I

I/O cache

► Buffer Pool

I/O Model of Computation

Donghui Zhang[1], Vassilis J. Tsotras[2]
[1]Northeastern University, Boston, MA, USA
[2]University of California-Riverside, Riverside, CA, USA

Synonyms

Disk-based model

Definition

The I/O model of computation measures the efficiency of an algorithm by counting how many disk reads and writes it needs. It is widely applicable to the database environment, since most data is stored on disks and disk access typically dominates CPU time.

Key Points

For many computing-intensive applications, the appropriate model of computation is to measure CPU time. Yet in data-intensive applications, such as databases, it is more relevant to measure the number of disk I/Os [1]. This is termed the "I/O model of computation," or disk-based model. Nowadays, most hard drives use the seek-rotate-transfer protocol [2]. In order to transfer some data from disk to memory (so as the data can be processed by the CPU), or to transfer data back to disk, the hard drive needs first to spend some "seek time" to move the read/write head to the cylinder where the data is located at. Then the "rotational delay" is spent until the sector containing the data rotates to a position under the read/write head. Finally, time is spent to actually transfer the data from/to the CPU. Typically, seek time is longer than rotational delay, which is in turn longer than transfer time. Therefore reading a few bytes of data takes roughly as long as reading thousands of bytes. Due to this reason, data is stored on disks in units called blocks or pages. Every disk I/O corresponds to reading or writing one such page. Moreover, a random disk I/O costs more than a sequential access. This is because the access of multiple sequential pages on the disk does not involve major seek and rotational times (since in sequential access, a page is accessed after its neighbor page). Hence a more accurate I/O model should account for the difference between random and sequential I/O.

There are three ways to minimize the disk accesses in a database environment: (i) by buffering in main memory pages that have already been accessed (and thus future accesses can be served by a buffer access and not a disk access), (ii) by transferring a number of consecutive pages at once, called bucket, anticipating the next requests due to data locality, and, (iii) by using structures (indices) that organize the data into pages so as searching for a particular record takes few page accesses.

To exemplify the importance of the I/O computation in data structures, consider the following scenario. Assume an application (query) requests a record from a database file. If a balanced binary search tree is directly implemented on top of this file, to search for a record would need $O(\log_2 n)$ I/Os, where n is the number of records in the tree. For example, if n is a million records, this means about 20 I/Os. If instead, a disk optimized structure is used (like the B+-tree) the same search is much more efficient (in number of page I/Os). The B+-tree extends the binary tree, by expanding a node size. Every tree node corresponds to one disk page. If for simplicity, every index node as well as leaf node in the B+-tree contains 100 entries, a million records fits into a three-level B+-tree. Then to search for any record in the file, only three I/Os are needed, which is much faster than the 20 I/Os. Disk resident data structures are called paginated or disk-based or external.

Before a new page read is executed, the buffer manager is first examined for whether it already contains the requested page. For instance, if a page is read 100 times, only the first read triggers an actual disk read, while subsequence reads are serviced by returning the in-buffer page (and thus avoiding the cost of a page I/O). Of course the buffer has limited capacity and issues like page replacement policies play an important role.

Cross-references

- ▶ Access Methods
- ▶ Buffer Management
- ▶ Indexing

Recommended Reading

1. Aggarwal A. and Vitter J.S. The input/output complexity of sorting and related problems. Commun. ACM, 31 (9):1116–1127, 1988.
2. Ramakrishnan R. and Gehrke J. Database Management Systems (3rd edn.). McGraw-Hill, New York, NY, 2003.

Icon

Stefano Levialdi
Sapienza University of Rome, Rome, Italy

Synonyms

Picture; Image; Representation

Definition

Originally, the Greek word *eikon* stood for an image, carrying some meaning as in typical portraits of sacred persons within the Orthodox Church. An operational definition of icon was given by Peirce [1] as *anything that stands for something else, to somebody, in some respect or capacity.* Being so general, it covers most customary practices, typically linking linguistic, pictorial (or even auditory) expressions to a *meaning* that needs to be *interpreted* by a human. In general discourse, an icon may imply an idol (as a pop star) or a symbol (the Rotary wheel worn on coat lapels) that represents a group of persons or a life style (the Nike symbol for sports). Within Human-Computer Interaction, the common understanding of an icon is to consider it as a visual metaphor representing a file, a directory, a window, an option or a program. Whenever a number of icons are presented together, this group is referred to as an icon bar, generally at the top of the page in most web browsers.

Key Points

The study of signs, a scientific discipline under the name of Semiotics, started from the work of two contemporary researchers: Ferdinand de Saussure (a linguist) [2] and Charles Sanders Peirce (a philosopher) [1]. De Saussure showed that language may be considered as a collection of *signs*, each one formed by a pair (a twofold nature) named signifier/signified, and only if both were present, the sign could be a valid indication of a meaning, otherwise the sign only represents itself. Peirce made a well known triangular classification (a threefold nature), where the sign stands in the center and the three vertices correspond to (i) the representation, (ii) the object (referent) and (iii) the interpretant (meaning) [3]. The main issue here is that the interpretation is subjective and dynamic, there is no unique meaning corresponding to a class of signs: the *icons*. Following Peirce's sign taxonomy, he indicated three possible sign classes: the *icon* (where a mental process is required to understand it), the index (having a causal relationship to its signified, like smoke for fire) and the *symbol* (having a totally arbitrary relationship, like a red cross for the corresponding, well known, international medical organization). The icon is wrongly considered to be similar to its signified; typically by looking at the icon one should infer information about its signified, yet this is not always the case, since different meanings may be attributed to the icon, depending on the observer. This fact provides a possible approach to ambiguity management in human-computer interfaces, by restricting the user model to a given class of users. The design of effective icons should be performed after an accurate study of the cultural background, age, and motivation of the potential users.

Cross-references

- ▶ Symbolic representation
- ▶ Visual metaphor
- ▶ Visual representation

Recommended Reading

1. Charles Sanders Peirce. Collected Papers of Charles Peirce, vol. 1–8, C. Hartshorne and P. Weiss (eds.). Harvard University Press, Cambridge, MA, 1931–1958.

2. de Saussure F. Introduction to the 2nd Course on General Linguistics (1908–1909), collected by Robert Godel, Raffaele Simone (ed.). Italian Edition, Ubaldini, Rome, 1970.
3. de Souza C. S. The Semiotic Engineering Approach to Human-Computer Interaction. The MIT Press, Cambridge, MA, 2005.

Iconic Displays

Georges Grinstein, Damon Andrew Berry
University of Massachusetts, Lowell, MA, USA

Synonyms

Icons; Glyphs; Iconographics

Definition

Iconic displays are visualizations that generalize traditional displays (especially scatterplots) where each record, instead of being drawn as a point, is represented by a more general primitive called an icon or glyph. The goals are to harness human perception, especially texture, and to display many more parameters. Whereas a pixel is driven by three data values from some color model (typically red, green, and blue) an icon is a geometric object driven by potentially many values, with some icons displaying over 30. Some icons are drawn using lines, some using colored areas, some move and vibrate, and some even have sound output. Some iconic displays drop the Cartesian base of the underlying display and use alternative layout techniques. However, in all of these, the key defining factor is the representation of a record in a visualization by a very general, most often geometric, primitive, with the goal of producing more perceptually-based displays.

Historical Background

The analysis of complex images is still performed largely by human visual means which is most highly effective in situations where the patterns of potential interest are directly visible. The most widely used technique for displaying multivariate data or multiparameter imagery is based on the representation of different parameters as points with two or three position coordinates and color based on a color model such as RGB (Red, Green, Blue) or HLS (Hue, Lightness, Saturation).

Such scatterplots employ simple graphic representations with attributes (retinal variables) characterized by Bertin [2] and Healy et al. [8]. The number of variables or parameters that can be displayed with such a two dimensional scatterplot is easily five using five degrees of freedom (x, y, shape, size, and color) and seven if one thinks of color as three separate variables (e.g., RGB).

But what if one wishes to go beyond five or seven variables? This was the motivation for the development of icons. By increasing the number of variables encoded with more complex representations, an arbitrary number of variables can be encoded. Examples include Chernoff faces [4], asymmetrical faces [5], star glyphs [3], the stick figure icons of Pickett et al. [1,7,13], color icons [11], moving icons [15], and moxels [19].

In addition to the different icons, there are a number of different algorithms for placing them on a two-dimensional plane. Ward presents an overview and taxonomy of such icon or glyph placement strategies [17].

Foundations

Perceptual Foundations

Gibson [6] developed an ecologically-based approach to perception, quite different from the then conventional approach. The human visual system is an ecologically evolved system still guiding the behavior of animals. Humans discriminate textures very effectively and use variations in visual texture as important sources of information in the detection and recognition of objects. Pickett [13] suggested that modern displays (visualizations) should be developed using this ecological spatiotemporal pattern recognition system of the human optical system. Further, experiments developed by Julesz, demonstrated that differences in textons (characteristic features of the surface) or their densities can be preattentively detected by the human visual system [9]. The work developed by Julesz shows that texture is a statistical property of textons. The perception and discrimination of textures seems to be based on the density (first order statistics) of textons.

In 1967, Jacques Bertin [2] in Sémiologie Graphique presented the fundamentals of information encoding via graphic representations as a semiology, a science dealing with sign-systems. His first key point is the strict separation of content (the information to encode) from the container (the properties of the graphic system). Bertin defined a graphical vocabulary consisting of three elements: marks, positional variables, and retinal variables. Marks are artifacts like points, lines, and areas. Positional variables are (usually)

two planar dimensions. Retinal variables are encoded entities such as size, color, shape, orientation, texture, and value. These were extended by others to include temporal aspects with animation and work in three dimensions. Additional parameters such as shading, connectivity, labels, visibility, and tabularity were added. Extensions to the non-visual have also appeared such as the use of sound as a perceptual variable [16] as well as touch and smell.

In all of the above the goal is to develop perceptually salient displays.

Iconographics

Iconographics initially developed at the University of Massachusetts Lowell, harnesses texture perception to increase the dimensionality of the represented data visible in an image [13]. Each datum is represented by an icon whose visual features are controlled by the data. Two of the data variables control the position of each icon on the display surface. When icons are used to represent imagery data, the icon position simply corresponds to the pixel position. With sufficient density, the icons form a surface texture display, and structures in the data are revealed as streaks, gradients, or islands of contrasting texture.

Humans discriminate textures very effectively and use variations in visual texture as important sources of information in the detection and recognition of objects. Based on the early work of Pickett, Pickett and Grinstein [13] developed data representation techniques that engaged and harnessed the mechanisms of texture perception. Later, Levkowitz et al. [14] came back to color as a basis for integrating multiparameter images, but with a more sophisticated approach based textural representations.

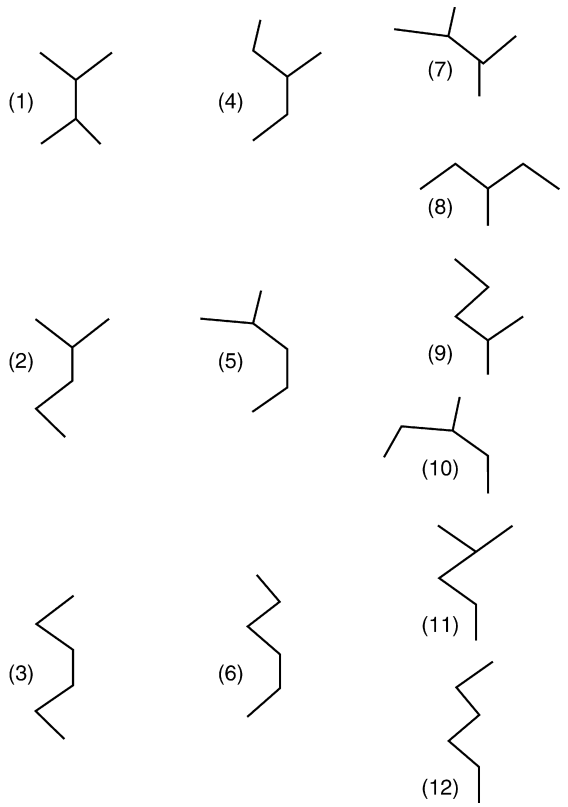
The iconographic technique extends the classic visualization approaches by representing each record of a multidimensional database as an icon whose features (such as color, geometry, and sound) are under the control of the various fields of the record. This technique allows the information content of the objects represented to have high dimensionality and allows for the fusion of multiple data sets (usually images) into a single integrated multi-modal display.

This is done through an icon which is an $m \times n$ box of screen pixels. A single pixel can be thought of as a 1×1 icon. Because an icon is a box of pixels, it can represent many more attributes (data parameters) than a single screen pixel. The increased per-element information content comes at the price of decreased screen resolution, a tradeoff that must be considered in the design of any iconographic display.

Stick Figure Icons

Figure 1 shows a typical stick-figure icon. The line segments are called the "limbs" of the icon. Each icon has a special limb, called the "body," which serves as a reference for the various geometric transformations that an icon can undergo. The first complete family of stick-figure icons had twelve members, each of which has up to five limbs connected in a unique configuration. The five-limbed icon [1] was designed to display multiparameter images on a single display in a way that would engage the vision system's ability to perceive textures.

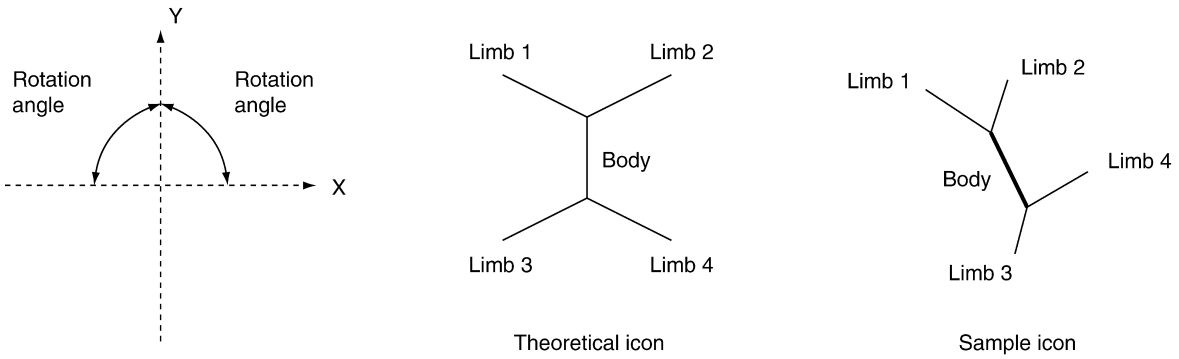
Each limb has at least three parameters that can be bound to data attributes: the angle, intensity, and length. The manner in which limbs are attached to the body or to each other defines the family of stick figure icons. Figure 2 shows one member of the family of stick figure icons. The figure shows the icon in its base



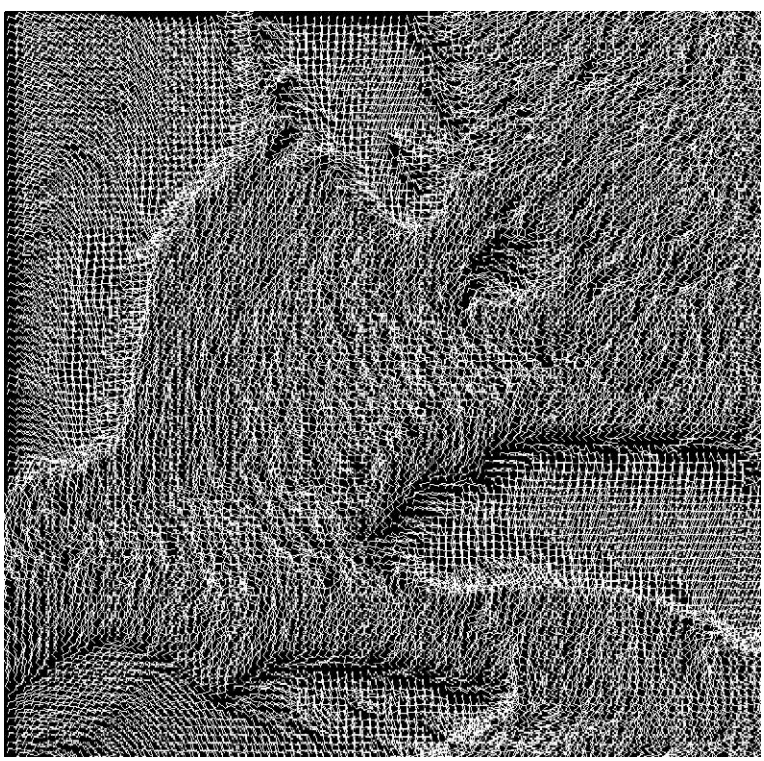
Iconic Displays. Figure 1. Family of stick figure icons.

configuration with no data mapped to any limb parameters and also a sample icon with data mapped to the limbs' parameters. The value of any variable is normalized into the unit interval [0,1]. The variable is mapped to a limb with the limb's reference angle or rotation angle determined from the normalized value (0 = 0°; 0.5 = 90°; 1.0 = 180°).

The general character of displays made with this icon can be seen in Fig. 3, one of Grinstein and Pickett's earliest displays, where they merged four grayscale images. The merged images are satellite images of a portion of the Great Lakes region (readers familiar with this area will immediately recognize Lakes Erie, Ontario, and Huron, as well as Georgian Bay and Lake Simcoe). Two of the images are from the infrared spectrum and two are from the visible spectrum. The goal is to show in one picture both the fine detail from the visible images and the heat signatures of the infrared images (here, the lakes are colder than the surrounding land). The mappings chosen produced an



Iconic Displays. Figure 2. One member of the stick-figure icon family.



Iconic Displays. Figure 3. Satellite image of the Great Lakes – line icon.

integrated display that shows the lakes as flat, concave features in relation to the rougher and predominantly convex land features. These three-dimensional effects are striking and highly effective.

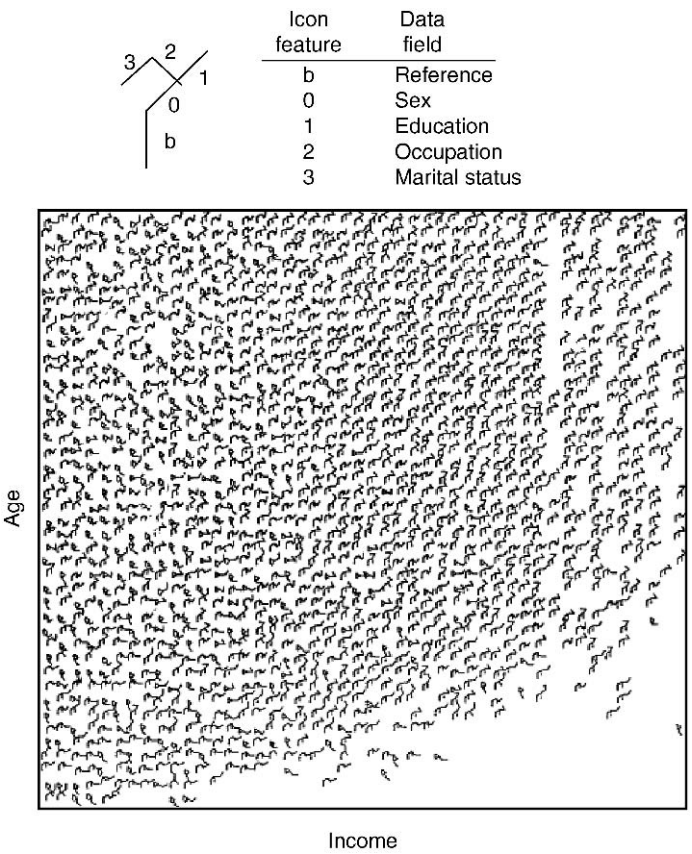
Figure 4 shows a scatterplot of a subset of the PUMS US Census data of engineers and technicians in New England, using the stick figure icon developed by Pickett and Grinstein in 1989. This icon encodes four variables (sex, education, occupation, and marital status). Each variable is mapped to a limb whose angles are determined by the value of the variable. Placing the icon on the scatterplot encodes two more variables (income and age). The result is an iconic display that encodes six variables.

Figure 5 represents some of the possible geometric layouts of icon number 4 from the stick figure family in Fig. 1 for various values of the variables. There are 16 possible configurations of the stick figure icon (Fig. 5), each corresponding to a particular geometric organization of the icon's limbs. Of particular interest are females with a high level of education since the icon forms a triangle, a highly perceptual object. For this data set (engineers and technicians in New England) almost all individuals have a high level of education. Since there are few women engineers and technicians with a low level of education one can state that all icons in Figure 3 having a triangle can be interpreted as female.

It is thus possible to get some insight into this subset. For example one can see the boundary between females and males; one can see that most female's salaries fall within the first third of the salary scale. Once can also quickly see outliers and trends in Fig. 4.

