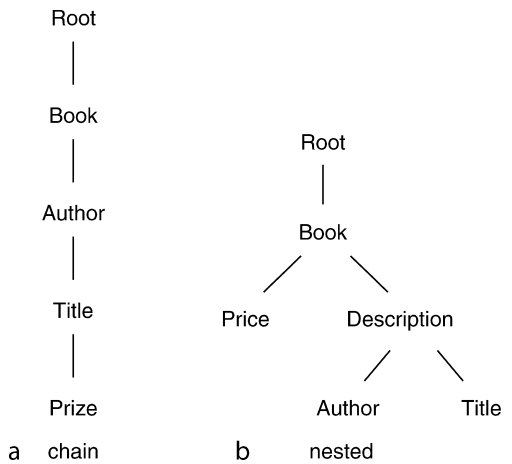
data model that reflects the extracted hierarchy. The hierarchy consists of *patterns* which can be seen as elementary wrapping operators. Each pattern in the hierarchy defines an extraction of one atomic tuple class from a given parent pattern. In the simplest case, data is structured in tabular form. Each tuple can be extracted separately and the attributes of each tuple are arranged in linear chains. A more difficult pattern structure are nested patterns where a parent pattern branches into multiple different child patterns (e.g., similar to a tree).

In Fig. 2, this is illustrated by extracting the pattern *author* from the parent pattern *description*.

Kushmerick's Wrapper Classes

In his PhD thesis, Kushmerick has analyzed delimiter-based wrappers for IE from strings. He devised six wrapper classes of varying complexity that can deal with the extraction of different pattern scenarios by identifying start and stop delimiters. In his formulation, Kushmerick defined the wrapper induction problem as a *Constraint Satisfaction Problem* (CSP) with all possible combinations of left and right delimiters as input. The constraints are formulated on the forbidden combinations on the input, which in turn vary with the wrapper class. What is interesting is that despite their simplicity, these wrapper classes yield decent results on a large benchmark set.

Wrapper Class LR LR (Left–Right) is the simplest wrapper class on which all subsequent classes are build upon. LR wrappers identify one delimiter token on each side of the tuple to be extracted, where L denotes the start token, and R denotes the end token. By taking all possible prefix and postfix strings in the document string for a given example tuple, the learning algorithm first identifies the candidate sets for the left and right delimiters. With L and R being mutually independent, a principled search can be performed via checking the validity of each candidate delimiter for the complete training set. This can also be expressed as constraints on the search space containing of all permutations of L and R delimiters.



Wrapper Induction. Figure 2. A wrapper represented by a hierarchy of patterns. Each pattern is defined by a set of rules describing how to locate its instances from its parent pattern.

Wrapper Class HLRT A more expressive wrapper class is Head-Left–Right-Tail, which is a direct extension from LR. It includes a page's head H and tail T for cases where similar delimiters are used for relevant and irrelevant information fields on a page. Delimiter H determines where to start looking for the LR pattern (the end of the head section), and T respectively determines where the search should end. Together, H and T define the search body of the page. The learning algorithm follows directly from the LR algorithm with the new constraints included. H and T must be learned jointly on l_1.

Wrapper Class OCLR Open–Close-Left–Right is an alternative to the HLRC wrapper class, where the head and tail delimiters are replaced with open and close delimiters. The difference here is that, contrary to the HLRT class where the complete search body for the wrapper is specified by the H and T delimiters, the O and C delimiters merely specify the beginning and end of a tuple in a page. It thus extends LR to a multi-slot extractor. O and C must be learned jointly on l_1.

Wrapper Class HOCLRT Head-Open–Close-Left–Right-Tail is the combination of the HLRT and the OCLR wrapper classes. It is the most powerful of the four basic wrapper classes, yet all four additional delimiters H, T, O and C must be learned jointly on l_1.

Wrapper Class N-LR The previous four wrapper classes work well on tabular (relational) data sets. As it is common to have hierarchical nestings in relational data, Kushmerick devised two extensions to the previous wrapper classes that can deal with nested data. Nested data are tree-like data structures where the attributes are nested hierarchically. The first nested wrapper class, N-LR, extends the simple LR class. It uses the relative position among attributes to determine which attribute to extract next, given a current attribute. Here, all attributes must be learned jointly.