Color Icons

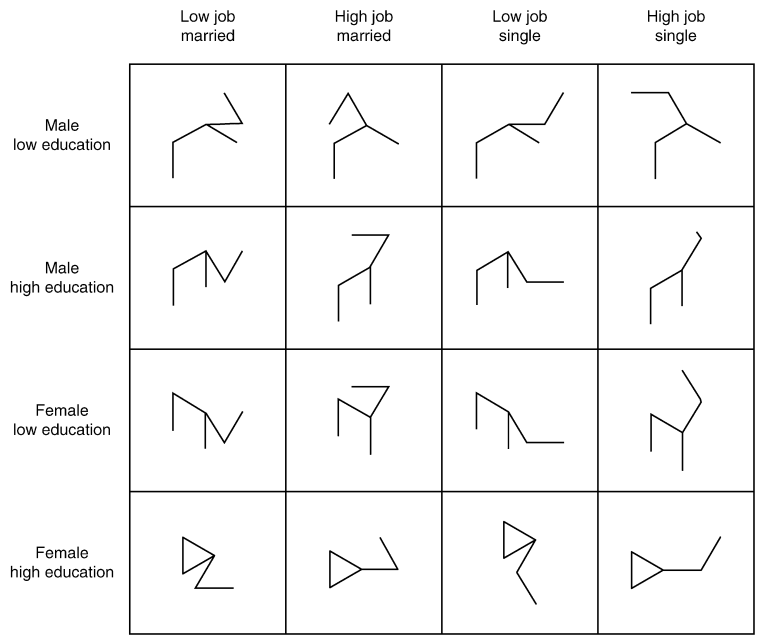
Applying the concepts learned with the stick-figure icon to color, Levkowitz developed a color icon [11]. In the color technique, a pixel in the input data is represented by an arbitrarily-sized m x n box of screen pixels. In the most frequently used version of the color



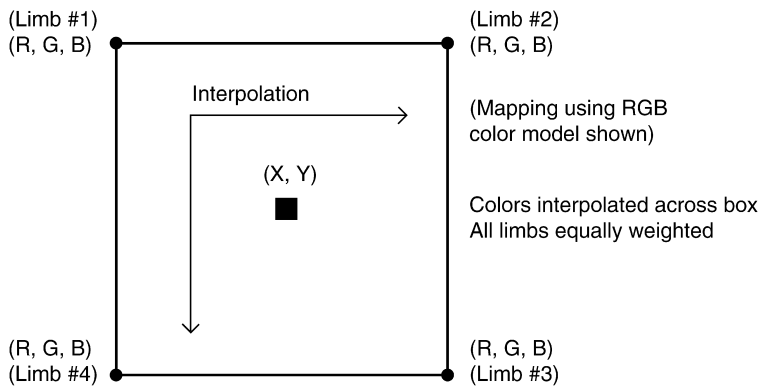
Iconic Displays. Figure 4. PUMS US Census data using a Picket icon.

icon, each of up to 12 data parameters is mapped to a color channel of the four corners of the icon box. Each data value determines the intensity of that color channel. The colors of the remaining pixels in the box are obtained by interpolation. The color icon is independent of color models and was tested with four color models: GLHS (Generalized Lightness, Hue, Saturation), RGB (Red, Green, Blue), Munsell Book of Color, or CIELUV (Commission on Illumination 1976 L*, u*, v*). The design of this icon is shown in Fig. 6. Figure 7 shows the color icon used to display several parameters of the FBI (Federal Bureau of Investigation) homicide database.

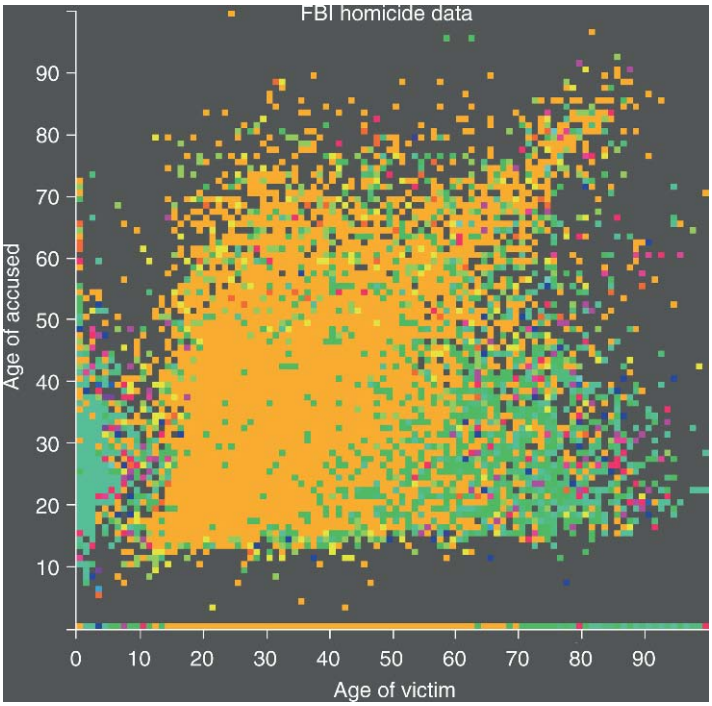
In Fig. 8, standout regions (which are common with other similar color icon displays of the same data set) can easily be identified. The first (labeled A), highlights very young victims with different attributes. The greenish color shows offenders whose homicides were family related. The second (labeled B) is the tail in the display which highlights spouses involved in the homicide. The final obvious pattern (labeled C) is that of zero-aged offenders, clearly produced by missing data.



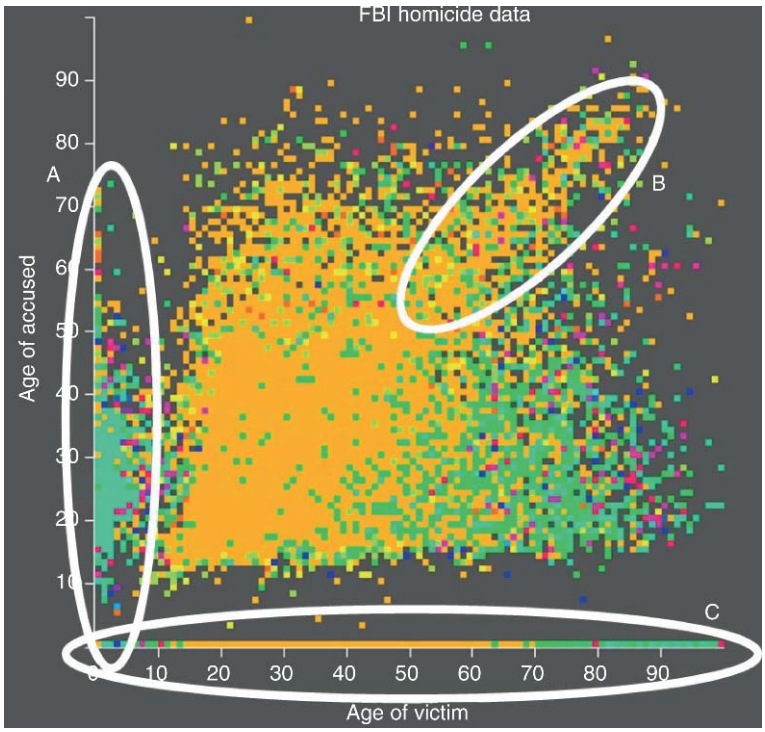
Iconic Displays. Figure 5. The stick figure icon's possible representation for various values of the parameters of a record from the Census data.



Iconic Displays. Figure 6. Design of color icon.



Iconic Displays. Figure 7. FBI homicide data using a color icon.



Iconic Displays. Figure 8. The same figure as in 7, but with three regions of interest.

Kinetic Displays and Moxels

Kinetic displays are extensions of the above icon displays in which motions of the graphical icon's limbs are used to represent attributes of the data, so the elements are in constant motion independent of user interactions. Example motions of data driven graphics include translations and rotations that produce visual vibrations and oscillations. Such kinetic displays are designed to capitalize on the human preattentive ability to perceive and understand motion.

The term "pixel" stands for "picture element;" similarly the term "moxel" stands for "moving element." Moxels are an extension of standard iconographic displays into three dimensional displays that incorporate motion coding of data dimensions.

The first such display was produced by Pinkney [15] with several interactions a "magnet" interaction; a "vibrating" interaction; a "comb" interaction; a "zoom" interaction; and a "pinwheel" interaction. In all of these interactions the mouse is used for specifying a region of interest. Interaction acts over all icons within the selected region. Users can select either a constant value or a variable (data attribute) to drive the interaction. Icons within the selected region are transformed in some way (depending on which interaction) that is proportional to the value driving the interaction. For example in the "magnet" interaction the mouse acts as a magnet repelling or attracting icons. Users can select if the magnet will affect the angle of the icons, the position of the icons, or both. The value associated with the interaction is used to determine the angular and positional response of the icon to the magnet, in the case of magnet interaction, lower values imply greater effect.

Yang et. al. [19] presented an information visualization system that links both static and kinetic visualizations. The images in Fig. 9 show a zoomed out view (left) and a zoomed in view (right) of a moxel visualization of the AVARIS (Airborne Visible/Infrared Imaging Spectrometer) dataset. These static images do not really do justice to the actual moving visualizations, where independent regions are clearly visible not only due to common textures, but due to coordinated movements as well.

Key Applications

Iconographic displays have become part of many different visualization efforts and environments. They have been used for both information visualization as well as scientific visualization in areas such as GIS (Geographic Information System) [20], physics [18], medicine [10], and computer network security [12].

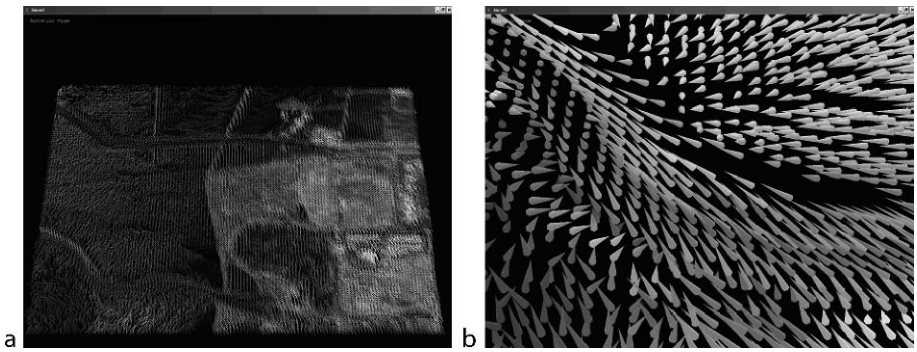
Data Sets

The PUMS US Census data used in Fig. 1 was from the 1980 census. Comparable data fro the last conducted US census may be found at: http://www.census.gov/main/www/pums.html.

The Great Lakes satellite image in Fig. 2 is based on data from the National Oceanic and Atmospheric Administration (NOAA) and is available from the National Geophysical Data Center (NGDC) at: http://www.ngdc.noaa.gov/dmsp/download.html.

The AVIRIS data used for the moxel visualizations in Fig. 9 is available at: http://aviris.jpl.nasa.gov/html/data.html.

The FBI Homicide Data visualized in Figs. 6 and 7 is available at: http://www.fbi.gov/ucr/ucr.htm.



Iconic Displays. Figure 9. Moxel visualizations of the AVIRIS dataset: a zoomed out image (left) and a zoomed in image (right).

Cross-references

- ▶ Applied Perception
- ▶ Glyphs
- ▶ Iconographics
- ▶ Icons
- ▶ Perceptual Displays
- ▶ Visual Perception
- ▶ Visualization

Recommended Reading

1. Bergeron R.D. and Grinstein G.G. A reference model for scientific visualization. In Proc. Eurographics '89', 1989, pp. 393–399.
2. Bertin J. Semiology of Graphics. The University of Wisconsin Press, Madison, WI, 1983.
3. Chambers J.M., Cleveland W.S., Kleiner B., and Tukey P.A. Graphical Methods for Data Analysis. Wadsworth, Belmont, CA, 1983.
4. Chernoff H. The use of faces to represent points in k-dimensional space graphically. J. Am. Stat. Assoc., 68:361–368, 1973.
5. Flury B. and Riedwyl H. Graphical representation of multivariate data by means of asymmetrical faces. J. Am. Stat. Assoc., 76:757–765, 1981.
6. Gibson J.J. The Ecological Approach to Visual Perception. Houghton-Mifflin, Boston, 1979.
7. Grinstein G., Pickett R., and Williams M.G. EXVIS: an exploratory visualization environment. In Proc. Graphics Interface '89, 1989.
8. Healey C.G., Booth K.S., and Enns J.T. Visualizing real-time multivariate data using preattentive processing. ACM Trans. Model. Comput. Simul., 5(3):190–221, 1995.
9. Julesz B. Textons, the elements of texture perception, and their interactions. Nature, 290:91–97, March 1981.
10. Kindlmann G., Weinstein D., Lee A., Toga A., and Thompson P. Visualization of anatomic covariance tensor fields. In Proc. 26th Annual Int. Conf. of the IEEE Engineering in Medicine and Biology Society 2004, pp. 1842–1845.
11. Levkowitz H. Color icons: merging color and texture perception for integrated visualization of multiple parameters. In Proc. IEEE Conf. on Visualization, 1991, pp. 164–170.
12. Perlman J. Visualizing network security events using compound glyphs from a service-oriented perspective. Available at: http://www.csee.umbc.edu/gavl/theses/jpearlman.pdf, 2007.
13. Pickett R.M. and Grinstein G.G. Iconographic displays for visualizing multidimensional data. In Proc. 1988 IEEE Conf. on Systems, Man and Cybernetics, 1988, pp. 514–518.
14. Pickett R.M., Grinstein G., Levkowitz H., and Smith S. Harnessing preattentive perceptual processes in visualization. Perceptual Issues in Visualization, Springer, NY, 1995, pp. 33–45.
15. Pinkney D. Intelligent Iconic Visualization. Ph.D thesis, University of Massachusetts, Lowell, 1997.
16. Smith S., Bergeron R., and Grinstein G. Stereophonic and Surface Sound Generation for Exploratory Data Analysis. In Proc. SIGCHI Conf. on Human Factors in Computing Systems, 1990.
17. Ward M. A taxonomy of glyph placement strategies for multidimensional data visualization. Inf. Vis., 1:194–210, 2002.
18. Wittenbrink C., Pang A., and Lodha S. Glyphs for visualizing uncertainty in vector fields. Trans. Vis. Comput. Graph., 2 (3):266–279, 1996.
19. Yang F., Goodell H., Pickett R., Bobrow R., Baumann A., Gee A., and Grinstein G.G. Data exploration combining kinetic and static visualization displays. In Proc. 4th Int. Conf. on Coordinated & Multiple Views in Exploratory Visualization, 2006, pp. 21–30.
20. Zhang X. and Pazner M. The icon imagemap technique for multivariate geospatial data visualization: approach and software. Cartogr. Geogr. Inf. Sci., 31(1):29–41, 2004.

Iconographics

▶ Iconic Displays

Icons

▶ Iconic Displays

Identity Disclosure

▶ Disclosure Risk

Identity-based Access Control

▶ Discretionary Access Control

IDF

▶ Inverse Document Frequency

IF

▶ Information Filtering

ILM

► Information Lifecycle Management (ILM)

Image

Valerie Gouet-Brunet
CNAM Paris, Paris, France

Synonyms

Multimedia; Digital image; Picture; Photograph; Synthetic image; Graphics

Definition

Image comes from *Imago* in Latin and designates funeral masks. Philosophically, an image represents the static and eternal double of a volatile or ephemeral reality. More commonly, it is a two-dimensional artifact that either records the visual appearance of physical objects, like photographs, or provides a visual representation of concepts or artificial data, like graphics or synthetic images. *Digital* images were born in the early 1920s, as a representation of a two-dimensional image using ones and zeros (binary) obtained by digital cameras, scanners or dedicated materials and softwares. They exist in different forms, as illustrated in Fig. 1. Nowadays, digital images are everywhere. They are involved in a large number of leading applications and cover various domains from medicine to video games, including architecture or robotics.

Historical Background

The first use of digital images dates back to the beginning of the twentieth century with the technological

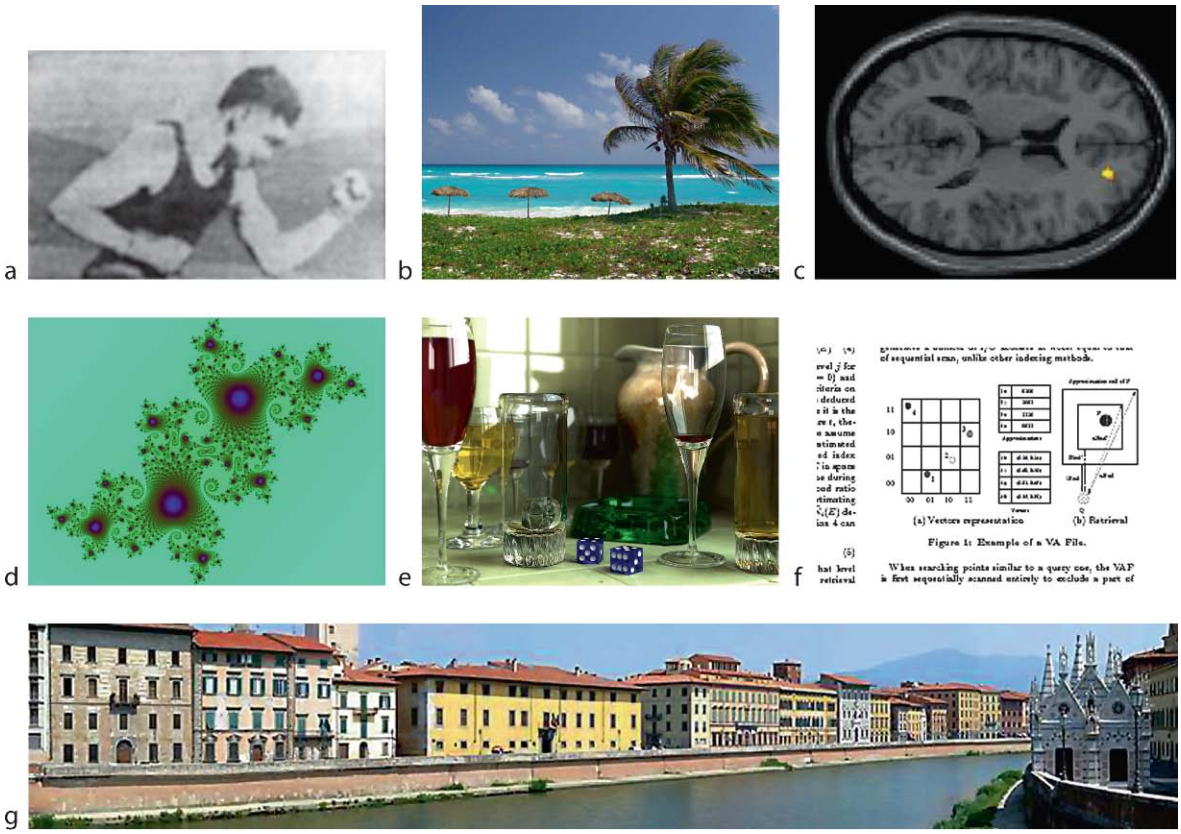


Image. Figure 1. Different representations of digital images: (a) Early digital image reproduced in 1921 by a telegraph printer with special type faces from a coded record (from McFarlane's article, 1972) (b) Photograph taken with a digital camera (c) Axial fMRI slice of a human brain (d) Synthetic image "Julia set," a fractal related to the Mandelbrot set (e) Synthetic image created by G. Tran with POV-Ray 3.6 using Radiosity (f) Zoom on a PDF document where images are vector graphics (g) Panorama stitched from 6 images with image mosaicing technique.

development of facsimile transmission for the newspaper industry. Among several systems developed for reducing picture transportation time, the Bartlane cable transmission system, invented in 1920 in Great Britain, was the first system that translated pictures into a digital code (Baudot code) for efficient picture transmission [7]; in Fig. 1, image (a) represents an image transmitted with the Bartlane system and reproduced by a telegraph printer. Scanners can be considered as the successors of telephotography input devices. The first digital image made on a computer came from a drum scanner associated with photomultiplier tubes built in 1957 by Russell Kirsch at NIST (US National Institute of Standards and Technology). But drum scanning declined and now relates to specialized applications, due to the development of low-cost flatbed and film scanners.

The advent of digital image technology is also closely tied to the development of computers as well as remote sensing applications. The first computers powerful enough to carry out meaningful image processing tasks appeared in the early 1960s, at the same time as development of space programs. With the aim of improving back transmission of images from space exploration on moon surface mapping, work using computer techniques for converting analog to digital signals began at the NASA Jet Propulsion Laboratory in Pasadena in 1964. Computer technology was advanced and pictures transmitted were processed to correct various types of image distortion inherent in the on-board television camera.

In the early 1970s, the invention of computerized axial tomography (CAT) was another important event in the development of digital imaging, in this case for medical diagnosis. From rings of sensors around the object and X-ray source, tomography consists of algorithms that construct an image that represents a slice of the object. Sir Godfrey N. Hounsfield and Prof. Allan M. Cormack shared the 1979 Nobel Prize in Medicine for this invention.

Medical research, government programs for space exploration and espionage with spy satellites clearly helped advance the science of digital imaging. However, the private sector in parallel also made significant contributions in the development of digital cameras. It first became possible to build these cameras with the availability of the CCD image sensor, invented by Willard S. Boyle and George E. Smith at Bell Labs in the early 1970s. In 1981, Sony released the Mavica, first commercial electronic still camera, followed by Canon with the RC-701 camera. Since the mid-1970s, Kodak has invented several solid-state image sensors for professional and home consumer use. In 1986, Kodak scientists invented the world's first CCD mega-pixel sensor, capable of recording 1.4 million pixels. In 1987, they released seven products for recording, storing, manipulating, transmitting, and printing still and video digital images. Since then, a large palette of digital cameras for the consumer-level market has been developed. Current research on digital cameras focuses mainly on sensors and aims at developing sensors with higher resolution, while increasing their sensitivity.

Foundations

Today, the continuing drop in prices of computers and storage, the democratization of digital images (multimedia PCs, digital cameras, cell phones, digital camcorders, etc) and the expansion of networking and communication bandwidth via the Internet or the HDTV (High-definition television) greatly contribute to the production and dissemination of larger and larger volumes of digital images. Digital images are now involved in a large number of leading applications that require the use and development of techniques belonging to several disciplines of computer science. These disciplines are presented below, after a brief introduction to the bases of digital image acquisition and representation.

Image Acquisition

Digital imaging refers to the technical process of digital image acquisition. A 2D digital image may be created directly from a physical scene by a digital camera or similar devices, as illustrated in Fig. 1 with image (b). Alternatively, it may be obtained by a scanner from a document in an analog medium, such as photographs, photographic films or printed papers. Most of these equipments have in common the digital sensors that capture light rays and convert them into electrical signals. At present, the most popular image sensors fall either in the category of charge-coupled devices (CCD) or active-pixel sensors (CMOS for Complementary Metal-Oxide-Semiconductor) for the most recent technology [6].