Wrapper Class N-HLRT Finally, wrapper class N-HLRT combines the expressiveness of nested attributes with the head tail constraints from the HLRT class. It is exactly N-LR with the additional delimiters *H* and *T* to be considered in the hypothesis space of all candidate wrappers.

While in practice, Kushmerick showed these wrapper classes to be able to learn the correct wrapper from relatively few examples, the theoretical PAC bounds for the same wrappers are very loose [11].

Other Wrapper classes

Apart from delimiter-based wrapper classes, there also exist other wrapper classes that can prove highly effective, but which depend on the domain of the input documents. For Web wrapper induction, interesting classes include:

1. *Edit distance similarity.* The edit distance (Levensthein Distance) exists both for string and tree structures. It measures the difference between two strings or trees as the number of basic edit operations that need to be performed to turn one argument into the other. Due to the tree structure of the DOM (Document Object Model) of Web pages, this method lends itself very well to identifying tree patterns in Web documents.
2. *Rule-based systems.* Prolog-like rules are induced from examples. Typically, systems generate a most-general set of rules from the given examples and use covering algorithms to reduce the rules to a minimum set covering all examples. The advantage of rule-based wrappers is the relative ease to interpret rules by a user.

Active Learning

The key bottleneck of wrapper induction is the need for labeled training data to be available. An effective solution to reduce the number of annotations in a training corpus is offered by *active learning*. Active learning tries to identify examples that are not labeled but which are informative to the learning process, and the user is asked to label only such informative examples. Active learning has been applied to the Stalker wrapper generation system in [15], using *co-testing*. In co-testing, redundant views of a domain are implemented through mutually-exclusive feature sets per view. Views can be exploited by learning a separate classifier per view, as the ensemble of all classifiers reduces the hypothesis space for the wrapper. ***[5] provides a comprehensive overview of different active learning strategies, including common machine learning techniques such as bagging (multiple partitioning of training data) and boosting (ensemble learning).

Key Applications

Several of the presented wrapper induction systems are known for being successfully commercialized. The Lixto system [2] has emerged into an equally named spin-off company of the Technical University Vienna, Austria (http://www.lixto.com). Kushmerick's work [10] is being integrated into products of a US-based company called QL2 (http://www.ql2.com). Similarly, technologies created by the authors of the STALKER system [14] are used in products of another US company called Fetch Technologies (http://www.fetch.com). Web Intelligence solutions offered by these companies have customers for example in the travel, retail or consumer goods industries. Typical services built on top of the technology are on-demand price monitoring, revenue management, product mix tracking, Web process integration or building of Web mashups.

Wrapper Induction Algorithms

WIEN is the first system for wrapper induction from semi-structured documents. The system learns extraction rules in two stages: In a fist step, simple LR rules are generated from the training data, that specify left and right delimiters for each example. The second step refines the LR rules into example conforming HLRT rules, where additional Head and Tail delimiters are added if necessary. The WIEN algorithm assumes attributes to be equally ordered among all tuples, and it cannot deal with missing attributes. WIEN supports both single-slot and multi-slot patterns.

Softmealy is based upon a finite-state transducer and it utilizes contextual rules as extraction rules. Before being input to the transducer, a document is tokenized and special separators are added between two consecutive tokens. The transducer regards its input as a string of separators, where state transitions are governed by the left and right context tokens surrounding the current input separator. SoftMealy machines have two learnable components: the graph structure of the state transducer and the contextual rules governing the state transitions. The graph structure should be kept simple and small, which is achieved

via a conservative extension policy. Contextual rules are learned by a set-covering algorithm that finds the minimal number of contextual rules covering all examples.

Stalker generates so-called landmark automata (SkipTo functions) from a set of training data. Nesting patterns allows the system to decompose complex extraction rules into a hierarchy of more simple extraction rules. Stalker employs a covering algorithm to reduce its initial set of candidate rules to a best-fit set of rules. Co-testing learns both forward and backward rules on the document. In case of agreement the system knows that the selected rule is correct, otherwise the user is given the conflicting example for labeling in order to resolve the conflict (active learning).

In boosted wrapper induction (*BWI*), delimiter-based extraction rules are generated from simple contextual patterns. Boundaries that uniquely describe relevant tuples are identified and assigned *fore* and *aft* scores. Learning in the BWI approach constitutes of determining fore detectors *F*, aft detectors *A*, and the pattern length function. The latter is a maximum likelihood estimator on the word occurrences in the training set. *F* and *A* are learned using a *boosting algorithm*, where several weak classifiers are boosted into one strong classifier. BWI works both in natural language and wrapper domains.

NoDoSe is an interactive tool that allows the user to hierarchically decompose a complex wrapper pattern into simple sub-wrappers. The system works both in the free text and the Web domain with the help of two distinct mining components. Learning rules consist of start and stop delimiters, either on text level (first component) or on HTML structure level (second component). Parsing generates a concept tree of rules that support multi-slot patterns and attribute permutation.

DEByE is another interactive induction system that applies bottom-up extraction rules. The authors define so-called attribute-value pair (AVP) patterns, and use window passages surrounding an example tuple to build lists of AVP patterns. Unlike other wrapper induction systems that typically work top-down, their approach allows for more flexibility as the order of individual patterns here is no longer relevant. They claim that, unlike other wrapper induction systems that typically work top-down, identifying atomic attribute objects prior to identifying the nested object itself adds flexibility to a system as partial objects are detected as well as nested objects with differing attribute orders. On the downside, this approach adds quite some complexity to the wrapper induction process.

Future Directions

Several trends can be identified in current wrapper induction research.

Shift to automation. The shifting of (semi-)supervised IE applications from free text to Web documents emphasized the cost problem of obtaining annotated examples from such sources. Unsupervised, fully automated methods using methods such as alignment, entity recognition, entity resolution or ontologies are motivated to overcome the so-called labelling bottleneck. Similarly, in semi-supervised systems, effort further focused to minimize the number of required examples (e.g., through active learning, see above).

Adoption of new technologies. Over time the complexity of Web sites has substantially increased. Technologies such as Web forms, AJAX or Flash are today used to build the so-called Deep Web. Many documents that are interesting for wrapper induction are therefore hidden in the Deep Web. Because usual crawling algorithm do not work well, new techniques such as focused crawling, Deep Web navigations, form mapping or metasearch receive increased research attention.

Shift to semantic annotations. The results of state-of-the-art wrapper induction systems support high confidence in success for many classic IE scenarios. With the quick and thourough adoption of the Web as a media and data communication protocol there comes an increasing demand for semantic services that allow users to enrich and interlink existing information with ontological meta-annotations. The envisioned semantic Web that would offer this functionality quickly proved to be a rather slow development process with no end in sight. Web IE systems on the other hand have started to add support for semantic labeling and even automatic detection of such relational information from labeled and unlabeled sources.

Data Sets

The most known data set used in information extraction is called RISE (http://www.isi.edu/info-agents/RISE). RISE is a collection of resources both from the Web and the natural text domains. Among others, it contains documents related to seminar announcements, Okra search, IAF address finder. Large fraction of the pages in the corpus originates before the year

2000. Because of their simplicity are those pages considered out-dated with respect to the current structure and layout of modern Web pages.

Cross-references

- ▶ Data Clustering (VII.b)
- ▶ Data Mining
- ▶ Decision Tree Classification
- ▶ Information Retrieval
- ▶ Semi-Structured Text Retrieval
- ▶ Text Mining