Unlike human vision, sensors that capture images are not limited to the visual band of the electromagnetic spectrum. Digital images can cover almost the

entire spectrum, ranging from gamma to radio waves. Many other specific equipments exist to capture 2D or 3D digital images, mainly for medical, military, astronomy or satellite domains: X-radiographs, MRI scanners (Magnetic Resonance Imaging, see image (c) in Fig. 1), PET scanners (Positron Emission Tomography), SLAR radars (Side-Looking Airborne Radars), radio telescopes, etc.

Finally, a digital image can also be computed from a geometric model or mathematical formula. In this case, it refers to 2D or 3D computer graphics discipline, as illustrated with images (d), (e), and (f) in Fig. 1.

Image Representation

Digital images can be represented either as vector graphics or bitmapped representation (also called raster representation). Vector images are geometrical 2D objects created with drawing software or CAD (computer-aided design) systems. They are represented with geometrical primitives such as points, lines, curves, and shapes or polygons, which are all based upon mathematical equations.

A bitmapped digital image is composed of a set of dots or squares, called pixels (for picture elements), arranged in a matrix of columns and rows. Each pixel has a specific color or shade of gray, and in combination with neighboring pixels it creates the illusion of a continuous tone image. Managing bitmapped images requires the manipulation of several parameters such as color model, dynamic range, and resolution.

Image Processing and Analysis

Because an image can be seen as a two-dimensional signal, image processing is a sub-field of signal processing. A lot of image processing techniques involve treating the image as a two-dimensional signal and applying standard signal processing techniques to it, while other techniques are specific to images. In the digital image community, there is not an unified definition of the tasks associated with image processing. Traditionally for some people, image processing techniques refer to image manipulation or filtering and then take an image as input and provides another image as output. For others, the output can also be a set of characteristics related to the input image, for instance either low-level features obtained by detection or extraction processes (contours, regions, etc), or high-level features obtained by analysis (face recognition, behavior interpretation, etc). According to [4], image processing gathers the three following computerized processing levels:

- Low-level process: This level involves preprocessing operations such as image geometrical transformations (sub-sampling, rotation), image compression, image enhancement (noise reduction, contrast enhancement, image sharpening, etc). Both the inputs and outputs are images.
- Mid-level process: The input is an image processed at previous level and generally the outputs are visual attributes extracted from this image. Mid-level processing on images involves tasks such as automatic extraction of meaningful features (contours, key points, regions, objects). This step is then followed by the visual description of these features to be able to make the distinction between them, and more generally to reduce them to a form suitable for higher-level computer processing.
- High-level process: This ultimate level tries to artificially perform the cognitive function usually associated with human vision. Associated with previous level, it refers to image *analysis* and *interpretation* in literature. From the features extracted at previous level, plus potential external information such as prior knowledge furnished by the application/domain or training data, the algorithms try to form a decision.

Computer Vision

As the name implies, computer vision is the science and technology that allow computers to see. Computer vision is concerned with the theory for building artificial systems that interpret images, with ultimate goal of reaching performances similar to human vision. This discipline shares many formalisms and techniques with the image processing and analysis one previously described. But in literature, image processing and analysis tends to be interested in the interpretation of 2D images, while computer vision rather focuses on the interpretation of 3D scene from its projection onto one or several images to analyze, e.g., how to reconstruct structure about the 3D scene or other geometrical information (like location of cameras or 3D trajectories of objects) from images. To achieve these tasks, one fundamental aspect of computer vision techniques is geometry and more precisely the manipulation of geometrical models [5], such as the camera

model (classically *pinhole model*) and the stereovision model (*epipolar geometry*).

The problems addressed by 3D computer vision are various [3], for example: camera calibration, i.e., the automatic determination of the characteristics of the camera from one image and a 3D pattern; auto-calibration from several images; 3D reconstruction of the scene structure from several images; determination of 3D motion/trajectory of objects in video sequences.

Image Synthesis

While scanners or digital cameras allow building digital images from physical objects, image synthesis focuses on the capability to create novel images representing realistic or artificial scenes. The traditional approaches of *computer graphics* have been to create synthesis images from a given geometric model in 3D by projecting it onto a two-dimensional image. Conversely, *image-based rendering* starts from one or several two-dimensional images in order to directly generate novel two-dimensional images, skipping the manual 3D modeling stage. Applications such as video games, motion picture, audiovisual, architecture, computer simulation, tourism (virtual travels) and e-commerce stand to benefit from these technologies. The most known associated exhibition is the SIGGRAPH annual conference (Special Interest Group on GRAPHics and Interactive Techniques) convened by the ACM SIGGRAPH organization since 1974 [12].

Computer Graphics Computer graphics is concerned with the creation and the manipulation of digital images [10]. The term can refer to 2D computer graphics, where artificial images are diagrams, logos or textures that may be generated by using vector graphics modelers (see example (f) in Fig. 1), mathematical models such as fractals (see example (d) in Fig. 1), or by processing existing images. Most of the time, computer graphics implicitly means 3D computer graphics, that renders images from the three-dimensional representation of geometric data. 3D computer graphics moves on several techniques stemmed from Computer-Aided Design (CAD):

- *3D modeling* is the process of developing a mathematical representation of any three-dimensional object that describes its appearance in terms of shape, colors, and potential textures. 3D models are often created with special softwares called 3D modelers, by an artist, a specialist or engineer or scanned into a computer from real-world objects.
- The object layout and its interactions with other objects must also be described. If the object moves or deforms, *animation* refers to its temporal description. Popular methods include inverse kinematics and motion capture; this last involves the recording of human actors actions, to animate digital character models.
- *Rendering* converts the 3D object into a image by projecting it onto a 2D image with a perspective projection model, while simulating light transport according to illumination models. The basic operations are *transport* (models for light movement) and *scattering* (models for light reflection on the object's surface). Different methods are better suited for either photo-realistic rendering, or real-time rendering. When the goal is photo-realism, the most famous global methods for rendering are *ray tracing* and *radiosity*. These techniques successfully address the problems of occlusion, transparency, shadows and take the interactions between objects into account, as illustrated with the image of Fig. 1e.

Image-Based Rendering Unlike traditional computer graphics approaches previously described in which 3D geometry of the scene is known, the aim of image-based rendering techniques is to render novel views directly from input 2D images, in a realistic way without full 3D model reconstruction [11]. Previous work on image-based rendering reveals a continuum of image-based representations based on the tradeoff between how many input images are needed and how much is known about the scene geometry; such techniques directly refer to geometric and algorithmic models of computer vision. The most popular applications of image-based rendering are *view morphing* and *image mosaicing*. The first one aims at generating intermediate views between two reference images, by interpolating parts of images in correspondence. In the 1990s, they were very popular in the audiovisual industry and motion pictures, while producing aesthetic and spectacular special effects at low cost, such as one face turning into another. The novel views rendered with such techniques do not respect the

geometry scene, in the sense that the objects may be distorted in the produced images. Image mosaicing produces panoramic images by registering or stitching multiple regular images that partially share a view of the scene; see example (g) of Fig. 1. The produced image can be projected to a cylindrical or a spherical map for visualization and virtual navigation with the feeling of immersion into the 3D world. The first and most popular application of mosaicing is QuickTimeVR [1].

Image Indexing and Retrieval

The huge volume of digital images now available on the Internet or simply on personal computers makes for images that cannot be easily located. Image indexing techniques aim at developing search engines that manage collections of images and in particular that facilitate their fast and accurate retrieval. But retrieval of images, more generally of multimedia data, is different from retrieval of structured data such as classical databases. It is necessary to exhibit a description of the images, often called *descriptor*, that represents the essence of the image for a given topic and that will be indexed to allow efficient retrieval among the many descriptors associated to the available images.

The most usual descriptors associated to images are textual, making tools of linguistics required to define them. Most of the time, this information is structured in *keywords* that can either be related to a controlled vocabulary (potentially connected to dedicated ontologies or semantic lexicons) or be freely assigned [13]. Keywords can be determined by analyzing the metadata associated with the image, the filename of the image, by parsing text adjacent to the image as Google Image Search does on the Internet, or by using more sophisticated methods such as text extraction from images content. Otherwise, keywords are designed manually by experts of the application domain. A popular new form of free-text keywords on the Internet is *folksonomy* where tags are assigned to images by non-experts or consumers collaboratively. Once the images are described with keywords, evaluating a search query to quickly locate images characterized with given words among a collection of images is done by exploiting an index data structure; inverted files are a popular technique for indexing text data [9].

Text-based indexing has several shortcomings such as ambiguities, subjectivity, language or context-dependence. Indexing the visual content of images is a recent alternative that may help to reduce these drawbacks by giving additional insight into the collections of images. When the number of images makes manual annotation unachievable, it is the only solution. Born in early 1990s, Content-Based Image Indexing (CBIR) is a discipline that exploits techniques of image processing/analysis in addition to databases tools [2]. Indexing an image by its content begins by extracting visual structures that describe the visual content relevantly for the considered application. Such structures are considered as the index of the image, which is digitally represented with one or several multidimensional vectors called *signature* of the image. Here, multidimensional index structures are required to perform retrieval efficiently in terms of processing time and disk access [9].

Key Applications

A classical problem in image processing, image analysis, computer vision and CBIR is the ability to determine whether or not an image contains some specific objects. This task, referred to *object recognition*, can normally be solved robustly and without effort by a human, but is still not satisfactorily solved by computers for the general case of arbitrary objects in arbitrary situations. The existing methods can at best solve it only for specific objects such as print or handwritten characters, human faces or vehicles and in specific situations, typically described in terms of well-defined illumination, background, and pose of the objects relative to the camera. Thus, while object recognition has been an active research area for 40 years [8], it is still challenging for several applications. Nevertheless, some patented approaches for specific object recognition have already proved their efficiency, they are offered as a service by consumer-oriented companies or are well established in several organizations and apply to a large palette of domains. For instance:

- *Surveillance of road traffic:* Automatic recognition of license plates is a mass surveillance method that uses character recognition on images to read plate numbers on vehicles. Many governments, police, and town councils have adopted this technology to automatically detect vehicles in driving offence, to supervise areas such as parking areas, or to control traffic in order to help monitor the

movements and flows of vehicles around a road network.

- *Authentication:* Biometrics technologies provide methods for uniquely identifying humans based upon one or more intrinsic physical or behavioral traits. Systems based on visual appearance analysis are based on particular methods of image analysis dedicated to specific images representing faces, retinas, hands, veins or fingerprints. For example, several banks in Japan have adopted palm vein authentication technology on their ATMs (technology developed by Fujitsu in 2003 and called PalmSecure). Among several governments, Israel has also adopted this technology to control the border crossing points between Israel and the Gaza Strip: an ID card, with stored biometrics of fingerprints, facial geometry and hand geometry, is required to go across.
- *E-commerce:* Since few years, several companies provide to their clients the ability to link their products and services directly to specific web pages on the mobile internet by the way of bi-dimensional barcodes called "smart codes." With their mobile phone's camera, consumers can scan the code placed near or on the product, then the code is recognized by image analysis, making it possible the connection to the associated service (product purchasing, timetables consultation, prices comparison, etc).

Cross-references

- ▶ Annotation-Based Image Retrieval
- ▶ Feature Extraction for Content-Based Image Retrieval
- ▶ Icon
- ▶ Image Content
- ▶ Image Content Modeling
- ▶ Image Database
- ▶ Image Querying
- ▶ Image Representation
- ▶ Image Retrieval
- ▶ Indexing and Similarity Search
- ▶ Multimedia Data
- ▶ Multimedia Databases
- ▶ Object Detection and Recognition
- ▶ Object Recognition
- ▶ Video
- ▶ Video Representation
- ▶ Visual Content Analysis
- ▶ Visual Representation

Recommended Reading

1. Chen S.E. QuickTimeVR: an image-based approach to virtual environment navigation. In Proc. Int. Conf. on Computer Graphics and Interactive Techniques, 1995, pp. 29–38.
2. Datta R., Joshi D., Li J., and Wang J. Z. Image retrieval: Ideas, influences, and trends of the new age. ACM Comput. Surv. 40(2), 2008, paper 5.
3. Forsyth D. and Ponce J. Computer Vision - A modern approach. Prentice Hall, 2002.
4. Gonzalez R. and Woods R. Digital Image Processing. Prentice Hall, 3rd edition, 2008.
5. Hartley R.I. and Zisserman A. Multiple view Geometry in Computer Vision. Cambridge University Press, 2nd edition, 2004.
6. Litwiller D. CCD vs. CMOS: Facts and Fiction. Photonics Spectra, January 2001.
7. McFarlane M.D. Digital pictures fifty years ago. Proc. IEEE, 60 (7):768–770, July 1972.
8. Ponce J., Hebert M., Schmid C., and Zisserman A. Toward Category-level Object Recognition, LNCS, vol. 4170. Springer, 2007.
9. Samet H. Foundations of Multidimensional and Metric Data Structures. Morgan Kaufmann 2006.
10. Shirley P., Ashikhmin M., Gleicher M., Marschner S., Reinhard E., Sung K., Thompson W., and Willemsen P. Fundamentals of Computer Graphics. A.K. Peters, Ltd., 2nd edition, July 2005.
11. Shum H. Y., Chan S. C., and Kang S. B. Image-Based Rendering. Springer, 2006.
12. ACM SIGGRAPH. http://www.siggraph.org/.
13. Yu L. Introduction to the Semantic Web and Semantic Web Services. Chapman & Hall/CRC, 2007.

Image Classification

- ▶ Automatic Image Annotation

Image Compression

- ▶ Image Representation

Image Content

- ▶ Image Representation

Image Content Modeling

Harald Kosch, Mario Döller
University of Passau, Passau, Germany

Synonyms

Image data model; Image meta-data; Conceptual image data model

Definition

Image Content Modeling deals with the issue of representing the content of image data; that is, designing the high- and low-level abstraction model of the raw image objects and their correlations to facilitate various operations. These operations may include media object selection, insertion, editing, indexing, browsing, querying, retrieval, and exchange. The image content model relies, therefore, on the extraction of feature vectors and their respective representations obtained during the annotation process. Several standards for representing the content of an image are known. The most prominent ones are MPEG-7 including low- and high-level abstractions or the EXIF data description vocabulary for the description of very specific technical attributes of an image.

Historical Background

Many models for describing the content of an image have been created in the past. Most of them rely on standards. Many standards are actually in use, mainly due to *different national and organizational interests* of service providers, standardization bodies, and usage groups. Whereas each standard works well in its dedicated application domain, problems arise in frequently occurring cross-domain working environments. The following is a non-exhaustive list of important standards:

- The *Exchangeable Image File (EXIF)* (http://www.exif.org/) format is an international specification that allows imaging companies to encode meta data information into the headers or application segments of a JPEG file. This meta data information includes shutter speed, aperture, date and time of the captured image. Current digital cameras store images using EXIF compressed files. It is a widespread standard, with a simple attribute/value specification. EXIF was last revised in April 2002.
- The *DIG35* Initiative Group of the I3A (International Imaging Industry Association) has also defined a meta data standard for digital images (http://www.i3a.org/i_dig35.html). The *DIG35 Specification* includes technical meta data on media, simple semantic meta data on persons, locations, events and intellectual property and rights management related meta data. The focus of DIG35 is on retrieval, categorizing and browsing of large image archives. The latest version stems from March 2000.
- The *SMPTE* (http://www.smpte-ra.org/mdd/) is a standardization organization that has created the Metadata Dictionary (MDD). Most meta data consists of media-specific attributes, such as timing information as well as some basic semantic descriptions (interpretation, administration and relationships among descriptions). The MDD definition was reviewed in 2004. The SMPTE Material eXchange Format (MXF) allows users to take advantage of non-real-time transfers, and to package together essence and metadata for effective interchange between servers and between businesses.
- The *P/Meta EBU P/Meta Project* (http://www.ebu.ch/metadata/pmeta/v0100/html/start_frame.html) has created the P_META Scheme, a set of definitions that provides a semantic framework for information which is typically exchanged along with audio-visual material. It includes the identification of concepts referenced by other elements. The P_META Scheme has been created for the use in a business-to-business scenario where the participating organizations may retain their internal data structures, workflows, and concepts. Version 1.0 appeared in 2002.
- The MPEG-7 (http://www.chiariglione.org/mpeg/standards/mpeg-7/mpeg-7.htm) is the MPEG standard for description and search of audio and visual content and is the first standard from MPEG which considers multimedia content models. From the point-of-view of the expressiveness of the proposed metadata descriptions it is the richest set, as it enables the user to describe structural, as well as semantic content of an image [3]. It supplies likewise means for the description of creation, usage and management of images. MPEG-7 version 1 was released as international standard in 2002. An improved version was published in 2006.

MPEG-7 visual description tools cover color, texture, shape and face recognition descriptors for images. They mainly use a histogram-based approach of representation; that is, they compute a

vector of elements each representing the number of pixels (regions) in a given image, which have similar characteristics. For instance, the color structure descriptor computes a histogram obtained by accounting all colors in a structuring element which slides over the images and takes therefore the spatial layout better into consideration, or the color layout descriptor which uses a discrete cosine transformation to represent more compactly the histogram. MPEG-7 high-level descriptions mainly represent the structure and semantics of an image. The media decomposition tools allow the recursive sub-division of an image into regions of interest. A decomposition tree describes the image source and the spatial structure of the image regions and can be used to create a table of content. Each region can be associated with its shape information and low-level visual descriptions, then information on creation, rights and usage, simple text annotation (what, where, when, why, how), and semantic descriptions. Semantic descriptions represent a narrative world as a set of semantic entities and semantic relations among semantic entities, and segments. Possible semantic entities are objects, agent objects, events, places, and time. MPEG-7 has been employed in many retrieval and database systems. Westermann et al. gives an overview of MPEG-7 based database solutions [11] and provides a persistent DOM (Document Object Model) model that can be used for MPEG-7 systems solutions. A full-fledged database system, MPEG-7 MMDB is described in [1].

The general goal of achieving *image content data interoperability* is affirmed by many researchers. It is expressed by panels and tutorials held at various international conferences, such as ACM Multimedia 2002 and 2004 (See, http://mm02.eurecom.fr/panel.html and http://www.mm2004.org/acm_mm04_call4tutorials.htm) or by input contributions to standardization (e.g., recently proposed to W3C as Semantic Web Image Annotation Interoperability (http://www.w3.org/2001/sw/BestPractices/MM/interop.html)). It has been clearly recognized that missing interoperability is still a major burden for an effective use of image content data in industrial media engineering.

Foundations

Keywords are by far the predominant features used to describe the content of image data. An indexer using keywords or a textual abstract describes the content of the image [6]. Another method, *content-based indexing*, refers to the actual content of the image data. Intensive research has focused on content-based indexing in recent years, with the goal of indexing the image data using certain features derived directly from the data [10]. These features are also referred to as *low-level features*. Examples of low-level visual features for images are color, shape, texture, spatial information, and symbolic strings. The extraction of such features may be done by an automatic analysis of the image data. Indexing *high-level features* (e.g., objects, events, etc.) of image data is an active research area [5]. Different detection mechanisms have been proposed for segmenting the image materials into regions of interest and for attributing semantic meaningful annotation to each region and (their semantic) relationships. In addition to this, and related to the previous issues, ongoing research concentrates on image classification. For example, is this image showing a sport event? And more specifically, is this sport image about basketball? The recognition of an image as basketball facilitates the semantic annotation process, as one knows what kind of objects and persons might appear.

Image Content Modeling refers now to the design of a high- and low-level abstraction model of the raw image objects and their correlations to facilitate various operations, such as media object selection, insertion, editing, indexing, browsing, querying, retrieval, and exchange. Several abstract models have been proposed in the literature [3]. Most of them deal with special low- and high-level features. At least two models introduce a general framework to cover the whole spectrum of available features, the DISIMA model [7] and the indexing pyramid [2]. The later one served as conceptual model for the semantic part of the MPEG-7 standard.

The DISIMA model relies on two main concepts, namely the image and salient objects (logical and physical) and operators to manipulate them [7]. Principally, the image type defines the content of an image as a set of physical salient objects that are regions of the image. The semantic meaning of a physical salient object is represented by a logical salient object. Oria et al. define several properties of physical salient objects, such as color, texture, and shape. The logical salient object gives semantics to a physical salient object; proposed examples are objects, persons and so on. Predicates for comparing objects (e.g., different similarity operators, spatial operator) and a special contain operator which

computes if an object is contained in an image, are proposed. As an image can be composed of more than one physical object, logical objects can be found several times. The model can be queried by Visual MOQL, an image query language based on OQL, proposed in a previous paper of [7].

Jörgensen et al. [2] introduced a representation model based on the indexing pyramid for classifying levels of indexing. The pyramid, as shown in Fig. 1, distinguishes between *syntactic* (first four levels) and *semantic* levels (next six levels). The syntactic levels hold attributes that describe the way in which the content is organized, but not their meanings. This corresponds to the definition of low-level features. In images, level 1 (type technique) could be "color image." Global distribution holds attributes that are global to the image (e.g., color histogram), whereas local structure deals with local components (e.g., lines and circles), and global composition relates to the way in which those local components are arranged in the image (e.g., symmetry). The semantic levels deal with the meaning of the elements. This corresponds to the definition of high-level features. With each level, more knowledge is required to represent the objects. They can be described at three levels:

1. Generic: every day objects (e.g., person)
2. Specific: individually named objects (e.g., Roger Moore)
3. Abstract: representing the highest semantic abstraction (e.g., power)

In a similar way, a scene (composition of elements) can be described at these three levels.