Recommended Reading

1. Adelberg B. NoDoSE: a tool for semi-automatically extracting structured and semistructured data from text documents. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1998, pp. 283–294.
2. Baumgartner R., Flesca S., and Gottlob G. Visual Web Information Extraction with Lixto. In Proc. 27th Int. Conf. on Very Large Data Bases, 2001, pp. 119–128.
3. Carme J., Ceresna M., and Goebel M. Web Wrapper Specification Using Compound Filter Learning. In Proc. IADIS Int. Conf. WWW/Internet 2006, 2006.
4. Chang C.H. and Kuo S.C. OLERA: Semisupervised web-data extraction with visual support. IEEE Intell. Syst., 19(6):56–64, 2004.
5. Finn A. and Kushmerick N. Active learning selection strategies for information extraction. In Proc. Workshop on Adaptative Text Extraction and Mining, 2003.
6. Freitag D. and Kushmerick N. Boosted Wrapper Induction. In Proc. 12th National Conf. on AI, 2000, pp. 577–583.
7. Hsu C.N. and Dung M.T. Generating Finite-state Transducers for Semi-structured Data Extraction from the Web. Inf. Syst., 23(8):521–538, 1998.
8. Irmak U. and Suel T. Interactive wrapper generation with minimal user effort. In Proc. 15th Int. World Wide Web Conf., 2006, pp. 553–563.
9. Knoblock C.A., Lerman K., Minton S., and Muslea I. Accurately and Reliably Extracting Data from the Web: a Machine Learning Approach. Q. Bull, IEEE TC on Data Eng., 23(4):33–41, 2000.
10. Kushmerick N. Wrapper Induction for Information Extraction. Ph.D. thesis, University of Washington, 1997.
11. Kushmerick N. Wrapper induction: Efficiency and expressiveness. Artif. Intell., 118(1–2):15–68, 2000.
12. Laender A.H.F., Ribeiro-Neto B., and da Silva A.S. DEByE – Date extraction by example. Data Knowl. Eng., 40(2):121–154, 2002.
13. Liu L., Pu C., and Han W. XWRAP: An XML-Enabled Wrapper Construction System for Web Information Sources. In Proc. 16th Int. Conf. on Data Engineering, 2000, pp. 611–621.
14. Muslea I., Minton S., and Knoblock C. STALKER: Learning extraction rules for semistructured, Web-based information sources. 1998, URL citeseer.ist.psu.edu/muslea98stalker.html.
15. Muslea I., Minton S., and Knoblock C.A. Selective Sampling with Redundant Views. In Proc. 12th National Conf. on AI, 2000, pp. 621–626.
16. Sahuguet A. and Azavant F. WysiWyg web wrapper factory (W4F). 2001, URL http://citeseer.ist.psu.edu/553711.html; http://www.ai.mit.edu/people/jimmylin/papers/Sahuguet99.ps.
17. Seymore K., McCallum A., and Rosenfeld R. Learning hidden Markov model structure for information extraction. In Proc. AAAI 99 Workshop on Machine Learning for Information Extraction. 1999.

Wrapper Generation

- ▶ Wrapper Induction

Wrapper Generator

- ▶ Web Data Extraction System

Wrapper Generator GUIs

- ▶ GUIs for Web Data Extraction

Wrapper Maintenance

Kristina Lerman, Craig A. Knoblock
University of Southern California, Marina del Rey, CA, USA

Synonyms

Wrapper verification and reinduction; Wrapper repair

Definition

A Web wrapper is a software application that extracts information from a semi-structured source and converts it to a structured format. While semi-structured sources, such as Web pages, contain no explicitly specified schema, they do have an implicit grammar that can be used to identify relevant information in the document. A wrapper learning system analyzes page layout to generate either grammar-based or "landmark"-based extraction rules that wrappers use to extract data. As a consequence, even slight changes in the page layout can break the wrapper and prevent it from extracting data correctly. Wrapper maintenance is a composite task that (i) verifies that the wrapper continues to extract data

correctly from a source, and (ii) repairs the wrapper so that it works on the changed pages.

Historical Background

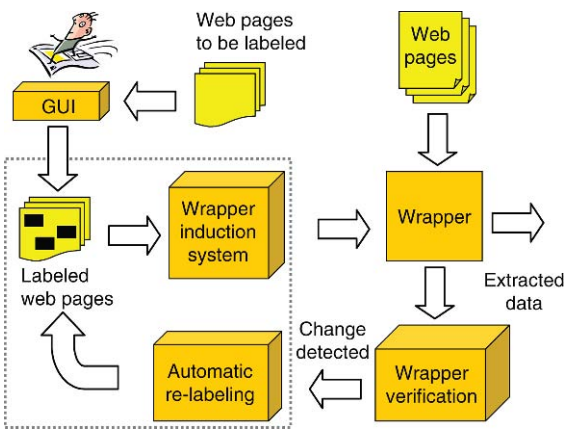
Wrapper induction algorithms [6,3,11] exploit regularities in the page layout to find a set of extraction rules that will accurately extract data from a source. As a consequence, even minor changes in page layout break wrappers, leading them to extract incorrect data, or fail to extract data from the page altogether. Researchers addressed this problem in two stages. *Wrapper verification* systems [5,9] monitor wrapper output to confirm that it correctly extracts data. Assuming that the structure and the format of data does not change significantly, they compare new wrapper output to that produced by the wrapper when it was successfully invoked in the past and evaluate whether the two data sets are similar. Wrapper verification systems learn data descriptions, also called content features, and evaluate wrapper output similarity based on these features. Two different representations of content features have been studied in the past. Kushmerick [4,5] and Chidlovskii [1] represent wrapper output by global numeric features, e.g., word count, average word length, HTML tag density, etc. Lerman et al. [8,9] instead learn syntactical patterns associated with data, e.g., they learn that addresses start with a number and are followed by a capitalized word. Both systems ascertain that the distribution of features over the new output is similar to that over data extracted by the wrapper in the past. If statistically significant changes are found, the verification system notifies the operator that the wrapper is no longer functioning properly.

If the wrapper is not working correctly, the next task is to automatically repair it by learning new extraction rules. This task is handled by a *wrapper reinduction* system. Since the wrapper learning algorithm simply needs a set of labeled examples of data on new pages in order to learn extraction rules, researchers have addressed the reinduction problem by automatically labeling data instances on new pages. The problem is more complex than simply looking for instances of old wrapper output on new pages because data can change in value (e.g., weather, stock quotes) or format (abbreviations added, etc.). Instead, to identify the new data instances, reinduction systems combine learned content features, heuristics based on page structure, and similarity with the old wrapper output. Lerman et al. [9] use syntactical patterns learned from old wrapper output to identify possible data instances on new pages. They then exploit regularities in the structure of data presented on the page to narrow candidates to ones likely to be the correct instances of a field. For example, a Web source listing restaurant information displays city, state and zipcode in the same order in all records. Likewise, instances of the same field are, in most cases, surrounded by the same HTML markup on all pages. Chidlobskii [1] take a similar approach, selecting data instance candidates based on learned numeric features, and exploiting page grammar and data structure to identify correct examples. Meng et al. [10] and Raposo et al. [12], on the other hand, rely on the document-object-model (DOM) tree to help identify correct examples of data on new pages.

Foundations

Figure 1 shows the life cycle of a wrapper. During its normal operation the wrapper is extracting data from the source, and learning content features that describe the data. The wrapper verification system monitors wrapper output to confirm that it is working correctly. Once it determines that a wrapper has stopped working, it uses the learned content features and page and data structure heuristics to identify data on the new pages. The labeled pages are then submitted to the wrapper induction system which learns new extraction rules.

Content-based features play a central role in both verification and reinduction. Data returned by online sources usually has some structure: phone numbers, prices, dates, street addresses, etc. all follow some format. Unfortunately, character-level regular expressions are too fine-grained to capture this structure. Instead,



Wrapper Maintenance. Figure 1. Life cycle of a wrapper, illustrating verification and reinduction stages.

researchers represent the structure of data as a sequence of tokens, called a pattern. Tokens are strings generated from an alphabet containing different character types: alphabetic, numeric, punctuation, etc. The token's character types determine its assignment to one or more syntactic categories: alphabetic, numeric, etc. The categories form a hierarchy, which allows for multi-level generalization. Thus, a set of zipcodes "90210" and "90292" can be represented by specific tokens "90210" and "90292," as well as more general types `5Digit`, `Number`, `Alphanum`. An efficient algorithm to learn patterns describing a field from examples was described in [9]. An alternate approach represents the content of data by global numeric features, such as character count, average word length, and density of types, i.e., proportion of characters in the training examples that are of an HTML, alphabetic, or numeric type. This approach would learn the following features for zipcodes: Count = 5.0, Digits = 1.0, Alpha = 0.0, meaning that zipcodes are, on average, five characters long, containing only numeric characters.