Note that the authors of the indexing pyramid also propose a retrieval system that allows a search specific to the levels of the pyramid. For instance, if a user enters "soccer" for image search at the syntactical level, one retrieves images with a description containing the keyword soccer (which does not yet mean that a soccer game is shown in the image). This is the same mechanism as the retrieval engine that Google uses (http://images.google.com). If a user enters soccer at the semantic level, only those images are retained in which the event, "soccer," took place. An online demo is available at http://www.ee.columbia.edu/~ana/mpeg-7/.

Image Content Modeling is dominated by the *use of standards*, as already mentioned in the historical considerations before. Two types of standards have to be considered. First those which intend to specify a standard way of describing the content of image data mainly for information search and exchange. Second those which aim to be used for image storage, manipulation and query. In the first category, MPEG-7 is an important representative [3]. MPEG-7 is introduced in the historical background, and is detailed in the multimedia/image metadata entries. An example MPEG-7 document is given in Fig. 2. In the second category, SQL/MM, recently been integrated into Oracle Multimedia for instance, is representative [8]. SQL/MM will be detailed in the next paragraphs.

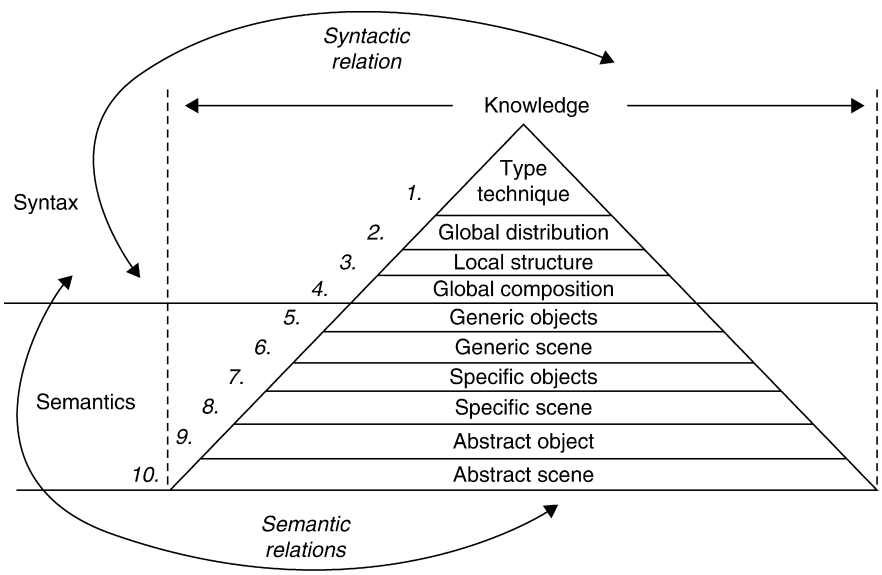


Image Content Modeling. Figure 1. Indexing Pyramid as base for MPEG-7.

SQL/MM as Conceptual Image Data Model for Databases ISO/IEC developed the SQL/MM Multimedia extensions to SQL to deal with image data in a database management system [8]. SQL/MM has been recently introduced to several commercial systems. It introduces new object data types, in the center are the *SI_StillImageType* and related feature vector representations. Principally, one color histogram (*SI_ColorHistogram*), and two non histogram color features are proposed, an average color (*SI_AverageColor*) and an array of dominant colors (*SI_PositionalColor*). The texture feature (*SI_Texture*) contains values that represent the image texture characteristics. The standard defines in addition a composite feature (*SI_FeatureList*) containing up to four different basic features and associated feature weights. A weight value specifies the importance attributed to a particular feature during image matching. The weight value varies from 0.0 to 1.0. A feature weight value of 0.0 indicates that the feature is not considered for image matching.

```
<Mpeg7>
  <Description xsi:type="ContentEntityType">
     <MultimediaContent xsi:type="ImageType">
        <Image>
          <Semantic>
             <Label>
                <Name> Anne takes a photo of the Leaning Tower </Name>
             </Label>
             <SemanticBase xsi:type="EventType" id="TakingPhoto">
                <Label>
                  <Name> Description of the event of taking a photo </Name>
                </Label>
                <Relation type="urn:mpeg:mpeg7:cs:SemanticRelationCS:2001:agent"
                     target="#Arme"/>
                <Relation type="urn:mpeg:mpeg7:cs:Semant.icRelationCS:2001:source"
                     target="#Tower"/>
                <SemanticPlace>
                  <Label>
                     <Name>Pisa</Name>
                  </Label>
                </SemanticPlace>
                <SemanticTime>
                   <Label>
                     <Name>September 3, 2007</Name>
                   </Label>
                </SemanticTime>
             </SemanticBase>
             <SemanticBase xsi:type="objectType" id="Tower">
                <Label>
                  <Name> Leaning Tower </Name>
                </Label>
             </SemanticBase>
             <SemanticBase xsi:type="AgentobjectType" id="Arme">
                <Label>
                  <Name> Anne </Name>
                </Label>
                   <Agent xsi:type="PersonType">
                   <Name>
                     <GivenName> Anne </GivenName>
                   </Name>
                </Agent>
             </SemanticBase>
          </Semantic>
        </Image>
     </MultimediaContent>
  </Description>
</Mpeg7>
```

Image Content Modeling. Figure 2. MPEG-7 Example Document.

It can be clearly seen that the SQL/MM data model covers only parts of the syntactical levels of the indexing pyramid proposed in [2]. For instance no means for describing the decomposition of an image into region of interests is given.

For instance, a photo book can be described by the following table definition:

```
CREATE TABLE PHOTOBOOK(
PHOTO_ID NUMBER(6), // id for identification
PHOTO SI_StillImage, // the image itself
AVERAGE_COLOR SI_AverageColor, // here comes the list of
COLOR_HISTOGRAM SI_ColorHistogram, // features for querying
FEATURE_LIST SI_FeatureList,
POSITIONAL_COLOR SI_PositionalColor,
TEXTURE SI_Texture,
COMMIT;
```

Key Applications

Key applications for image content modeling are image databases and retrieval systems. SQL/MM is the standard for the use in commercial database management systems. Its simple conceptual model does not allow the use of complex low-level representations, nor high-level descriptions. Instead, a recent research issue is on image meta data databases, which for instances expresses in the search for a good mapping of an image content model into database schemes (see e.g., MPEG-7 databases in [1,11]).

Cross-references

► Image Database
► Multimedia Metadata

Recommended Reading

1. Döller M. and Kosch H. The MPEG-7 Multimedia database system (MPEG-7 MMDB). J. Syst. Software, 81(9):1559–1580, 2008.
2. Jörgensen C., Jaimes A., Benitez A.B., and Chang S.-F. A conceptual framework and empirical research for classifying visual descriptors. J. Am. Soc. Inf. Sci. Technol., 52(11):938–947, 2001.
3. Kosch H. Distributed Multimedia Database Technologies supported by MPEG-7 and MPEG-21. CRC Press, Boca Raton, FL, 2004.
4. Kosch H., Böszörményi L., Döller M., Kofler A., Schojer P., and Libsie M. The life cycle of multimedia meta-data. IEEE Multimedia, 12(1):80–86, 2005.
5. Liu Y., Zhang D., Lu G., and Ma W.Y. A survey of content-based image retrieval with high-level semantics. Pattern Recognition, 40(1):262–282, 2007.
6. Lu G. Multimedia Database Management Systems. Artech House, London, UK, 1999.
7. Oria V., Özsu M.T., and Iglinski P.J. Foundation of the DISIMA image query languages. Multimedia Tools Appl. J., 23:185–201, 2004.
8. Stolze K. Still image extensions in database systems – A product overview. Datenbank-Spektrum, 2:40–47, 2002.
9. Tseng B.L., Lin C.-Y., and Smith J.R. Using MPEG-7 and MPEG-21 for personalizing video. IEEE Multimedia, 11(1):42–52, 2004.
10. Veltkamp R.C. and Mirela Tanase. Content-Based Image Retrieval Systems: A Survey. Technical Report, Utrecht University, The Netherlands, 2000.
11. Westermann U. and Klas W. An analysis of XML database solutions for the management of MPEG-7 media descriptions. ACM Computing Surv., 35(4):331–373, 2003.

Image Data Model

► Image Content Modeling

Image Database

Mario Döller[1], Harald Kosch[1], Paul Maier[2]
[1]University of Passau, Passau, Germany
[2]Technical University of Munich, Munich, Germany

Synonyms

Image retrieval; Image retrieval system; Content-based image retrieval (CBIR)

Definition

Given a collection of images, a full-fledged image database provides means and technologies that support an efficient and rich modeling, storing, indexing, retrieval, and manipulation of images and its meta-data. The modeling of images can range depending on the used meta-data format (e.g., MPEG-7) from simple technical annotations such as file size, creator, etc. to more sophisticated annotations such as low level features (e.g., color) or even high level features (e.g., objects, events, etc.). The storing component is responsible for mapping the used meta-data format to an adequate database schema. Indexing facilities should support efficient retrieval and need to provide means (depending on the used meta-data) for indexing text, multidimensional feature vectors and high level representations. The retrieval and query specification should support some or all of the following

concepts: query by example, query by sketch, query by textual information (e.g., creator), query by category browsing, query by concept. Finally, manipulation of images should allow operations such as rotating, shrinking, format conversion, thumbnail creation, feature extraction, etc.

Historical Background

Activities in storing and retrieving images within databases can be traced back to the late 1970s, where first conference (e.g., Data Base Techniques for Pictorial Applications, 1979) contributions introduced the use of relational databases for images [2]. In general, these early works focused on the annotation and retrieval of images by textual information. For this purpose, the images were described by keywords or textual descriptions and common relational database technologies and their text-based retrieval approaches were used for searching within the pool of annotated images. A substantial survey for text-based image retrieval can be found in [11]. The main drawbacks of these early-approaches concern the annotation of images by text. In general, manual annotation of images and its content is very time consuming and for large image sets infeasible. Furthermore, the assignment of descriptions or keywords to images is not based on the use of a controlled and universally valid vocabulary or classification scheme, and is highly affected by cultural, subjective and domain specific influences. First discussions on this topic were raised by Shatford [9] which indirectly resulted in various thesaurus catalogues (e.g., thesaurus for graphical material, Art and Architecture Thesaurus (AAT), etc.).

Based on the identified drawbacks of text-based image retrieval, researches tended to move in the late 1990s to investigate the retrieval of images by content. It is often referred in the literature as content-based image retrieval (CBIR) and concentrated on technologies for the extraction, indexing, comparison and query of images by their low level features. These low level features include representations for color, texture and shape which can be extracted automatically.

In general, a typical CBIR system covers the following main steps. First, the low level features of all searchable images are extracted and stored as high-dimensional feature vectors in a database. Then, in order to improve retrieval efficiency, the feature vectors are indexed by specific access methods which are tuned for that kind of data (see [3] for a survey of existing access methods). Finally, during the query process, an example image (represented by its features) is provided by the client. Based on this, the similarity/distance in relation to the stored features is calculated. The size of the result set depends on the chosen query type (range query or nearest neighbor) and contains the most similar (in terms of the low level features) images according to the given one.

Examples of early successful systems and applications focusing on CBIR were, for instance, the IBM QBIC system (which stands for Query by Image Context), Blobworld or CalPhotos. The CalPhotos image retrieval system had its roots in a 1993–1994 research project called Chabot, at the University of Berkeley. R. C. Veltkamp and M.Tanase compared (updated in 2004) 43 available products [12]. Smeulders et al. surveyed the early years of CBIR systems in [10] with 200 references.

Foundations

As already outlined in the historical background section, the search within image collections has a long tradition in the field of database management systems. In the following, the main components of an image database are introduced in order to support the storage, indexing, and retrieval of images.

Image Database Schema

First approaches transferred meta-data of raw media data into database relations in an ad-hoc way. These simple models only support certain types of queries and operations and were mostly limited to keyword retrieval. The nature of multimedia data, consisting of alphanumeric, graphical, image, video, and audio objects, differs in many ways to simple alphanumeric data that relational database management system are able to handle. One of the first models proposed for visual information that considers semantic queries, was called the Visual Information Management System (VIMSYS) [4] from Virage. The creators recognized within their model that image and video information is preferred to be retrieved by content rather than by keywords or additional textual descriptions. During the past years, research concentrated on the development of data models for images (e.g., [1]) and content-based retrieval. In most cases, these systems concentrate on the retrieval based on low-level features. Recently, data models for image information that support high-level features have been introduced. The data models of these systems are

often specialized to one type of genre, e.g., retrieval on the basis of soccer games (e.g., [5]).

Index Structures

Besides the necessity of developing semantically rich image data models and database schemas, an efficient indexing and search of images is essential. During the last decade, image based retrieval concentrated on the use of low-level features and their feature vector representations.

In order to improve search efficiency, different access methods have been established for indexing multidimensional feature vectors, for instance (to mention just a few) LPC-File, X-tree, Pyramid Technique, R-tree and its variants, SR-tree, M-tree, etc. An excellent survey and overview paper of index structures is given by e.g., Gaede et al. [3]. An evaluation of different quantization techniques, such as A-tree versus IQ-tree is given by Garcia-Arellano (University of Toronto) in his master thesis. A complete generalized framework, called GiST, has been established by J. Hellerstein and his group at the University of California, Berkeley. It consists of a set of pre-implemented index structures and generalized access interfaces. Support is given by sophisticated graphical tools providing access statistics. The framework has been used for various image retrieval applications (e.g., Blobworld) and the development of new index structures. In addition, the framework has been integrated within research projects in the Informix database and Oracle database.

One problem in processing feature vectors is their high dimensionality. The quality of results of a similarity search depends on a high degree on the underlying data set (feature vectors). For instance, the more colors a color histogram represents the merrier the similarity search will work. This correlates with a higher dimensionality of the resulting feature vector. This is also the case, when multiple features are combined, such as for images the color, the shape and the texture. Unfortunately, query performance of access methods decreases, if dimensionality of the underlying data set becomes high. This phenomenon is known as *curse of dimensionality* [3]. One possibility of avoiding this problem is the development of specific index structures that are specialized for high-dimensional data (e.g., see [3] for a survey). Another possibility is the reduction of the vector dimension. The main goal of reducing the dimension is to find a low-dimensional representation of the vector that preserves (most of) the information of the original data. The reduction can be realized by two different approaches: *Feature selection* and *Feature extraction*. *Feature selection* stands for choosing a subset of all the features that represent most of the desired information. *Feature extraction* describes the creation of new features by combining the existing ones. Wu et al. [13] introduced a dimensionality reduction for image retrieval called *weighted multidimensional scaling*. Another commonly used technology is the *singular value decomposition* (SVD).

Query Optimization

A general architecture of a query optimizer has commonly the following form [1]: A query is forwarded to a *query parser* which checks the syntactical correctness and transforms the query into an internal representation (mostly an algebraic expression). In the following step, the *query optimizer* evaluates the most effective algebraic expressions which represents the given query and chooses the cheapest one. The *code generator* transforms the resulting query plan into calls to the *query processor*, which does the actual query execution. The main task of a query optimizer is to examine all possible alternatives to a given query so that the best alternative can be chosen and the query cost is minimized.

In image databases, queries often contain similarity operations, such as Range or Nearest Neighbor (NN)-operation for low-level features (e.g., color histogram), especially if the system supports content-based retrieval. This can be associated with a similarity-based selection operation in an image database management system. In general, a query optimizer should use three approaches: *selectivity*, *cost model*, or *operator ordering*. Most approaches concentrate on the cost-model and the selectivity, based on the fact that modern database systems, such as Oracle, only provide means for their enhancement.

The cost of a selection operation is in general composed of two different cost factors. The *selectivity* of an operation defines the amount of tuples returned by the selection operation. Whereas, the *cost* of an operation is counted as the amount of data pages which has to be accessed for fulfilling the given task and as the number of necessary CPU cycles. In the image environment, the cost (selectivity, and amount of data pages) of Range- and k-NN-queries in high-dimensional spaces is important. An efficient processing of such kind of queries

is guaranteed by a multidimensional index. Therefore, the calculation of the amount of accessed pages (further simply called cost) needs an evaluation of the used index structure. This evaluation can be simple in one case, for instance the selectivity of a k-NN query is determined through k, or difficult, if the selectivity for a Range query is considered which needs an approximation. The cost of a Range query can be easily calculated, based on the known radius which results in an intersection operation of the hypersphere and the minimum bounding rectangles (MBRs) of the index structure. The cost for a k-NN query depends largely on the density around the query location.

In the literature, several cost models exist that concentrate on calculating the amount of page access for Range- and k-NN-queries (e.g., [6]). Selectivity estimation for Range queries is often limited to either the assumption of uniformity and independence of data sets or to the 2- or 3- dimensional data spaces in geographic information systems. The uniformity assumption does not hold for real data sets. In [6], the authors present an efficient cost model for predicting the performance of the k-NN-query independently of the used index tree. The model is accurate for low- and mid-dimensional data with non-uniform distributions. For this purpose, the authors introduce two new concepts: the regional average volume and the density function. The regional average volume is used for calculating the average radius of the hyper-sphere which contains the k-NNs. As indicated before, the average radius is not adequate for highly skewed non-uniform distributions. To overcome this limitation, the authors introduced a density function that estimates the regional radius depending on the location in the data space.

Database Extensions to Support Images

In the last decades, traditional (Object-) Relational Database Management Systems ((O)RDBMS) have been very effective and efficient in managing alphanumerical data. They basically offer services such as indexing, query optimization, buffer management, recovery or concurrency control, which are well investigated and optimized. Unfortunately, common ORDBMS and their techniques have several drawbacks in handling image data. The most important drawback concerns the concept of matching. In traditional databases, the concept of matching relies on a filtering operation which decides for every tuple whether it fits the requirements or not. Basically, the main retrieval paradigm in image data repositories is similarity search. Therefore, extensions to ORDBMS mainly concern the implementation of this search type and for ranking the results, as well as for supporting index structures and query optimization. In the following subparts, various topics for image extensions are addressed.

Query Extension for Multimedia: SQL/MM Traditional query languages, such as SQL, do not provide means for handling multimedia types and their appropriate operations to satisfy the requirements in retrieving multimedia data. Therefore, several query languages (e.g., MOQL, MMDOC-QL, etc.) have been introduced by researchers in order to overcome these limitations. One standard which has been developed explicitly for image data is SQL/MM.

The SQL/MM [8] standard (MM for MultiMedia) is an extension of SQL for supporting any kind of multimedia data. SQL/MM is a multi-part standard and was developed by an ISO subcommittee, namely JTC1/SC32. The various parts of SQL/MM are independent from one another; e.g., Part I (framework) contains definitions of common concepts that are used through out the other parts or *Still Image* (Part V) is all about image data. The SQL/MM Still Image part provides structured user defined types that allow the storage of images within databases and specifies methods for modifying them in various ways and retrieving them efficiently. Images in SQL/MM are represented by the *SI_StillImage* structured data type. The *SI_StillImage* type can store images of several formats depending on what the underlying system supports. Furthermore, it extracts information about each image, such as format, height and width in pixels, color space, etc. In addition, the type provides several methods e.g., for scaling, for rotating or for creating thumbnails of the original image. Moreover, there exist various additional data types for describing several features of images. For instance, the *SI_AverageColor* type represents the average color of a given image. The *SI_ColorHistogram* type provides information about the distribution of colors within the image. The *SI_PositionalColor* type represents the location of specific colors in an image. The SQL/MM Still Image part provides CBIR functionality by combining these types and methods with accurate index structures. For instance, the *SI_PositionalColor* type supports queries like "Give me all images with a color representation of red or orange above dark blue."

This query would lead to images that in most cases show sunsets at sea.

Multimedia Extensions in Products

Oracle Multimedia The extensibility service provided by Oracle is called *Data Cartridge*. Oracle databases are built as a modular architecture which provides several extensible services. The common way for using Oracle's Extensibility Services is to implement a Data Cartridge that extends the extensibility interface.

Currently, several Data Cartridges have been developed to support multimedia data. Oracle itself provides among others, Cartridges for spatial data and for multimedia data such as image, audio and video. Cartridges of other vendors are for instance, the *Viisage Cartridge* from Virage which provides face recognition on behalf of images. This company provides several face recognition tools for controlling access to restricted areas e.g., Berlin Airport or identifying criminals in a crowd of people.

IBM DB2 The IBM DB2 database provides *Extenders* for enhancing their database management system to meet new requirements, e.g., the storage and retrieval of multimedia data. DB2 Extenders generally specify UDT's (user-defined types) for extending the core database types. At present, there exist a large number of available extenders from different vendors. For instance, IBM itself provides the *AIV* extender for audio, image and video data. Extenders of other vendors in this area are for instance, the *Spatial Data Extender* from Environmental Systems Research Institute (ESRI) for GIS and geo-spatial data or the *SpatialWare Extender* from MapInfo for spatial data. SQLSummit provides an accurate list of Extenders.

IBM Informix The IBM Informix database management system deploys the feature of *DataBlade Modules* to extend their functionality for new requirements. This DBMS provides three different tools for developing DataBlade modules, namely *BladeSmith* for creating DataBlade modules, *BladePack* for packaging them, and *BladeManager* for making them available in the database. Up to now, several DataBlade modules exist such as *Spatial DataBlade module* for managing spatial data inside the database, *Excalibur Image DataBlade module* for storing and querying images and special vendor products, like DataBlades for MPEG-1.

Image Retrieval Systems from Scratch In contrast to database extensions such as Oracle's Multimedia for managing multimedia data, several image retrieval systems have been developed from scratch (e.g., DISIMA which is an image database management system developed at the University of Alberta). R. C. Veltkamp and M. Tanase compared (updated in 2004) 43 available products [5]. A survey of CBIR systems supporting high level semantics can be found in [7]. In the following, two currently active systems are introduced:

- DISIMA is a pure image database that considers images as a set of salient objects (regions of interest). Salient objects are classified according to a user defined type hierarchy. Other currently active systems are for instance:
- Behold is a large scale image retrieval system and has been developed at the Imperial College in London. It provides means for text-based search and NN-search based on a parameterized similarity metric.