Wrapper Verification

Wrapper verification methods attempt to confirm that wrapper output is statistically similar to the output produced by the same wrapper when it was successfully invoked in the past. Henceforth, successful wrapper output are called training examples. Wrapper verification system learns content-based features and their distribution over the training examples. In order to check that the learned features have a similar distribution over the new wrapper output, one approach [5] calculates the probability of generating the new values randomly for each field. Individual field probabilities are then combined to produce an overall probability that the wrapper has extracted data correctly. Another approach to verification [9] uses training examples to learn data patterns for each field. The system then checks that this description still applies to the new data extracted by the wrapper.

Wrapper Reinduction

Content-based features are used by the wrapper reinduction system to automatically label the new pages. It first scans each page to identify all text segments that match the learned features. The candidate text segments that contain significantly more or fewer tokens than expected are eliminated from consideration. The learned features are often too general and will match many, possibly hundreds, of text segments on each page. Among these spurious text segments are the correct examples of the data field. Researchers use additional heuristics to help narrow the set to correct examples of the field: true examples are expected to appear in the same position (e.g., after a <b> tag) and in the same context (e.g., restaurant address precedes city name) on each page. In addition, they are expected to be visible to the user (not part of HTML tag) and appear within the same block of HTML page (or the same DOM sub-tree). This information is used to split candidate extracts into groups. The next step is to score groups based on their similarity to the training examples. This technique generally works well, because at least some of the data usually remains the same when the page layout changes. Of course, this assumption does not apply to data that changes frequently, such as weather information, flight arrival times, stock quotes, etc. However, previous research has found that even in these sources, there is enough overlap in the data that the approach works.

The final step of the wrapper reinduction process is to provide the extracts in the top-scored group to the wrapper induction algorithm, along with the new pages, to learn new data extraction rules.

For some types of Web sources automatic wrapper generation is a viable alternative to wrapper reinduction. One such system, developed by Crecenzi et al. [2], learns grammar-based extraction rules without requiring the user to label any pages. Once wrapper verification system detects that the wrapper is not working correctly, it collects new sample pages from which it learns new extraction rules. Still another approach to data extraction uses a combination of content-feature learning and page analysis to automatically extract data from Web pages [7].

Key Applications

Automatic wrapper maintenance can be incorporated into a wrapper learning system to reduce the amount of user effort involved in monitoring wrappers for validity and repairing them.

Experimental Results

To evaluate the performance of wrapper maintenance systems, researchers collected data over time, saving Web pages along with data extracted from them by wrappers. In one set of experiments researchers [9] monitored 27 wrappers (representing 23 distinct data

sources) by storing results of 15–30 queries periodically over 10 months. The wrapper verification system learned content descriptions for each field and made a decision about whether the new output was statistically similar to the training set.

A manual check of the 438 comparisons revealed 37 wrapper changes attributable to changes in the page layout. The verification algorithm correctly discovered 35 of these changes and made 15 mistakes. Of these mistakes, 13 were *false positives*, which means that the verification program decided that the wrapper failed when in reality it was working correctly. Only two of the errors were the more important *false negatives*, meaning that the algorithm did not detect a change in the data source. Representing the structure of content by global numeric features instead would have missed 17 wrapper changes.

Researchers also evaluated the reinduction algorithm only on the ten sources that returned a single tuple of results per page. They used the algorithm to extract (35) fields from ten Web sources for which correct output is known, regardless of whether the source had actually changed or not.

The output of the reinduction algorithm is a list of tuples extracted from ten pages, as well as extraction rules generated by the wrapper induction system for these pages. Though in most cases they were not able to extract every field on every page, they still learned good extraction rules as long as a few examples of each field were correctly labeled. The algorithm was evaluated in two stages: first, researchers checked how many fields were successfully identified; second, they checked the quality of the learned extraction rules by using them to extract data from test pages.

The algorithm was able to correctly identify fields 277 times across all data sets making 61 mistakes, of which 31 were attributed to false positives and 30 to the false negatives. On the extraction task, the automatically learned extraction rules achieved 90% precision and 80% recall.

The results above apply to sources that return a single item per page. Other researchers have applied reinduction methods to repair wrappers for sources that return lists of items per page [1,13] with similar performance results.

Data Sets

The data set used in the wrapper maintenance experiments discussed above is available from the University of Southern California Information Sciences Institute: http://www.isi.edu/integration/datasets/reinduction/wrapper-verification-data2.zip.

The data contains results of invocations of wrappers for different Web sources over a period of 10 months. Each wrapper was executed with the same small set of queries (if they contained no time-dependent parameters). The data set contains both the returned Web pages and data extracted by the wrapper from the source. Over the data collection period, several sources changed in a way that prevented wrappers from correctly extracting data. The wrappers were repaired manually in these cases and data collection continued. The performance of the USC's verification and reinduction systems on this dataset is reported in [9].

Cross-references

- ▶ Web Data Extraction System
- ▶ Web Information Extraction
- ▶ Wrapper Induction

Recommended Reading

1. Chidlovskii B. Automatic repairing of web wrappers by combining redundant views. In Proc. 14th IEEE Int. Conf. Tools with Artificial Intelligence, 2002, pp. 399–406.
2. Crescenzi V. and Mecca G. Automatic information extraction from large websites. J. ACM, 51(5):731–779, 2004.
3. Hsu C.-N. and Dung M.-T. Generating finite-state transducers for semi-structured data extraction from the web. J. Inform. Syst., 23:521–538, 1998.
4. Kushmerick N. Regression testing for wrapper maintenance. In Proc. 14th National Conf. on AI, 1999, pp. 74–79.
5. Kushmerick N. Wrapper verification. World Wide Web J., 3(2):79–94, 2000.
6. Kushmerick N., Weld D.S., and Robert B. Doorenbos. Wrapper induction for information extraction. In Proc. 15th Int. Joint Conf. on AI, 1997, pp. 729–737.
7. Lerman K., Gazen C., Minton S., and Knoblock C.A. Populating the semantic web. In Proc. AAAI Workshop on Advances in Text Extraction and Mining, 2004.
8. Lerman K. and Minton S. Learning the common structure of data. In Proc. 12th National Conf. on AI, 2000, pp. 609–614.
9. Lerman K., Minton S., and Knoblock C. Wrapper maintenance: a machine learning approach. J. Artif. Intell. Res., 18:149–181, 2003.
10. Meng X., Hu D., and Li C. Schema-guided wrapper maintenance. In Proc. of 2003 Conf. on Web Information and Data Management, 2003, pp. 1–8.
11. Muslea I., Minton S., and Knoblock C.A. Hierarchical wrapper induction for semistructured information sources. Auton. Agent Multi Agent Syst., 4:93–114, 2001.

12. Raposo J., Pan A., Alvarez M., and Hidalgo J. Automatically generating labeled examples for web wrapper maintenance. In Proc. of 2005 IEEE/WIC/ACM Int. Conf. on Web Intelligence, 2005, pp. 250–256.
13. Raposo J., Pan A., Álvarez M., and Hidalgo J. Automatically maintaining wrappers for semi-structured web sources. Data Knowl. Eng., 61(2):331–358, 2007.

Wrapper Repair

► Wrapper Maintenance

Wrapper Robustness

► Wrapper Stability

Wrapper Stability

Georg Gottlob
Oxford University, Oxford, UK

Synonyms

Wrapper robustness; Wrapper adaptability

Definition

A *wrapper*, whether hand written or generated by a *Web data extraction system*, is a program that extracts data from information sources of changing content and translates the data into a different format. The *stability* of a wrapper is the degree of insensitivity to changes of the presentation (i.e., formatting, layout, or syntax) of the data sources. A stable wrapper is ideally able to extract the desired data from source documents even if the layout of the current documents differs from the layout of those documents that were used as examples at the time the wrapper was generated. Thus, a wrapper is stable or robust if it is able of coping with perturbations of the original layout.