Key Applications

Surveillance Systems

The detection of criminals, wanted persons and known terrorists on airports or border stations bases on face detection algorithms and large scale image (showing persons and/or faces) databases that support the retrieval among millions of data entries.

Medical Diagnosis Supporting Systems

Recent research concentrates on applying image database in the medical domain. Every hospital has a large set of x-rays showing a diversity of diseases. These image collections are used for supporting doctors in completing their diagnosis. For instance, in case of a cancer patient, the doctor can query the image database for similar x-rays based on the patient's.

Web Search

Finding meaningful and high quality images on the web is a goal which appeals to many different people, private and professional alike. The amount of images available on the web is vast, so efficient retrieval mechanisms like CBIR are essential. Research efforts in this direction are for example the before mentioned system Behold (see section, Image Retrieval Systems from scratch).

Cross-references

► Image Content Modeling
► Nearest Neighbor Query

Recommended Reading

1. Besufekad S.A. Modélisation et traitement de requêtes images complexes. PhD Thesis, L'Institut National des Sciences Appliquées de Lyon, 2003.
2. Chang N.S. and Fu K.S. Query by pictorial example. IEEE Trans. Softw. Eng., 6(6):519–524, 1980.
3. Gaede V. and Günther O. Multidimensional access methods. ACM Comput. Surv., 30(2):170–231, 1998.
4. Gupta A., Weymouth T., and Jain R. Semantic queries with picture: the VIMSYS model. In Proc. 17th Int. Conf. on Very Large Data Bases, 1991, pp. 69–79.
5. Kosch H., Böszörmenyi L., Bachlechner A., Hanin C., Hofbauer C., Lang M., Riedler C., and Tusch R. SMOOTH - a distributed multimedia database system. In Proc. 27th Int. Conf. on Very Large Data Bases, 2001, pp. 713–714.
6. Lee J.-H., Cha G.-H., and Chung C.-W. A model for k-nearest neighbor query processing cost in multidimensional data spaces. Inf. Process. Lett., 69(2):69–76, 1999.
7. Liu Y., Zhang D., Lu G., and Ma W.Y. A survey of content-based image retrieval with high-level semantics. Pattern Recognit., 40(1):262–282, 2007.
8. Melton J. and Eisenberg A. SQL multimedia application packages (SQL/MM). ACM SIGMOD Rec., 30(4):97–102, 2001.
9. Shatford S. Analyzing the subject of a picture: a theoretical approach. Cataloging and Classification Quarterly, 6(3):39–62, 1986.
10. Smeulders A.W.M., Worring M., Santini S., Gupta A., and Jain R. Content-based image retrieval at the end of the early years. IEEE Trans. Pattern Anal. Mach. Intell., 22(12):1349–1380, 2000.
11. Tamura H. and Yokoya N. Image database systems: a survey. Pattern Recognit., 17(1):29–43, 1984.
12. Veltkamp R.C. and Tanase M. Content-Based Image Retrieval Systems: a Survey. Technical Report, Utrecht University, The Netherlands, 2000.
13. Wu P., Manjunath B.S., and Shin H.D. Dimensionality reduction for image retrieval. In Proc. Int. Conf. Image Processing, 2000, pp. 726–729.

Image Distance

► Image Retrieval

Image Indexing

► Feature Extraction for Content-Based Image Retrieval
► Visual Content Analysis

Image Management for Biological Data

Arnab Bhattacharya[1], Vebjorn Ljosa[2]
[1]Indian Institute of Technology, Kanpur, India
[2]Broad Institute of MIT and Harvard, Cambridge, MA, USA

Synonyms

Image management for life sciences; Databases for biomedical images

Definition

Image management for biological data refers to the organization of biological images and their associated metadata and annotations in a computer in such a way that they can be searched and shared.

Historical Background

The need to manage digital micrographs stored in a computer arose in the early 1990s, when digital cameras started to replace film cameras. Rapid improvement in microscopy technology coupled with steady decline in prices led to an unprecedented increase in the collection of digital biomedical images, and the need for systems to organize, search and share the images became clear. At the same time, the successes of large scale acquisition and analysis of genomic and gene expression data have prompted an increased interest in large-scale image analysis and the use of statistical methods to find patterns in large image sets. The ability of images to capture spatial and temporal relationships not immediately available from other data sources make them vital in understanding the underlying processes in biology that produce them. Database systems for these images promise important benefits, ranging from ease of organization and maintenance to the ability to share, search, explore, summarize, and analyze the data.

Foundations

Content-Based Similarity Search

Similarity search by semantic content may be performed at the level of whole images (content-based image retrieval, CBIR) or regions (region-based image retrieval, RBIR). Datta et al. provide an excellent survey [3]. Some important image retrieval systems are SIMPLIcity, WALRUS, Virage, QBIC, NeTra, Photobook,

VisualSEEk, and Keyblock. The two important components of such image comparison systems are the image features and the distance metrics. Examples of general image features are the MPEG-7 standard features, color histograms, texture, wavelets, and shape descriptors. In addition, there are domain-specific feature sets, e.g., for cultured cells or for whole organisms of the *Caenorhabditis elegans* nematode. The most commonly used distance functions are the L_2 (Euclidean) and L_1 (Manhattan) distances. The earth mover's distance (EMD) [6] has also been shown to be useful due to its ability to capture the spatial relationships among the objects and features inside the images. Relevance feedback methods [15] can improve the feature selection and the distance function by capturing the human notion of image similarity.

In biomedical images, particularly in fluorescence micrographs, large portions of the images consist of background that contains little or no information. Therefore, for RBIR purposes, it is imperative to distinguish the foreground image patterns from the irrelevant background. Segmentation and tiling are common techniques for this. Scoring-based mechanisms that model the background content have also been proposed.

Summarization

To explore an image dataset, it is useful to discover latent concepts, found in the literature under different names such as "visual vocabulary," "eigenfaces," "blobs," "visterms," and "visual keywords." Such key patterns can be thought of as analogous to keywords extracted from text, and have been useful for object detection and retrieval, as well as image classification and captioning.

Key Applications

Electronic Notebook

Computer-controlled microscopes and digital cameras have made it possible to acquire images faster than before, but the image collections are in many ways less organized than in the days of photographic films. It is common for images to be spread across personal computers, portable hard drives, and recordable compact disks. Metadata – the data about data, essentially, the description of what was imaged, how, when, and by whom – are kept in handwritten notebooks or in spreadsheets. Besides being cumbersome, this method of organization is also error prone. Images must be converted from a format specific to the vendor of the microscope to a format that is palatable to other programs. Further, contrast enhancement and other image processing is often necessary to visually inspect the images. Through these transformations, the connection between image and metadata is easily lost, and images can be lost accidentally in the process of file format conversion and image processing.

Electronic-notebook–style image databases store the original image files on a server. Derived files for visualization and image processing can be extracted, but the originals are kept pristine. Metadata are also stored on the server, persistently linked to the respective images. Queries to the electronic notebook are simple, and mostly on metadata only. Mainly, users want to find an experiment or an image based on keywords, dates, and other metadata. In particular, they already know what they are looking for, and do not expect to search by image content. They also want to browse experiments and projects by investigator or by lab, explore the images within an experiment, and download a set of images from an experiment.

It is important that the electronic notebook be flexible in regard to the metadata that can be stored. Some fields (such as the configuration of the microscope) are always useful, but others (such as the details of sample preparation and labeling) vary greatly between labs, and even between experiments. The metadata schema must be easily adaptable to the changing needs of the researchers, or the system is unlikely to be adopted.

MetaXpress (`http://www.moleculardevices.com/pages/software/metaxpress.html`), BioImage [2], Global Image Database (GID) [5], and Cell-Centered Database (CCDB) [9] are examples of electronic-notebook-style image databases. GID provides minimal set of annotations in a fixed format that caters to many image types and are useful for sharing. In CCDB, a backbone schema is provided which can be enhanced by specialized tables. There is additional provision for storing the results of image analyses as pre-defined object types. The open microscopy environment (OME) [13] and Bisque systems (`http://www.bioimage.ucsb.edu/`) also serve as electronic notebooks, although they provide more advanced functionality, described in later sections. Further, Bisque allows the user to modify the metadata schema.

Sharing and Exploration

Sharing images beyond the investigator's immediate colleagues leads to two new requirements for the

image database: integration of metadata and image-content–based search. The flexibility of modifying the schema that is so valuable in an electronic notebook impedes sharing because the system will need to know which fields in one schema corresponds (perfectly or partially) to which fields in another. Schema matching techniques have been developed that unify the schemas based on both structure and contents [1]. A related problem pertains to the contents of the fields. For instance, the same antibody can be known under different names. As a second example, one researcher may describe a cell as a "photoreceptor" whereas another scientist may be more specific and call it a "cone." Whereas the Gene Ontology (GO, http://www.geneontology.org/) is a start, it will need to be combined with other ontologies or controlled vocabularies.

For *in vivo* images, the location in the tissue or organism that was imaged is another important piece of metadata. To understand and explore such information, registration and mosaicking techniques have been developed to align the image to an *atlas* [4].

So far, in addition to the images themselves, only metadata, i.e., information about the images, have been considered. A second kind of image-associated data are *annotations*. Annotations are data *derived from* the images. They can, for instance, indicate specific objects of interest (e.g., crescent-shaped nuclei, neurite branching points) or summarize measurements (e.g., cell counts, neurite length distributions). Image features can also be considered annotations. Annotations can be stored in an electronic notebook, just like metadata. Annotations are, however, more important for sharing and exploration, for they allow the user to find interesting images even if they were acquired for a completely different purpose. For instance, a researcher may want to find all perturbations (drug treatments, gene knockdowns, etc.) that led to a high rate of crescent-shaped nuclei.

Although metadata and annotations have many similarities, the latter are much more challenging for the database designer. Each metadata field has one true value which is provided by the original investigator, typically at the time the image is uploaded to the database (and perhaps updated or corrected later). In contrast, because annotations are interpreted from the image, more than one user or tool may add different values for the same annotation. Two researchers may disagree on what kind of cell a certain object is, and two cell counting tools will produce different results. The database must therefore be able to store uncertain information and track the provenance of the annotations. Handling uncertain information in databases is an active research area [7,11].

Another open question is how to control access to annotations. When sharing sets of images between labs, issues such as who owns the data, who are allowed to access them, who can run analysis tools on them, who can update the database with new results, etc., are critical. Another important factor is ensuring the quality of data, including the validation, accuracy, completeness, and integrity of the images being processed or uploaded, as well as the annotations produced by analysis tools. Similar issues are discussed by Toga et al. [14].

Although search by metadata and annotations is important and useful, it has an obvious limitation: one can only find what others have annotated. For effective exploration, one must be able to search the images by content (see Content-Based Similarity Search and Summarization in the Scientific Fundamentals section).

Analysis

The third principal application of image databases to biomedical images is as a platform for numerical and statistical analysis of the annotations derived from the images.

Combinations of analysis steps can be viewed as networks, where the results of one analysis feeds into another. For example, ridges can be extracted directly from the image, and can, therefore, be considered as low-level annotations. Next, the ridges are filtered and linked into neurites, which are then combined with cell bodies (detected by some other method) into whole neurons. Note that neurites and neurons are objects that have biological meaning. Finally, one may measure the amount of a certain protein within the cell. The Open Microscopy Environment (OME) [13] provides a module system for tracking the provenance of the resulting data. Each module execution is recorded, relating the module and its parameters to the input and output data.

Analysis of annotations frequently involves classification. The reason is that clustering and other unsupervised methods often pick up cell-cycle–dependent variations and other differences that are real and profound, but irrelevant to the question of interests. This can be avoided by classifying perturbed samples from controls [8].

Future Directions

Whereas the size of the images is a problem for image databases, managing the analyzed data is an even bigger problem because the analyses can be numerous and the annotations produced by such analyses can be queried in complex ways. As an example, a single experiment can comprise of 350GB of images, and over 600 features are extracted from each cell for a total of over 20GB of annotations [10]. Future experiments are expected to reach 20 times this size. Therefore, efficient index structures and database access methods will be needed.

Queries posed to databases as part of an analysis process generally involve a large number of records. For instance, a classification task may need to look at the records of all the cells perturbed in a certain way. However, due to feature selection, only part of the record may need to be examined – say, 50 of the 600 features. Column-oriented DBMSs [12] have the potential to improve the performance dramatically for such queries. Another very general approach to speeding up queries is to approximate the result based on only part of the data. Techniques that give statistical guarantees about the quality of approximation have been proposed for many query types, but are so far only available in specialized applications and in research prototypes of database systems.

Cross-references

- ▶ Biological Metadata Management
- ▶ Biostatistics and Data Analysis
- ▶ Image Database
- ▶ Machine Learning in Computational Biology
- ▶ Ontologies and Life Science Data Management

Recommended Reading

1. An Y., Borgida A., Miller R.J., and Mylopoulos J. A semantic approach to discovering schema mapping expressions. In Proc. 23rd Int. Conf. on Data Engineering, 2007, pp. 206–215.
2. Carazo J.M., Stelzer E.H., Engel A., Fita I., Henn C., Machtynger J., McNeil P., Shotton D.M., Chagoyen M., de Alarcón P.A., Fritsch R., Heymann J.B., Kalko S., Pittet J.J., Rodriguez-Tomé P., and Boudier T. Organising multi-dimensional biological image information: the BioImage database. Nucleic Acids Research, 27(1):280–283, 1999.
3. Datta R., Li J., and Wang J.Z. Content-based image retrieval: approaches and trends of the New Age. In Proc. 7th ACM SIGMM Int. Workshop on Multimedia Information Retrieval, 2005, pp. 253–262.
4. Fowlkes C.C., Hendriks C.L.L., Keränen S.V.E., Biggin M.D., Knowles D.W., Sudar D., and Malik J. Registering *Drosophila* embryos at cellular resolution to build a quantitative 3D atlas of gene expression patterns and morphology. In Proc. Int. Workshop on Bioimage Data Mining and Informatics, IEEE Computational Systems Bioinformatics Conference, 2005.
5. Gonzalez-Couto E., Hayes B., and Danckaert A. The life sciences global image database (GID). Nucleic Acids Research, 29(1):336–339, 2001.
6. Ljosa V., Bhattacharya A., and Singh A.K. Indexing spatially sensitive distance measures using multi-resolution lower bounds. In Advances in Database Technology, Proc. 10th Int. Conf. on Extending Database Technology, 2006, pp. 865–883.
7. Ljosa V. and Singh A.K. APLA: indexing arbitrary probability distributions. In Proc. 23rd Int. Conf. on Data Engineering, 2007, pp. 946–955.
8. Loo L.-H., Wu L.F., and Altschuler S.J. Image-based multivariate profiling of drug responses from single cells. Nature Methods, 4(5):445–453, 2007.
9. Martone M.E., Zhang S., Gupta A., Qian X., He H., Price D.L., Wong M., Santini S., and Ellisman M.H. The cell-centered database: a database for multiscale structural and protein localization data from light and electron microscopy. Neuroinformatics, 1(4):379–395, 2003.
10. Moffat J., Grueneberg D.A., Yang X., Kim S.Y., Kloepfer A.M., Hinkle G., Piqani B., Eisenhaure T.M., Luo B., Grenier J.K., Carpenter A.E., Foo S.Y., Stewart S.A., Stockwell B.R., Hacohen N., Hahn W.C., Lander E.S., Sabatini D.M., and Root D.E. A lentiviral RNAi library for human and mouse genes applied to an arrayed viral high-content screen. Cell, 124:1283–1298, 2006.
11. Sarma A.D., Benjelloun O., Halevy A., and Widom J. Working models for uncertain data. In Proc. 22nd Int. Conf. on Data Engineering, 2006.
12. Stonebraker M., Abadi D.J., Batkin A., Chen X., Cherniack M., Ferreira M., Lau E., Lin A., Madden S., O'Neil E., O'Neil P., Rasin A., Tran N., and Zdonik S. C-store: A Column-Oriented DBMS. In Proc. 31st Int. Conf. on Very Large Data Bases, 2005, pp. 553–564.
13. Swedlow J.R., Goldberg I., Brauner E., and Sorger P.K. Informatics and quantitative analysis in biological imaging, Science, 300(5616):100–102, 2003.
14. Toga A.W. Neuroimage Databases: The Good, the Bad and the Ugly. Nature Reviews Neuroscience, 3(4):302–309, 2002.
15. Zhou X.S. and Huang T.S. Relevance Feedback in Image Retrieval: A Comprehensive Review. Multimedia Systems, 8(6):536–544, 2003.

Image Management for Life Sciences

▶ Image Management for Biological Data

Image Metadata

Frank Nack
University of Amsterdam, Amsterdam,
The Netherlands

Synonyms

Pictorial metadata; Picture metadata; Image representation

Definition

A digital image is a representation of a two- or three-dimensional image, where the representation can be of vector or raster type.

Metadata is data about data of any sort in any media, describing an individual datum, content item, or a collection of data including multiple content items. In that way, metadata facilitates the understanding, characteristics, use and management of data.

Image metadata is structured, encoded data that describes content and representation characteristics of information-baring image entities to facilitate the automatic or semiautomatic identification, discovery, assessment, and management of the described entities, as well as their generation, manipulation, and distribution.

Historical Background

Many of the techniques of digital image processing were developed in the 1960s at, among others, the MIT, Bell Labs, and the University of Maryland. These works tried to automatically generate content representations that could support research on satellite imagery, wire-photo standards conversion, medical imaging, character recognition, and photo enhancement. In the 1970s, digital image processing improved mainly because cheaper computers and dedicated hardware that allowed real-time image processing [2] became available. The next 20 years not only saw a further improvement on automatic feature extraction for domains, such as scientific, industrial, medical, and environmental research, but also a steady infiltration of those technologies in everyday media environments, such as image editing tools (Photoshop (http://en.wikipedia.org/wiki/Adobe_Photoshop), Illustrator (http://en.wikipedia.org/wiki/Adobe_Illustrator_CS2), GIMP (http://en.wikipedia.org/wiki/GTK%2B), or Maya (http://en.wikipedia.org/wiki/Maya_%28software%29)), new media authoring tools (Director/Shockwave (http://www.adobe.com/products/director), or Flash (http://www.adobe.com/products/flash)), and, more importantly, in the development of digital cameras. Images turned into a common information item that people could handle as easily as text.

The real push for the digital image came with the emergence of the world wide web (web) and the improvements of sensor technology, especially in digital photography. These swift developments in image distribution and generation in combination with easy to use web presentation technology (e.g., Dreamweaver (http://en.wikipedia.org/wiki/Dreamweaver), Frontpage (http://en.wikipedia.org/wiki/Microsoft_FrontPage), HTML (http://www.w3.org/MarkUp), and SMIL (http://www.w3.org/AudioVideo)) deeply changed the social way of exchanging information, increasing the available amount of images dramatically.

The answer to the resulting media-based information flood was research that focused again on automatic ways to index the available images in a timely and meaningful way. The goal was now to make use of image processing based on features required for interactive image understanding. Machine-generated metadata, however, turned out to be problematic as it is exclusively organized around the sensory surface structures of media, i.e., the physical features of an image, resulting in the sensory and semantic gap [14].

By the beginning of the twenty-first century this need for semantic-aware metadata schemata forced research to explore new ways of content representation. A large number of initiatives developed metadata standards to allow machines as well as humans to access the semantics of media items, such as Dublin Core (http://www.dublincore.org/), the Art and Architecture Thesaurus (AAT) by the J. Paul Getty Trust (http://www.getty.edu/research/conducting_research/vocabularies/aat/), the semantic web activity of the W3C (http://www.w3c.org/2001/sw/), and ISO's MPEG-4, MPEG-7 and MPEG-21 (http://www.chiariglione.org/mpeg/).

The interesting aspect of particularly the ISO and related standards had been that they tried to merge the high-level conceptual aspects of their content description with low-level structures of feature representation as used in signal processing, to allow the processing of audio-visual information over several semantic levels.

The common foundation for this fusion applies XML-based description languages.

A drawback of these approaches, namely regarding the process of attributing metadata to a media item as a terminated process, resulted in research that introduced new mechanisms to overcome the static structure of annotations, where flexibility is achieved by agreeing upon the collection of semantic-based and machine-processable metadata during established media workflow practices [5,6,9].

Around 2003, the web saw the growth of massive image databases, such as Flickr (http://www.flickr.com), that are mainly build on user generated content. In such environments people do not agree beforehand on an annotation taxonomy or ontology. As a result, research developed folksonomy tagging (also known as collaborative tagging, social classification, social indexing, and social tagging), a method of collaboratively creating and managing tags to annotate and categorize content. In folksonomy tagging metadata is not only generated by experts but mainly by creators and consumers of the content, where a tag is a keyword or term associated with or assigned to a piece of information (a picture, a map, etc.), which enables keyword-based classification and search. The advantage of tagging is its ease of use - creating a vocabulary based on freely chosen keywords instead of a controlled set of terms and structures. This approach, though highly popular, carries serious problems. Typically there is no information about the semantics of a tag, no matter if it is a single tag or a bag of tags. Additionally, different people may use drastically different terms to describe the same concept. This lack of semantic distinction can lead to inappropriate connections.

Foundations

When a person looks at an image, they first perceive the image on an optical level where they try to identify as many objects as possible in the available time of perception. Each object is mentally transformed into an iconic sign, which processes some properties of the object represented. The signification of iconic signs is based on socially determined small semantic systems (codes) and rules for their combination. Some of the code systems are: recognition, tonal, iconic, iconographic, rhetoric, and stylistic [7].

The iconic code is by far the most valuable code, because it defines the articulation potential of visual material. The creation of meaning in visual material is based on a *triple articulation* of figure, sign and *semes* and receives its expression by convention. The three elements are described as such:

1. A *figure* forms conditions for perception, such as relationships between object and background, or contrast in light.
2. A *sign* denotes, using conventionalized graphical methods, units of understanding (nose, sky), abstract models, or idealized diagrams of the object (the sun as a circle with thread-shaped beams).
3. *Semes* are complex iconic phrases, such as "this is a man standing in profile." They are the most simply catalogued, since the iconic code works most often on their level only.

The fact of a conventionalized triple articulation is important since it describes the essential difference from natural language, which has two articulations (phonemes and morphemes). Thus, comparing an object in an image with the corresponding word, the visual object always exceeds the concept of the word, as the image will portray specific qualities about the object for which the word is simply inadequate (see the tag problem mentioned above).

The organization of signs in an image is provided by two types of structures:

1. Syntagmatic, which represents a sequence of signs in which the relation of parts determines their meaning.
2. Paradigmatic, which represents potential substitutions in which a range of potential candidates can take the place of a sign in a syntagmatic structure.

The main application of image metadata is the retrieval of an image, where content features play an essential role. Search based low-level features functions on an iconic level, as most of the low-level feature based description methods. The key methods of how users search for an image are:

- Search by specification: aims for searching the identical replicate of the image the user has in mind or the identical object the user needs. Essential is the provision of optional variants (prosodic features) that form the conditions for perception and thus support identification [4].
- Search by categorization: aims for retrieving an arbitrary image representative of a specific class.

Categories may be derived from labels or emerge from the database [19]. This type of search usually requires a domain specific definition of similarity. Here, ontologies might become applicable.
- Search by association: aimed at browsing through a large set of images from unspecified sources. Association-based search usually implies iterative refinement of the query, which mainly asks for establishing similarities between images. Important in such search applications is the provision of relevance feedback [8] or the provision of additional sources [16].

Supporting these types of search methods are needed and allow the description of content so that a machine or a human user can access the images. Some of them can be applied in an automatic fashion, those are image feature based, and some provide high-semantic descriptions, which are often based on manual annotation. The latter are required, because, as described above, there is additional context that specifies the need of a query, e.g., the esthetics' the picture has to provide.

Content Description Based on Invariant Features

The purpose of image processing is to enhance aspects in the image data relevant to the user query. This can be achieved through the description of invariant features, such as, color, shape and texture. Invariant features tend to be object-specific information as they are insensitive to the accidental conditions of the sensing. It is important to note that it is the user who has to specify the minimal set of invariant features as it is part of his or hers intention [3].

- *Color:* The two general problems addressed in work on color are variances in color and differences in human perception. Color representation techniques, among others, are:
  - RGB color representation describes the image in its literal color properties, namely the Red, Green and Blue properties of each pixel in an image, where two-dimensional images are recorded in frontal view under standard conditions.
  - Opponent color representations, which uses the opponent color axes that isolate the brightness information on the third axis.
  - Lab-space approach, which exploits the Euclidean distance between two color representations to model human perception of color differences.
  - The HSV-representation exploits the hue, which is invariant under the orientation of the object with respect to the illumination and camera direction.
  - Hidden Markov models [11]
  - Clusters in a color histogram, where the RGB space is searched to identify which pixels in the image originate from one uniformly colored object.
- *Shape:* Shape is considered as the collected properties that capture differential geometrical details in the image [15]. Methods to represent shape are scale-based theory, conspicuous shape geometric invariants, or differential geometric invariants.
- *Texture:* Texture covers every aspect of an image that is not described through color and shape. The essential texture analysis techniques are: the Markovian analysis, multiscale autoregressive MRSAR-models, or wavelets.

Content Description Based on Semantic Features

The invariant features serve as a preprocessing step within the usual approach in content-based image retrieval, which divides the image first into parts and then computes the features for those parts. The essential segmentation methods are strong and weak segmentation.

- Strong segmentation divides the data into regions in such a way that a region contains the pixels of the silhouette of an object in the real world and nothing else. The problem is that this approach usually only succeeds sophisticated techniques in very narrow domains, such as trademark validation.
- Weak segmentation aims to support broad domains of general images by grouping image data into inconspicuous regions that are internally homogeneous according to some criterion. Weak segmentation is used in many retrieval systems, either as a purpose of its own or as a preprocessing stage for data-driven model-based object segmentation.

Both approaches end up in a preferred set of features [14], which can be classified as follows:

- Accumulating features aggregate the spatial information of a partitioning irrespective of the image data. The histogram [17] is the common method applied, such as the color histogram. Alternatives are the correlogram, or the autocorrelogram. An

example of accumulative features might be those calculated from the top part of a picture, which effectively identify an image as indoor or outdoor space. The danger of accumulative features is the inability to discriminate among different entities and semantic meanings in the image.

- Salient features are a typical example of weak segmentation. The idea here is that grouping of the data is performed resulting in homogeneous regions. From the merged regions, a selection is made on their salience, where saliency is understood as those special points that survive longest while gradually blurring the image in scale space.
- Shape and object features, which focus on segmenting the object in the image. The problem is the automatic segmentation in broad domains. Yet, often it is not necessary to know exactly where an object is located but rather that its presence can be identified. Techniques for that are elastic matching and multi-resolution representation of shapes, multi-scale models of contours, the description of the object boundaries, or the description of global shape invariants.
- Combined entities of features, where calculations for different entities in the image and relationships between them are available. Such a structural feature set may contain feature values and spatial relationships, a hierarchically ordered set of feature values, relationships between point sets or object sets, or a graph of relations between blobs.

Interpretation of Features

In general there are two ways to make use of computed feature sets for the interpretation of an image.

- Deriving an unilateral interpretation from the feature set. This approach encodes an approximate subset of possible interpretations of an image relevant for a particular application. In that way the subset describes the semantics associated to features for the application [13].
- Comparing the feature set with the elements in a given data set based on similarity.

Similarity is an interpretation of the image based on the difference with another image. For each of the earlier described feature types, a different similarity measure is needed:

- *Similarity of objects* can be established through shape comparison based on transforms, moments, deformation matching, scale space matching, and dissimilarity measurement.
- *Similarity of structure* can be established based on a Bayesian framework, topological arrangements of relevant domain parts, spatial relationships between objects, or tree or graph representations.
- Similarity of *salient features* can be achieved by means of the distance between the feature vector composed of, e.g., the color, texture, position, and direction of the two ellipses, derived from the feature values measured of the blobs resulting from weak segmentation of the two images.

Under normal circumstances similarity comparison is not performed on a 1:1 image basis but rather on a 1:n image basis. For the latter cases, as in large image databases, gained experiences during the comparison of images can be exploited for better results. In that way similarity detection is understood as a classification problem to be solved based on statistical pattern recognition. Additional learning methods are the use of transduction (e.g., for partially labeled data), probabilistic constellations of features (e.g., for unlabeled data sets), or latent semantic indexing.

Image Interpretation Based on High-Level Semantic Metadata

As pointed out earlier, the representation of an image based on low-level features might be useful for narrow domains, such as the validation of trademarks. Every information that is based on higher-level codes than the iconic, e.g., the name of the image creator or the image's contribution to the technique of painting or photography, can not be extracted in that manner.

For those codes the annotation is based on concepts. The most basic form of such an annotation is a tag. The most fundamental set of tags for the annotation of images is certainly the Dublin Core set. The most complex description framework so far are those developed by MPEG-7 and the W3C's Multimedia Incubator Group (see also the Media Annotation Working Group: http://www.w3.org/2008/01/media-annotations-wg.html). The metadata falling in this category often require manual annotation [1], where the available structures are organized either in hierarchical or graph structures (the syntagmas) and use thesaurus or ontolological formats (class, subclass and their properties as well as relations between

them) to allow for paradigmatic methods of choice to support advanced search (query extension) [10], or interpretation and generation.

Dublin Core [18]

The Dublin Core Metadata Element Set is a vocabulary of fifteen properties for use in resource description. The fifteen properties are: contributor, creator, date, description, format, language, publisher, relation, rights, source, subject, title, type.

Since January 2008, Dublin Core includes formal domains and ranges in the definitions of its properties. For supporting conformance of existing implementations of "simple Dublin Core" in RDF, domains and ranges have not been specified for the fifteen properties of the dc: namespace (http://purl.org/dc/elements/1.1/). Instead, the Dublin Core Metadata Initiative has opted for creating fifteen new properties with "names" identical to those of the Dublin Core Metadata Element Set Version 1.1 in the dcterms: namespace (http://purl.org/dc/terms/). These fifteen new properties have been defined as subproperties of the corresponding properties of DCMES Version 1.1 and assigned domains and ranges as specified in the more comprehensive document "DCMI Metadata Terms."

MPEG-7 [12]

MPEG-7 standardizes descriptions of AV data content in multimedia environments. It provides descriptions of multimedia content on varying complexity levels to let users search, browse, filter, or interpret content using search engines, filter agents, or any other program. MPEG-7 offers a set of AV description tools in the form of descriptors (Ds) and description schemata (DS) a valid MPEG-7 description should adhere to. Descriptors usually bind a feature to a set of values. Description schemata specify the structure and semantics of the relationships between the components of descriptors and between other description schemata. These structures let users create application-specific content descriptions (see Fig. 1 as an example).

The standard has eight parts, each responsible for one aspect of the functionality, of which 3 are relevant for visual data:

- Part 2 - the DDL specifies the language for defining the standard set of description tools (description schemata, descriptors, and data types), new tools, and the main parser requirements. The DDL is based on XML-Schema, developed by the W3C.

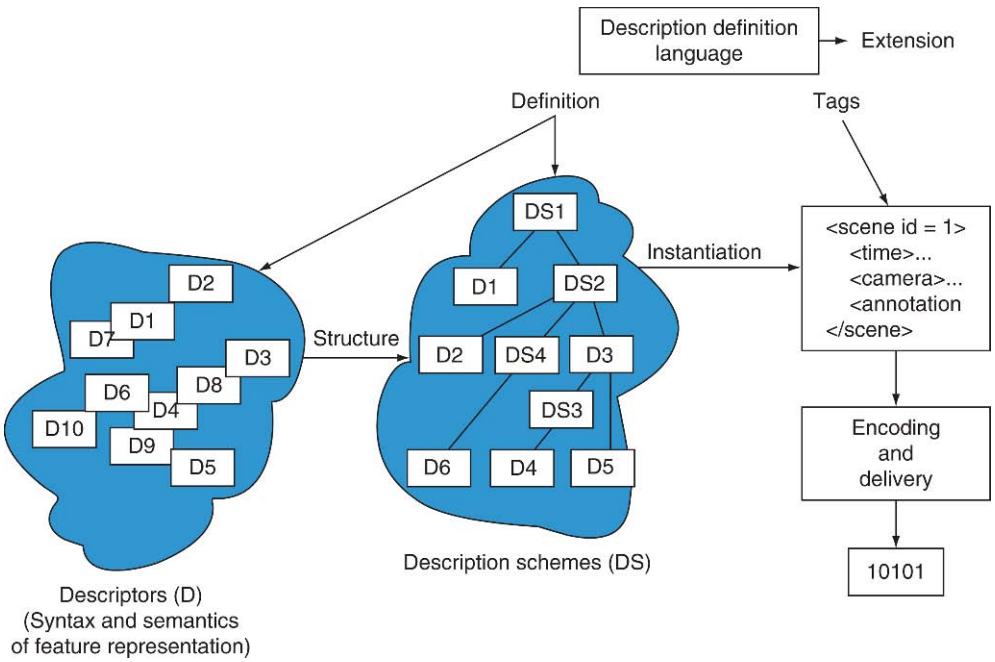


Image Metadata. Figure 1. The main MPEG-7 elements. Content authors can use these structures to create application-specific content descriptions.

- Part 3 - Visual consists of schemata and descriptors covering basic visual features such as color, texture, shape, and face recognition. It provides the descriptor syntax and description schemata in normative DDL specifications and the corresponding binary representations.
- Part 5 - Multimedia Description Schemes (MDS) specifies generic description tools pertaining to multimedia, including audio and visual content. MDS covers the basic elements for building a description, the tools for describing content and relating the description to the data, and the tools for describing content on organization, navigation, and interaction levels. The MDS alone forms more than half of the complete standard and has its own internal structure, shown in Fig. 2.

As the standard is rather large, MPEG has begun to establish media profiles (MPEG-A to MPEG-E) that integrate multiple MPEG technologies. The relevant visual profile is MPEG-C.

The W3c's Incubator Group [20]

The work of this group has explored the advantages of using Semantic Web languages and technologies for the creation, storage, manipulation, interchange and processing of image metadata. In addition, it provided guidelines for Semantic Web-based image annotation, illustrated by use cases. In its publications the relevant RDF and OWL vocabularies are discussed, along with a short overview of publicly available tools.

Key Applications

Image metadata is useful for the creation, manipulation, retrieval and distribution of 2D or 3D digital image sources within domains, such as

- The creative industries (e.g., fine arts, entertainment, journalism, etc.),
- Education (e.g., in computer based training courses, military or industrial training),
- Environmental research (e.g., Interpretation of satellite images),
- Medicine (e.g., Identification of cancer cells in a lung scan etc.).

Future Directions

Research has still to address how the merge between high-level semantic descriptions and low-level feature-based descriptions can be achieved. In this context, solutions need to be found that merge the results of folksonomy tagging and defined description vocabularies into a suitable description

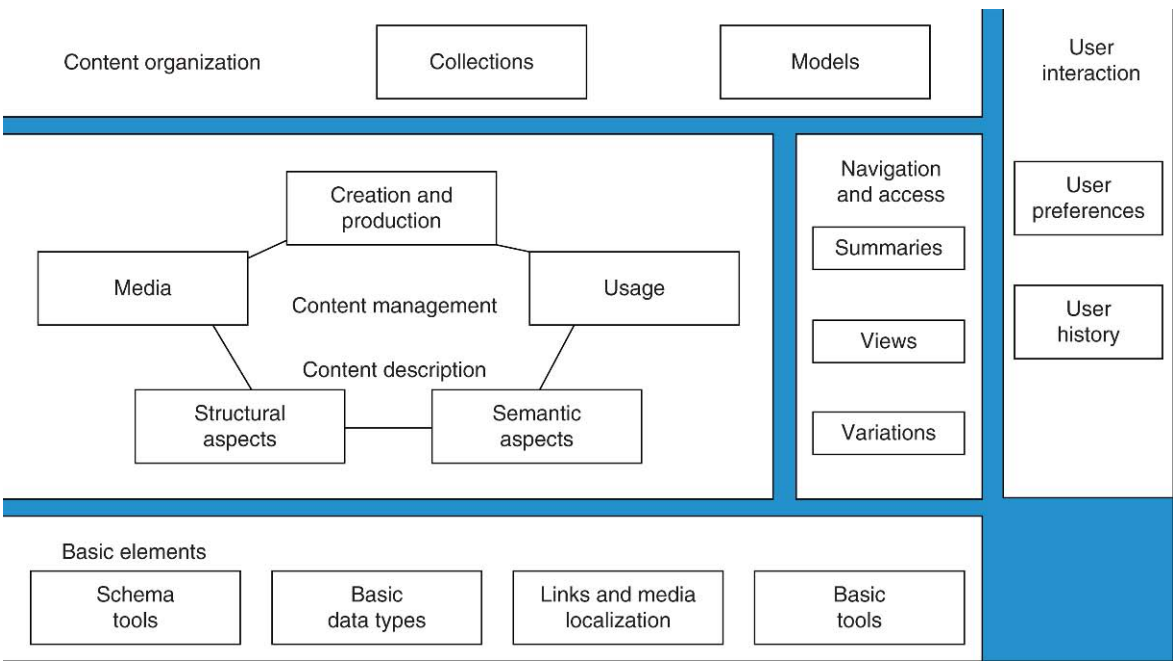


Image Metadata. Figure 2. Overall organization of MPEG-7 multimedia description schemes.

framework. Finally, it is important that easy to use annotation tools will be developed.

Cross-references

- ▶ Image Content Modeling
- ▶ Image Representation
- ▶ Multimedia Metadata
- ▶ Video Metadata

Recommended Reading

1. Ahern S., Davis M., Eckles D., King S., Naaman M., Nair R., Spasojevic M., and Hui-I Yang J. ZoneTag: designing context-aware mobile media capture to increase participation. In Proc. Pervasive Image Capture and Sharing: New Social Practices and Implications for Technology Workshop, 2006.
2. Blasser A. (ed.) Database Techniques for Pictorial Applications. Lecture Notes in Computer Science, Springer, London, UK, 1979.
3. Burkhardt H. and Siggelkow S. Invariant features for discriminating between equivalence classes. Nonlinear Model-Based Image Video Processing and Analysis. Wiley, NY, 2001, pp. 269–307.
4. Cox I.J., Miller M.L., Minka T.P., and Papathomas T.V. The Bayesian image retrieval system, picHunter: theory, implementation, and pychophysical experiments. IEEE Trans. Image Process., 9(1):20–37, 2000.
5. Davis M. Active capture: integrating human-computer interaction and computer vision/audition to automate media capture. In Proc. IEEE Int. Conf. on Multimedia and Expo, 2003, pp. 185–188.
6. Dorai C. and Venkatesh S. Bridging the semantic gap in content management systems – computational media aesthetics. In Media Computing Computational Media Aesthetics, C. Dorai, S. Venkatesh (eds.). Kluwer, Boston, MA, 2002.
7. Eco U. Articulations of the cinematic code. In Movies and Methods, B. Nichols (ed.). University of California Press, Berkeley, 1976, pp. 590–607.
8. Frederix G., Caenen G., and Pauwels E.J. PARISS: Panoramic, Adaptive and Reconfigurable Interface for Similarity Search. In Proc. Int. Conf. Image Processing, 2000 vol. 3, pp. 222–225.
9. Hardman L., Obrenovic Z., Nack F., Kerherve B., and Piersol K. Canonical processes of semantically annotated media production. Multimedia Systems, 14(6):327–340, 2008.
10. Hollink L. Semantic Annotation for Retrieval of Visual Resaources. Ph.D thesis, Vrije Universiteit, Amsterdam.
11. Lin H.C., Wang L.L., and Yang S.N. Color image retrieval based on hidden Markov models. IEEE Trans. Image Process., 6(2):332–339, 1997.
13. Nack F., Windhouwer M., Hardman L., Pauwels E., and Huijberts M. The role of highlevel and lowlevel features in style-based retrieval and generation of multimedia presentations. New Rev. Hypermedia Multimedia, 7(1):7–37, 2001.
14. Smeulders A.W.M., Worring M., Santini S., Gupta A., and Jain R. Content-based image retrieval: the end of the early years. IEEE Trans. Pattern Anal. Mach. Intell., 22 (12):1349–1380, December 2000.
15. Smith S.M. and Brady J.M. SUSANÐA new approach to low level image processing. Int. J. Comput. Vis., 23(1):45–78, 1997.
16. Swain M.J. Searching for multimedia on the World Wide Web, icms. In Proc. Int. Conf. on Multimedia Computing and Systems, 1999, pp. 32–37.
17. Swain M.J. and Ballard B.H. Color indexing. Int. J. Comput. Vis., 7(1):11–32, 1991.
18. The Dublin core metadata initiative. Available at: http://www.dublincore.org/. Access date: October 12th 2008.
19. Weber M., Welling M., and Perona P. Towards automatic discovery of object categories. In Proc. IEEE Int. Conf. on Computer Vision and Pattern Recognition, 2000, pp. 2101–2108.
20. W3C multimedia incubator group. Available at: http//www.w3.org/2005/incubator/mmsem/. Access date: October 12th 2008.

Image Query Processing

- ▶ Image Querying

Image Querying

Ilaria Bartolini
University of Bologna, Bologna, Italy

Synonyms

Image query processing

Definition

Image querying refers to the problem of finding objects that are *relevant* to a user query within image databases (Image DBs). The classical solutions to deal with such problem include the *semantic*-based approach, where an image is represented through metadata (e.g., keywords), and the *content*-based solution, commonly called *content-based image retrieval* (CBIR), where the image content is represented by means of low-level features (e.g., color and texture). While with the semantic-based approach the image querying problem is transformed into an information retrieval problem, for CBIR more sophisticated query evaluation techniques are required. The usual approach to deal with this is illustrated in Fig. 1: By means of a graphical user interface (GUI), the user provides a query image, by

sketching it using graphical tools, by uploading an image she/he has, or by selecting an image supplied by the system. Low-level features are extracted for such image (possibly dividing it into regions, see below); such features are then used by the query processor to retrieve the DB images having similar characteristics.

How the set of relevant DB images is determined depends on which low-level features are used to characterize image content, on the criterion used to *compare* image features, on how DB objects are *ranked* with respect to the query (based on either a quantitative measure of similarity or qualitative preferences), and, finally, on whether the user is interested in the whole query image or only in a part of it. All these aspects strongly influence the *query evaluation* process.

Historical Background

In spite of the many efforts spent so far, the problem of retrieving relevant objects within *image databases* (Image DBs) is still a complex task. Following the semantic-based approach, images are described by means of metadata such as keywords, captions, or descriptions and the retrieval is performed over such words using *annotation-based image retrieval* techniques. In this direction, several solutions have been recently proposed, such as image extensions to public search engines like Google (http://images.google.com/) and Yahoo (http://images.search.yahoo.com/). Such systems consider the contextual information of a crawled image (like the image filename, its title, the surrounding text, etc.) to infer the relevance of the image to the query. In a similar way, some systems, like flickr (http://flickr.com/), assess the relevance of an image to the query by taking into account the characterization of the image provided by the user. However, such a manual image annotation process is expensive and time-consuming. In order to overcome such limitations, there has been a large amount of research done on (semi-)automatic image annotation with the aim to assign meaningful keywords to images by exploiting the information of a pre-annotated set of objects.