Key Points

Documents, and, Web pages in particular are usually susceptible to slight layout changes over time. Most of these changes are minor changes. For example, an advertisement may be added to the top of a Web page. One would ideally wish that changes in the layout of the source will not prevent a wrapper to extract the desired data, i.e., that the wrapper remains stable under such changes (or, equivalently, robust to these changes), as long as the data items to be extracted still appear in the document. For obvious reasons, totally stable wrappers can hardly be generated. However, there are sensible differences in stability between simple *screen scrapers* or *Web scrapers* on one hand, and wrappers generated by advanced *Web data extraction systems*, on the other hand. Screen scrapers or Web scrapers often rely on exact data positions in a document or in the parse tree of an HTML Web page. For example, a screen scraper may memorize that a certain price to extract from a Web page is the data item occurring as the 37th leaf of the Web page's parse tree. Of course, in case the page's layout is altered by adding, say, an advertisement in front of the data, such a wrapper will fail to extract the intended price. Advanced Web data extraction systems, on the other hand, characterize the objects to be extracted via syntactic, contextual, or semantic properties rather than by their position only. For example, the Lixto system [1] could characterize a desired price as a numeric data item preceded by a currency symbol, occurring in the second column of a table entitled "Price list," such that a certain article name appears in the first column of the same row. Clearly, such a characterization is much more stable than a purely positional one. There are only very few studies on Wrapper stability, among which, for example [2], and this topic remains an important one for future research. One issue of particular interest is the automatic adaptation of wrappers to new layouts [2–5].

Cross-references

► Web Data Extraction System

Recommended Reading

1. Baumgartner R., Flesca S., and Gottlob G. Visual web information extraction with Lixto. In Proc. 27th Int. Conf. on Very Large Data Bases, 2001, pp. 119–128.
2. Davulcu H., Yang G., Kifer M., and Ramakrishnan I.V. Computational aspects of resilient data extraction from semistructured sources. In Proc. 19th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems, 2000, pp. 136–144.
3. Meng X., Hu D., and Li C. Schema-guided wrapper maintenance for web-data extraction. In Proc. Fifth ACM CIKM Int. Workshop on Web Information and Data Management, 2003, pp. 1–8.
4. Phelps T.A. and Wilensky R. Robust intra-document locations. In Proc. 9th Int. World Wide Web Conf., 2000, pp. 105–118.

5. Wong T. and Lam W. A probabilistic approach for adapting information extraction wrappers and discovering new attributes. In Proc. 2004 IEEE Int. Conf. on Data Mining, 2004, pp. 257–264.

Wrapper Verification and Reinduction

► Wrapper Maintenance

Write Once Read Many

Kenichi Wada
Hitachi Limited, Tokyo, Japan

Synonyms

Write once read mostly; Write one read multiple; WORM

Definition

A data storage technology that allows information to be written to storage media a single time only, and prevents the user from accidentally or intentionally altering or erasing the data. Optical disc technologies such as CD-R and DVD-R are typical WORM storage.

Key Points

There are two types of WORM technology: physical and logical WORM.

1. Physical WORM uses storage media which physically can be written only once, and prevents the user from accidentally or intentionally altering or erasing the data. Optical storage media such as CD-R and DVD-R are good examples. Offering fast access and long-term storage capability, physical WORM has historically been used for archiving data that requires a long retention period.
2. Logical WORM uses storage systems which provide a WORM capability by using electric keys or other measures to prevent rewriting even if with non-WORM storage media, such as disk drives. Driven by the recent growth of legislation in many countries, logical WORM is now increasingly being used to archive corporate data, such as financial documents, emails, and certification documents, and to find them quickly in case of discovery.

Cross-references

► Digital Archives and Preservation

Write Once Read Mostly

► Write Once Read Many

Write One Read Multiple

► Write Once Read Many

WS-BPEL

► Composed services and WS-BPEL

WS-Discovery

► Discovery