With respect to the *content-based image retrieval* (CBIR) solution, the aim is to avoid the use of textual descriptions. This is usually done by using visual similarity to retrieve images, for example, asking for images that are similar to a user-supplied query image, i.e., following the *query by example* (QBE) paradigm fist adopted in the IBM's query by image content (QBIC) system [5]. In particular, each image is characterized using global *low-level features*, such as color and texture, and the result of a query consists in the set of DB

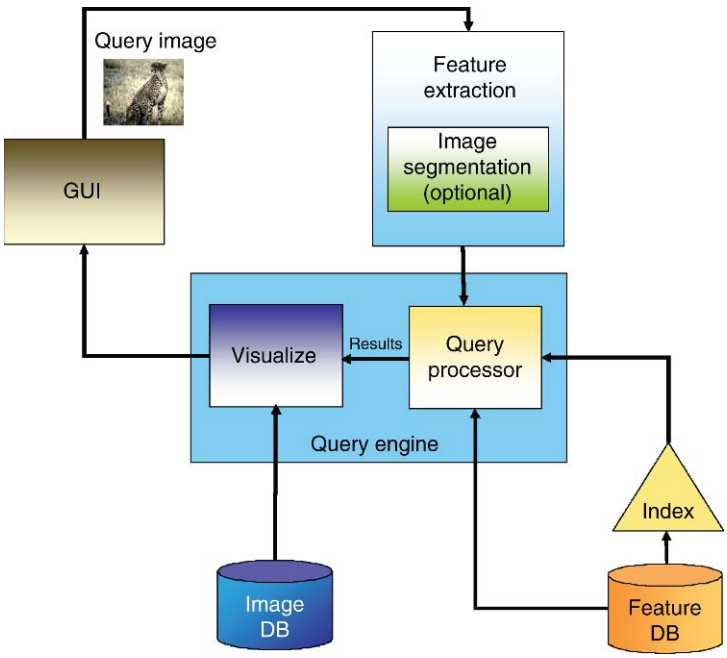


Image Querying. Figure 1. The image querying scenario for CBIR.

objects that better match the visual characteristics of the target image, according to a predefined *similarity criterion*, which is in turn based on low-level features [8]. Although CBIR represents a completely automatic solution for the image querying problem, the accuracy of its results is not always completely satisfactory for the user, especially for high-level concept queries, for which low-level features are hardly exploitable due to their low discriminative power. This is largely due to the so-called semantic gap existing between the concept of similarity as percept by the human brain and the one implemented by the system. The effectiveness of this approach still calls for improvement: The use of *relevance feedback techniques* could be of help, but it is still not enough to reach acceptable levels of accuracy.

More recently, the *region-based image retrieval* (RBIR) approach has been proposed, which has lead to promising results. With respect to the case in which images are represented by means of global descriptors, RBIR is able to characterize the image content in a more precise way by *segmenting* each image into a set of homogeneous regions from which a set of low-level features are extracted. As a consequence, most of the modern image database systems adopts the RBIR paradigm in order to improve the retrieval accuracy [9,1,3,4,6,10]. Almost all such systems treats each region as a separate query and somehow aggregate the so-obtained partial results in order to derive the final answer. This property introduces a number of new interesting query processing problems with respect to the case in which the segmentation is not considered. Among these, which constraints must be satisfied by the aggregation rule in order to provide the query result and which criterion has to be followed to order the DB images with respect to the query. Finally, with RBIR, new query types, such as partial queries, are supported.

Foundations

The general approach followed by RBIR systems is to divide an image I into a set of homogeneous regions, i.e., set of pixels that share similar visual characteristics, and to represent each of them by means of a set of low-level features, such as color and texture. Thus, any image I is seen as a complex object. Regions comparison is obtained by defining a *region similarity function*, s_R, able to produce a scoring value which quantifies their visual similarity. Given a query image I^q, the set of relevant DB images to I^q is computed starting from the similarities between the query regions and the regions of DB images. This requires first to somehow *match* regions of the query to regions of DB images, by using the proper aggregation of region similarities, and then to *rank* DB images so as to produce the query result (see Fig. 2).

Formally, the image querying problem can be concisely formulated as follows:

Problem

Given a query image I^q composed of regions, an image database $\mathcal{IDB}$, where each image $I \in \mathcal{IDB}$ is composed of regions, and a region similarity function, $s_R(R_i, R_j)$, that for each pair of regions, (R_i, R_j), returns their

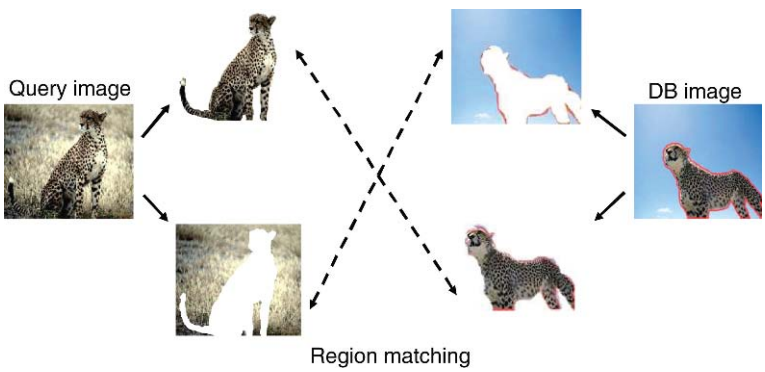


Image Querying. Figure 2. The similarity between the query image and a DB image is assessed by taking into account the similarity between matched regions.

similarity score, determine the set of relevant *images in* $\mathcal{IDB}$ *wrt* I^q.

Instantiating the general problem can be done in different ways, since different coordinates are involved in the definition of what "relevant" actually means. Among these coordinates, the rules according to which a region of the query can be coupled to regions of a DB image (conventionally called *matching type*) and the aggregation modality applied to the region similarity scores in order to assess the query result (i.e., the *ranking model*).

Matching Type

The matching type defines which set of constraints applies when the component regions of the query image $I^q = \{R_1^q,...,R_n^q\}$ have to be matched to the component regions of a DB image $I = \{R_1,...,R_m\}$.

Two relevant cases for matching types are the *one–to–one* ($1 - 1$) and the *many–to–many* ($n - m$) matching types. In the $1 - 1$ case, each region of image I^q is associated to at most one region of I, and vice versa. In particular, each matching has to be *complete*, i.e., if $n > m$ (respectively, $n < m$) then only $n - m$ (resp., $m - n$) regions of I^q (resp., I) have to remain unmatched (refer to Fig. 3 for an example).

With the $n - m$ matching type, each region of I^q can be associated to many regions of I, and vice versa. This, however, could lead to undesired (pathological) results. For example, a single region of the query could be matched to *all* regions of a DB image. This has been termed "the two tigers problem" in [11] since it arises when a single region (a tiger) of the query image is very similar to multiple regions of a DB image (e.g., containing two tigers).

A special case of $n - m$ matching that avoids this problem is the Earth Mover Distance (EMD) matching [7], where variable-sized pieces of regions are allowed to be matched (the size of each region defines the maximum amount for its matching). This contrasts with the $1 - 1$ matching, where elements of fixed size (i.e., regions) are matched individually.

Ranking Model

Two generic models of ranking are possible: *k-Nearest Neighbors* (*k*-NN) and *Best Matches Only* (BMO) [2]. The *k*-NN ranking model (also known as *Top-k selection*) requires to define the image similarity of a DB image I with respect to a query image I^q, $s_I(I^q,I)$, by means of a *numerical scoring function* (*sf*), such as the average, which aggregates the region similarity scores into a global similarity value. In particular, among all valid matchings that satisfy the constraints of the specific matching type, the rationale is to select the one that maximizes the aggregated score. This can be modeled as an optimization problem whose solution depends on the particular choice of the scoring function. For the most commonly used functions efficient algorithms exist: For example, when using the average function with the $1 - 1$ matching type, the problem takes the form of the well-known *assignment problem*, while with the $n - m$ (EMD) matching type (see above), this corresponds to the *transportation problem*. For both such problems the optimal solution can be efficiently found without performing an exhaustive search; however, in the general case, the optimal matching can not be easily found. Figure 3 shows an example of matching for a query image I^q with three regions and a DB image I with four regions under the assumption of $1 - 1$ matching type and the average scoring function. Similarities between regions of I^q and regions of I are arranged in a matrix. Circled cells are those involved in the matching. Note that, since $n < m$, in valid matchings $4 - 3 = 1$ region of I remains unmatched.

Finally, given the query image I^q and two DB images I_1 and I_2, image I_1 will be considered more similar than I_2 to I^q iff $s_I(I^q,I_1) > s_I(I^q,I_2)$ holds. In such a way it is possible to linearly order DB images and return to the user only the k highest scored ones.

The main limitation of the *k*-NN ranking model is that the choice of a particular scoring function clearly influences the final result, i.e., different scoring functions will likely yield different results. This can lead to missing relevant images, because the choice of the scoring function is a difficult task for the user. Moreover, the use of scoring functions limits the expressive power of queries that can be submitted to the system, since all of them will always define a simple linear order on objects. This might prevent their applicability to modern multimedia systems asking for more flexibility in querying [2]. In the BMO model, the result of the query depends on a specific *preference relation* $\succ_p$, where $\succ_p$ is only required to define a strict partial order over images. Image $I_1 \in \mathcal{IDB}$ is in the query result if and only if no other image $I_2 \in \mathcal{IDB}$ is *better* than (or *dominates*) I_1 according to $\succ_p$. Clearly,

preference relation $\succ_p$ is based on regions similarity scores (see Fig. 3).

Thus, even if region scores are numerical (by definition), the BMO ranking model does not need to aggregate them using a scoring function. Actually, the result of a BMO query with image I^q is the set of undominated images in $\mathcal{IDB}$, i.e., all and only those images for which no better image (with respect to I^q and to $\succ_p$) can be found in the database.

When considering together the matching type and ranking model coordinates, different scenarios are derived. In the following, algorithms for k-NN and BMO queries are provided by considering the simplest way to solve the image querying problem, i.e., when using a sequential scan of the DB. Note that the efficiency of such solutions is clearly quite limited. It is possible to derive efficient algorithms [3,11,2] by exploiting index structures, such as *multi-dimensional or metric indices* (built either on regions of the DB images or on the DB images themselves).

The steps described in Algorithm 1 show the logic of the sequential algorithm for k-NN queries, named k-NNSeq, to determine the k nearest neighbors of the image query I^q: Given the image query I^q, the scoring function *sf*, and the cardinality of the result k, the algorithm correctly returns the k images that are most similar to I^q according to *sf*. This algorithm covers both cases of $1-1$ and $n-m$ matchings.

Algorithm 2, named BMOSeq, describes the main steps for the sequential evaluation of BMO queries, with the assumption of $1-1$ matching: Given the image query I^q and the preference relation $\succ_p$, the algorithm correctly returns set of undominated images with respect to the query I^q.

It is also important to consider the type of images the user is interested in. The above description deals with *full image* search, i.e., when the user is interested in all regions of the query, but other possibilities exist that introduce minor modifications in the query evaluation process.

For example, *part-of* queries request DB images whose regions are all matched to some query region (the presence of unmatched query regions is not penalized). Two other query types are introduced when the user is given the possibility to select, possibly exploiting a

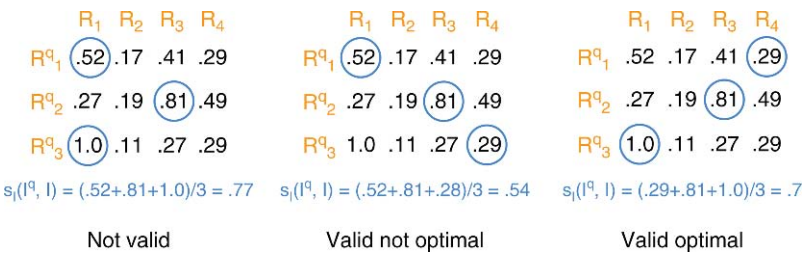


Image Querying. Figure 3. Example of similarity assessment between images I^q and I when adopting the 1–1 matching type and the average scoring function: not valid matching (*left*), valid not optimal matching (*center*), and valid and optimal matching (*right*). If an alternative matching type (e.g., the general $n-m$ matching) is considered, the left matching could become valid (and optimal).

Algorithm 1: k-NNSeq

Require: I^q: query image, k: cardinality of result, $\mathcal{IDB}$: image DB, sf: scoring function
Ensure: set of relevant k images
  for all images $I \in \mathcal{IDB}$ **do**
    for all regions $R_j \in I$ **do**
      for all regions $R_i^q \in I^q$ **do**
        compute $s_R(R_i^q, R_j)$
    compute matchings between regions $R_i^q \in I^q$ and regions $R_j \in I$
    select the matching that maximizes $s_I(I^q, I)$ by means of sf
  return the k images having the highest overall similarity scores s_I

Algorithm 2: BMOSeq
Require: I_q: query image, $\mathcal{IDB}$: image DB, $\succ_p$: preference relation
Ensure: set of undominated images
$\mathcal{U} \leftarrow \emptyset$
for all images $I_1 \in \mathcal{IDB}$ **do**
  for all regions $R_j \in I_1$ **do**
    for all regions $R_i^q \in I^q$ **do**
      compute $s_R(R_i^q, R_j)$
  $\mathcal{U} \leftarrow \mathcal{U} \cup \{I_1\}$
  for all images $I_2 \in \mathcal{U}$ **do**
    if $I_1 \succ_p I_2$ **then**
      $\mathcal{U} \leftarrow \mathcal{U} \setminus I_2$
    else if $I_2 \succ_p I_1$ **then**
      $\mathcal{U} \leftarrow \mathcal{U} \setminus I_1$
      break (**for** at line 7)
return images in $\mathcal{U}$

suitable graphical interface, only a subset of query regions: In *partial match* queries the user is looking for DB images containing selected regions of the query (the presence of other regions in the DB image should not be penalized); on the other hand, with a *contains* query DB images are requested to contain selected query regions only (other existing regions reduce the image similarity, differently from the case of part-of queries).

Key Applications

Image querying is an important tool for many modern multimedia applications, such as *digital libraries*, *e-commerce* (where electronic catalogues have to be browsed and/or searched), *edu-tainment* (for example, to search in clipart repositories, or to search and organize personal photo albums in mobile phones or PDAs).

Another interesting application area is the one related to (semi-)automatic *image annotation techniques*, which can be based on assigning to an unlabeled image I the keywords associated to the DB images most similar to I. Finally, image querying techniques have been also profitably used in *image classification*, for example, to search for similar logo images, for copyright infringement issues, and for the detection of pornography images.

Cross-references

- ▶ Annotation-based Image Retrieval
- ▶ Feature Extraction for Content-Based Image Retrieval
- ▶ Image Database
- ▶ Image Retrieval and Relevance Feedback
- ▶ Image Segmentation
- ▶ Indexing and Similarity Search
- ▶ Top-K Selection Queries on Multimedia Datasets
- ▶ Video Querying

Recommended Reading

1. Ardizzoni S., Bartolini I., and Patella M. Windsurf: region-based image retrieval using wavelets. In Proc. 1st Int. Workshop on Similarity Search, 1999, pp. 167–173.
2. Bartolini I., Ciaccia P., Oria V., and Özsu T. Flexible integration of multimedia sub-queries with qualitative preferences. Multimedia Tools Applicat., 33(3):275–300, June 2007.
3. Bartolini I., Ciaccia P., and Patella M. A sound algorithm for region-based image retrieval using an index. In Proc. 4th Int. Workshop on Query Processing and Multimedia Issues in Distributed Systems, 2000, pp. 930–934.
4. Carson C., Thomas M., Belongie S., Hellerstein J.M., and Malik J. Blobworld: a system for region-based image indexing and retrieval. In Proc. 3rd Int Conf. on Visual Information Systems, 1999, pp. 509–516.
5. Flickner M., Sawhney H.S., Ashley J., Huang Q., Dom B., Gorkani M., Hafner J., Petkovic D., Steele D., and Yanker P. Query by image and video content: The QBIC system. IEEE Computer, 28(9):23–32, September 1995.
6. Natsev A., Rastogi R., and Shim K. WALRUS: a similarity retrieval algorithm for image databases. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1999, pp. 396–405.
7. Rubner Y. and Tomasi C. Perceptual Metrics for Image Database Navigation. Kluwer Academic, Boston, MA, December 2000.
8. Smeulders A.W.M., Worring M., Santini S., Gupta A., and Jain R. Content-based image retrieval at the end of the early years. IEEE Trans. Pattern Analy. Machine Intell., 22(12):1349–1380, December 2000.
9. Smith J.R. and Chang S.-F. VisualSEEk: A fully automated content-based image query system. In Proc. 4th ACM Int. Conf. on Multimedia, 1996, pp. 87–98.
10. Wang J.Z., Li J., and Wiederhold G. SIMPLIcity: semantics-sensitive integrated matching for picture libraries. IEEE

Trans. Pattern Anal. Machine Intell., 23(9):947–963, September 2001.
11. Weber R. and Mlivoncic M. Efficient region-based image retrieval. In Proc. Int. Conf. on Information and Knowledge Management, 2003, pp. 69–76.

Image Representation

Valerie Gouet-Brunet
CNAM Paris, Paris, France

Synonyms

multimedia; File format; Image compression; Image metadata; Image content; Image standards

Definition

In computer science, the representation of an image can take many forms. Most of the time, it refers to the way that the conveyed information, such as color, is coded digitally and how the image is stored, i.e., how is structured an image file. Several open or patented standards were proposed to create, manipulate store and exchange digital images. They describe the format of image files, the algorithms of image encoding such as compression as well as the format of additional information often called metadata. Differently, the visual content of the image can also take part in its representation. This more recent concept has provided new approaches of representation and new standards, gathered together into the discipline named *content-based image indexing*.

Historical Background

The first use of digital images began in the early 1920s with the technological development of facsimile transmission, and in particular with the Bartlane cable transmission system that was the first system that translated pictures into a digital code for efficient picture transmission. Later, in the 1960s and 1970s, the advent of digital image technology is closely tied to the development of government programs for space exploration and espionage and also to medical research with the invention of computerized axial tomography. With the availability of the CCD image sensors (charge-coupled device), the private sector also began to make significant contributions in the development of digital cameras. Dissemination of digital images quickly required the proposal of common frameworks for representing images as an improvement of proprietary formats most of the time under license and not easily exchangeable. In the mid-1980s, image format TIFF was created by the company Aldus with the aim of agreeing on a common file format for bitmapped images issued from scanners. In parallel, researchers at Xerox PARC had developed the first laser printer and had recognized the need for a standard means of defining page images. After several fruitless attempts, Adobe Systems proposed the PostScript language in 1982, quickly adapted for driving laser printers. Since then, other file formats dedicated to digital images were also proposed, such as GIF (1987), JPEG (1992), PDF (1993), PNG (1995) and SVG (1998). Today, a great effort is done to propose standard formats able to preserve data integrity for archiving purposes, to migrate easily to future technologies as well as to provide efficient compression for access and dissemination. In 2000, the standard JPEG 2000 was proposed to deal with these objectives for a large range of application domains. Moreover Ambitious programs, MPEG-7 and MPEG-21, started in the late 1990s, are focusing on the harmonization of methods for representing, storing, sharing and accessing multimedia contents (text, audio, image and video) in an unified framework.

Foundations

Basics of Image Representation

Digital images can be classified into two main categories: vector graphics and bitmapped images (also called raster images). Vector images are geometrical 2D objects created with drawing software or CAD (computer-aided design) systems. They are represented with geometrical primitives such as points, lines, curves, and shapes or polygons, which are all based upon mathematical equations. Unlike bitmaps that are resolution-dependent, vector images are scalable, which means that the scale at which they are shown will not affect their appearance. Such images are dedicated to the representation of images with simple content, such as diagrams, icons or logos.

A bitmapped image is composed of a set of dots or squares, called pixels (for picture elements), arranged in a matrix of columns and rows. Each pixel has a specific color or shade of gray, and in combination with neighboring pixels it creates the illusion of a continuous tone image. Unlike human vision, sensors that capture images are not limited to the visual band of the electromagnetic spectrum. Digital images can cover almost the entire spectrum, ranging from gamma to radio waves.

Managing bitmapped images requires the choice and manipulation of several parameters such as:

- *Color model.* A color model is an abstract mathematical model describing the way human color vision can be represented as tuples of numbers. From this model, the combination of dedicated primary colors provide all the colors possible, that are embedded in the corresponding color space. RGB, CMYK, CIELAB and CIELUV are the most known color models and spaces.
- *Dynamic range.* The dynamic range of a digital image, also called color depth, determines the maximum range of gray level or color values carried by each pixel. The number of bits used to represent each pixel determines how many colors can appear in the image. Photographic-quality images are usually associated with 24-bit dynamic range, such as in the JPEG format.
- *Resolution.* Resolution expresses the density of elements, pixels for instance, within a specific area. This term does not have any sense when dealing with digital images as files, but it applies when associating a digital image with a physical support, such as display on a screen, printing on a printer or capture with a scanner. Resolution is classically represented in terms of dpi (dots per inch) unit, which was originally the unit adopted for printing. Appearance of bitmapped images, which are made up of a fixed grid of pixels, clearly depends on the resolution chosen, unlike vector images that are scalable and then have the same appearance whatever the dimensions chosen for visualization.

Image Compression

Image compression is the process of shrinking the size of digital image files. Methods have in common the processes of finding and storing redundant data (e.g., pixels with similar color information) more efficiently or of eliminating information that is difficult for the human eye to see [8]. Compression algorithms are especially characterized by two factors: compression ratio and generational integrity. Compression ratio is the ratio of compressed image size to uncompressed size and generational integrity refers to the ability for a compression scheme to prevent or mitigate loss of data, and therefore image quality, through multiple cycles of compression and decompression. Lossless compression ensures that the image data is retained, as with Run-length encoding, Huffman coding and LZW coding. On the other hand, lossy compression schemes involve intentionally sacrificing the quality of stored images by selectively discarding pieces of data. Most of them have a compression ratio that can be parameterized by the user, to optimize the results for each situation, such as the standards JPEG and JPEG 2000.

Run-Length Encoding Run-length encoding (RLE) is probably the most simple form of lossless data compression: sequences in which the same data value occurs in many consecutive data elements are stored as a single data value and count. Image data is normally run-length encoded in a sequential process that treats the image data as a 1D stream, line by line, column by column or diagonally in a zigzag fashion. Also used in fax machines, common digital image formats for run-length encoded data include TGA, PCX and is possible with BMP, TIFF and JPEG.

Huffman Coding Created in 1951 by David A. Huffman, the Huffman coding is an entropy encoding algorithm used for lossless data compression. The basic idea of this algorithm is to code with few digits the most common input symbols of a document. Each symbol is encoded by using a variable-length code table, where the codes are defined according to the estimated probability of occurrence for each possible symbol. The technique works by creating a binary tree of nodes that contain symbols with their probability for leaf nodes and cumulated probabilities for internal nodes. Traversing this binary tree from the root to the leaves, with the convention '0' when following the left child and '1' when following the right one, allows associating a bit string with each symbol. The result is a prefix-free code: the bit string representing some particular symbol is never a prefix of the bit string representing any other symbol. Today, Huffman coding is often used during the final process of some other compression methods such as JPEG and MP3.

LZW Coding Lempel-Ziv-Welch (LZW) is a lossless data compression algorithm created by Abraham Lempel, Jacob Ziv, and Terry Welch. It was published in 1984 as an improved version of the LZ77 and LZ78 algorithms published by Lempel and Ziv in 1977 and 1978. The technique was patented by IBM and Unisys Corporation in 1983 and the patents expired in 2003. The compression algorithm build a string translation table that is based on fixed-length codes (usually

12-bit). As the system character-serially examines the document, if the string read is not stored in the table, a new code is created in the table and associated with this string, otherwise the current string is encoded with an existing code. This algorithm became very widely used after it became part of the GIF image format in 1987. In contrast to other compression techniques such as JPEG, it allows preserving very sharp edges, suitable with line art images often stored in GIF format.

JPEG Compression JPEG is the most common image format used for compressing and storing digital cameras and other photographic image capture devices. "JPEG" stands for Joint Photographic Experts Group, the name of the committee that created the standard. The group was organized in 1986, issuing a standard in 1992, which was approved in 1994 as ISO 10918-1 standard. The associated algorithm, described in Fig. 1, is a lossy compression technique suited for photographs and paintings of realistic scenes, but not for line drawings and other textual or graphics, where the compression cause noticeable artifacts on sharp contrasts between adjacent pixels. This algorithm stands on the representation of the image in the frequency domain by using a two-dimensional DCT (Discrete Cosine Transform), that describes the variability of the signal in terms of low-level and high-level frequencies. The human eye notices small differences in brightness over a relatively large area, but does not distinguish the exact strength of a high frequency brightness variation very well. Consequently, the amount of information in the high frequency components of the DCT can be neglected without drastically affecting perceptual image quality: the DCT components are divided by factors of a quantization matrix that increase with the spatial frequency, and then rounded to the nearest integer. This is the main lossy operation in the whole process. Typically, many of the higher frequency components are rounded to zero and the other components become small numbers, which take many fewer bits to store [10].

File Formats

There are a lot of image formats for vector graphics as well as for bitmapped images [3,6]). Most of them are open standards and patent expired for the others. Vector image formats contain a geometric description of the objects which can be rendered smoothly at any desired size. Among the most common formats, there are:

- **SVG** (Scalable Vector Graphics) is an open XML-based standard created in 1998 and developed by the World Wide Web Consortium to address the need for a versatile, scriptable and all-purpose vector format for the web and otherwise.
- **EPS** (Encapsulated PostScript) is a standard file format created by Adobe Systems in the mid-1980s. It follows DSC (Document Structuring Conventions) rules that are a set of standards for PostScript.

File formats abound for bitmapped images, but many digital imaging projects have settled on the formula of TIFF, JPEG, GIF and also PNG files:

- **TIFF**, for Tagged Image File Format, is a file format for storing images such as photographs as well as graphics. It was originally created by the company Aldus, was then under the control of Adobe Systems and is now in the public domain. TIFF supports several lossless and lossy techniques of image

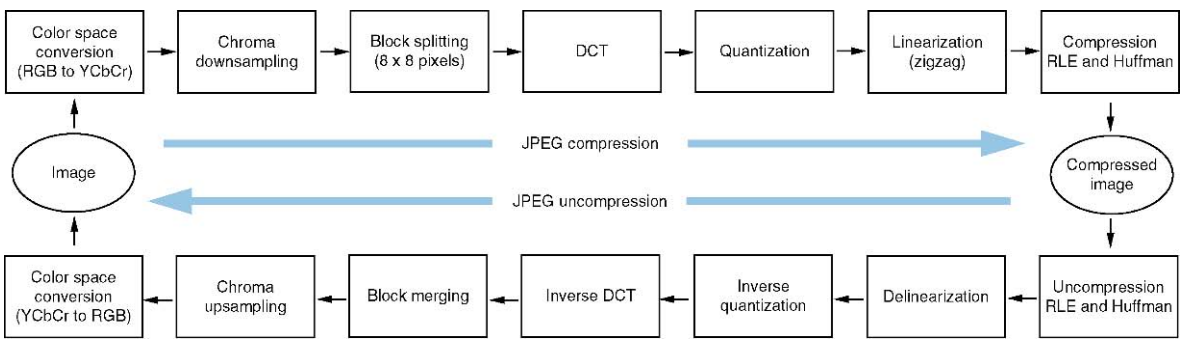


Image Representation. Figure 1. Main steps of the JPEG algorithm for compression and uncompression.

compression, such as LZW, Huffman coding and JPEG. The ability to store image data in a lossless format makes TIFF files a useful method for archiving images and preservation purposes. However, there are so many different implementations of TIFF that many applications can read certain types of TIFF images but not others.

- **JPEG**, for Joint Photographers Experts Group, is a file format that was developed specifically for high-quality compression of photographic images in a 24-bit RGB color model. It is generally employed for online presentation and dissemination; the associated lossy algorithm for compression makes it unappropriated for archiving purposes. The file format associated is JFIF (JPEG File Interchange Format, 1992), a public domain storage format for JPEG compressed images. Unlike TIFF, JFIF does not allow for the storage of associated metadata, a failing that has led to the development of SPIFF (Still Picture Interchange File Format), which is now the international standard.
- **GIF**, for Graphics Interchange Format, is an 8-bit image format for indexed colors, that was introduced by CompuServe in 1987 and has since come into widespread usage for art images such as diagrams or logos with a limited numbers of colors. The original version was called 87a; in 1989, CompuServe developed an enhanced version, called 89a, that supports animations. Due to infringement of Unisys' patent on the LZW compression technique used with GIF, in 1995 CompuServe proposed the **PNG** format (Portable Network Graphics) as a replacement for the GIF format without patent license. PNG offers a better and lossless compression technique called DEFLATE (that combines LZ77 with Huffman coding). Since 2003, it is an international standard.

Today, the status of TIFF as the de facto standard format for archival digital image files is challenged by other formats such as PNG and JPEG 2000, that are able to preserve data integrity as well as to provide efficient compression ratios for access and dissemination.

Metadata Representation

Metadata are commonly defined as "data about data." They constitute the documentation or a structured description associated with a document. Image files automatically include a certain amount of metadata that are stored in an area of the file defined by the file format and called *the header*, but information may also be stored externally.

In the widely used TIFF format, the term "Tagged" indicates that developers can define and apply dedicated tags to enable them to include their own proprietary information (called " private tags") inside a TIFF file without causing problems of compatibility. More recently, Exif format (Exchangeable image file format) was created by the Japan Electronic Industries Development Association. The latest version was published in 2002 and while the specification is not currently maintained by any industry or standards organization, its use by camera manufacturers is nearly universal. Exif fields are generated at the creation of the image and should not be modified after with the aim of including additional information like title or keywords. To do this, other formats are recommended, such as XMP (eXtensible Metadata Platform). This last is an XML-based standard for creating, processing and storing standardized, extensible and proprietary metadata, created by Adobe Systems in 2001 [1]. XMP metadata can be embedded into a significant number of popular file formats: it is used in PDF and other image formats such as JPEG, JPEG 2000, GIF, PNG, TIFF and EPS.

In parallel to generic standards, standards dedicated to specific image applications also exist, such as DICOM (Digital Imaging and Communications in Medicine). This is a standard created in 1992 and widely adopted by hospitals, for handling, storing, printing and transmitting information in medical imaging [7]. It includes a file format definition and a network communications protocol.

Metadata constitute the documentation of all aspects of digital files essential to their persistence, usefulness and access. Images without appropriate metadata may become hard to view, migrate to new technology, or to access among large volumes of images. When annotation is unappropriate or is missing, the representation of the visual content of images by image analysis may be an interesting alternative, as described in the following.

Image Content Representation

Born in early 1990s, Content-Based Image Indexing (CBIR) is a discipline that exploits techniques of image analysis and databases [2]. Indexing an image by its content consists of automatically extracting structures that describe the visual content relevantly for the considered application. These structures can

describe the visual content of an image globally or locally by characterizing its distribution of color, shape and texture, or parts or objects of the image. The visual structures exhibited are considered as the index of the image, they are digitally represented with one or several multidimensional vectors called *signature* of the image. According to this description, searching for a particular image in a database of images consists of searching in multidimensional spaces.

Key Applications

JPEG 2000 is a standard that gathers an image file format and an algorithm of image compression, created by the Joint Photographic Experts Group committee in 2000 [9]. The name refers to all of the eleven parts of the standard, some of them are now published as an International Standard, while others are under development.

The coding algorithm of JPEG 2000 is similar to the JPEG one, it mainly differs in the use of wavelets instead of a DCT. Wavelets provide a decomposition of the image into a pyramid of sub-images which store different levels of resolution of the image. They can be of two types, according to the objective: (1) a Daubechies wavelet transform, that requires quantization to reduce the amount of bits representing data, as JPEG does, and then imposes lossy compression; (2) a rounded version of Le Gall wavelet transform, that uses only integer coefficients and then does not require quantization, providing lossless coding.

On average, JPEG 2000 gains up to about 20% lossy compression performance for medium compression rates in comparison to the first JPEG standard. Moreover, the edges remain sharper and more contrasting than with JPEG where blocky artifacts can also appear. The aim of this standard is not only improving compression performance but also adding features, among which transmission error resilience and region of interest (ROI). This last offers the opportunity of storing parts of the same picture using different quality. Some parts of particular interest such as faces, can be stored with higher quality, to the detriment of other ones where low quality/high compression can be tolerated. JPEG 2000 also allows delivering these parts *before* other parts of the image.

The JPEG 2000 standard defines two file formats that support embedded XML metadata: JP2, which supports simple XML, and JPX, which has a more robust XML system based on an embedded metadata initiative of the International Imaging Industry Association (the DIG35 specification). However, as of this writing, commercial implementations for JPEG 2000 are just beginning to appear. The democratization of JPEG 2000 is presently less important than the JPEG standard: It is supported in several web browsers, but is not generally used on the World Wide Web. Nevertheless, this standard will take more place in a near future, because it has been developed to efficiently deal with many applications and markets such multimedia consumers applications, military/surveillance, medical imagery, editing and storage, etc.

Future Directions

The continuing drop in prices of computers and storage, the ocean of image/video/audio data produced and the expansion of networking and communication bandwidth via the Internet or the HDTV (High-definition television) greatly contribute to the production and dissemination of larger volumes of multimedia content. Many techniques of different disciplines of computer science have been studied and developed for managing text, image, video and audio. But today, there is a need in the development of an unified framework for the creation, representation, storage, access, delivery, management and protection of multimedia contents. Standardization goes in this direction by proposing new standards for managing these contents jointly, such as the international standards MPEG-21 and MPEG-7. MPEG-21 is a standard started in 1999 by MPEG (Moving Picture Experts Group) and now normalized as ISO/IEC 21000. Its main objectives are to define an open framework for multimedia applications and more precisely to provide a standardized structure for various media contents and to facilitate their access, delivery, management and protection [4]. MPEG-7 is another ISO/IEC standard started by MPEG in 1998 and formally called *Multimedia Content Description Interface*. It aims at specifying a standard set of descriptors and description schemes dedicated to various types of multimedia information. One of its main objectives is to provide unified and efficient searching, filtering and content identification methods for these media [5].

Cross-references

▶ Feature Extraction for Content-Based Image Retrieval
▶ Image
▶ Image Content Modeling
▶ Image Metadata
▶ Image Salient Points and Features

▶ Image Segmentation
▶ Image Segmentation and Features
▶ Metadata
▶ Multimedia Data
▶ Multimedia Data Storage
▶ Multimedia Databases
▶ Multimedia Metadata
▶ Video
▶ Video Representation
▶ Visual Content Analysis
▶ Visual Representation

Recommended Reading

1. Adobe XMP main page: http://www.adobe.com/products/xmp/index.html. 5
2. Datta R. Joshi D. Li J. and Wang J. Z. Image retrieval: Ideas, influences, and trends of the new age. ACM Comput. Surv. 40 (2), 2008.
3. Fileformat.info. http://www.fileformat.info/format/could.htm. 3
4. Ian S., Burnett, Pereira F., Van de Walle R., and Koenen R. The MPEG-21 Book. Wiley, 2006.
5. Manjunath B.S., Salembier P., and Sikora T. Introduction to MPEG-7: Multimedia Content Description Interface. Wiley & Sons, 2002.
6. Murray J. D. and vanRyper W. Encyclopedia of Graphics File Formats. 2nd ed. O'Reilly, 1996.
7. Oleg S., Pianykh. Digital Imaging and Communications in Medicine (DICOM): A Practical Introduction and Survival Guide. Springer, 2008.
8. Salomon D., Motta G., and Bryant D. Data Compression: The Complete Reference. Springer, 4th edition, 2006.
9. Taubman D. S. and Marcellin M. W. JPEG 2000: Image Compression Fundamentals, Standards and Practice. Kluwer International Series in Engineering and Computer Science, Secs 642, 52001.
10. Wallace G. K. The JPEG still picture compressin standard. Commun. ACM, 34(4):30–44, 3 April 1991.

Image Similarity

Tao Mei[1], Yong Rui[2]
[1]Microsoft Research Asia, Beijing, China
[2]Microsoft China R&D Group, Redmond, WA, USA

Synonyms

Image Distance; Visual Similarity; Similarity Measure

Definition

Given a pair of images each described by a feature set, image similarity is defined by comparing the feature set on the basis of a similarity function. In a typical Visual Information Retrieval system, while searching for a query image among the elements of the data set of images, knowledge of the domain will be expressed by formulating a similarity measure between the query and data set based on some visual features. Therefore, measuring meaningful image similarity consists of two intrinsic elements: finding a set of features for adequately describing the image content and finding a suitable metric for assessing the similarity on the basis of feature space. The feature set can be computed globally for the entire image or locally for a small group of pixels such as regions or objects. The similarity measure can be different depending on the types of features. Typically, the feature space is assumed to be Euclidean. The algorithm for image similarity aims to essentially reduce the semantic gap between low-level features and high level semantics as much as possible.

Historical Background

Comparing two images is the fundamental operation for many Visual Information Retrieval systems, in which the user selects a query image and image similarity to the query according to the given criteria are retrieved and presented [11]. Figure 1 shows the role of image similarity in the context of multimedia information retrieval. A set of visual features is first computed for each image or video frame in the database and for the query. Then, given a query image, image similarity is computed for each pair of query and image based on the feature space and some distance metric, and may be further tuned in an interactive way according to user feedback. The images which are visually, semantically, or perceptually similar to the query are finally presented to the user.

A wide variety of methods for image similarity have been devised with research expanding in content-based multimedia retrieval in the last decade. In earlier work, image content is usually described by a set of global features such as color, texture, shape, and so on [8]. The advantage of global features is the high speed for both feature extraction and similarity computation. The features are then transformed and represented by a set of vectors. Image similarity is computed based on different distance metrics on these feature vectors. For example, color histogram serves as an effective representation of the distribution of colors in an image. A color histogram is created by plotting the number of pixels in the image

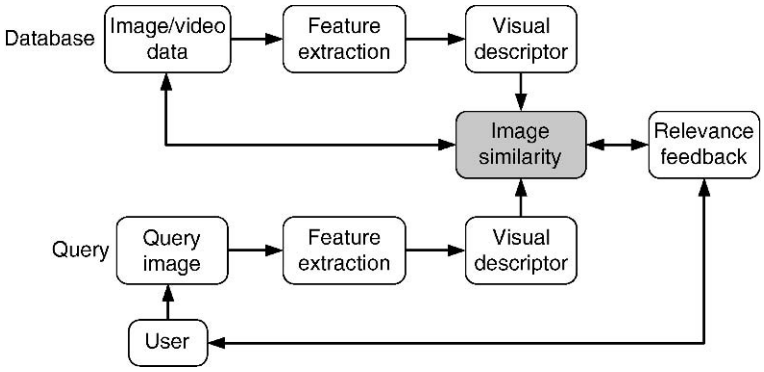


Image Similarity. Figure 1. Image similarity in the context of multimedia information retrieval.

with a particular range of quantized color value in some color space. The Minkowski-form distance or Histogram intersection can be used to compute the similarity between two histograms [5]. However, using an individual type of feature cannot well characterize image content. The multimodal methods combine the similarities from different types of features. The simplest way for combination yielding a scalar is to compute the weighted sum of these similarities based on the assumption that the features are not of the same importance. The weights can be decided manually or by relevance feedback mechanism [9].

As global features are often too rigid to represent an image, there has been a paradigm shift from global feature representation to local descriptors. Local features are computed based on a subset of the image usually in neighborhood of a salient point or a region. It is deemed that local features are closer to the perception of human visual system, as they often correspond to meaningful image components such as salient points, rigid objects, and homogenous regions [2]. The similarity is then computed based on user selected points/objects/regions or weighted sum of all of them. For example, an image is represented by a set of regions where each region is assigned a feature set and a weight indicating the importance of the corresponding region [4]. The image-to-image similarity is measured by the Earth Mover's Distance (EMD) [7] or an Integrated Region Matching method (IRM) [2]. The recent work in computer vision treated image as a set of salient points and extracted Scale Invariant Feature Transform (SIFT) features for each point [6]. Image matching is performed based on the Euclidean distances between pairs of salient points or the histogram comparison built upon a codebook.

Foundations

By the nature of the task of image retrieval, image similarity boils down to two intrinsic problems: (i) how to describe an image using a set of visual features, and (ii) how to assess the similarity between two images based on these features. Figure 2 shows the paradigms of image similarities, including three types of features for describing image content, formulation of the features, distance metrics for computing the similarities, and techniques used for computing the distance. The distance metrics used for image similarity depend on the selection of features and their corresponding formulations.

Image Descriptors

The preliminary step for image similarity is the description of image content. Research has proceeded toward effectively characterizing image content by a variety of visual features (or referred to as *signatures*). These features can be categorized into three types according to the pixels used, i.e., global, regional, and local features. The global features are extracted over the entire image or sub-image based on grid partition, the regional features are computed based on the results of image segmentation which attempts to segment the image into different homogenous regions or objects, while the local features aim at robust descriptors invariant to scale and orientation based on local maxima.

In global features, the most typical ones are color, texture, and shape which are widely used in many retrieval systems [5]. Color histogram is an effective and easy-to-compute representation of the color distribution in an image. It is also robust to the translation and rotation about the view axis, as well as slight occlusion. A color histogram is created by counting the

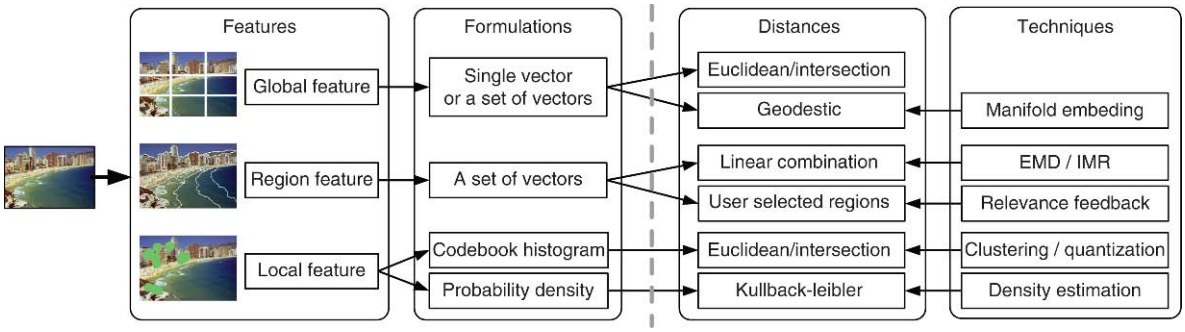


Image Similarity. Figure 2. Paradigms of image similarities, including the corresponding features, formulations, and distance metrics, and techniques. This figure is designed similar to the figure used in [5].

pixels falling into a quantized bin in a specific color space, such as RGB, HSV, and LUV spaces. In addition, color moments are also popular color features. Usually the first three order (i.e., *mean*, *variance*, and *skewness*) color moments are suitable enough to represent the color distribution of image. An alternative way to compute color histogram and moments is to divide an image into non-overlapping blocks, compute histogram or moments for each block, and concatenate all the features into one vector. To deal with large-scale image database, a compact descriptor is proposed in [12] where each image is converted to a K-bit (K is no more than 32) hash code according to its content. Other color features include color coherence, color correlogram, and so on. Texture features are intended to capture the granularity and repetitive visual patterns in an image. The basic texture features include Tamura, multi-resolution simultaneous auto-regressive model (MRSAR), Gabor filter, wavelet transform, and Wold features [11,5]. In contrast to color and texture features, shape features are usually computed after images being segmented into regions. The shape features capture the local geometrical properties. Typical shape features include normalized inertia [1], moment invariants, Fourier descriptors [5], and so on. Using global features, an image can be represented by a single vector corresponding to a uni-modaltiy or a set of vectors and weights corresponding to multi-modal features and their importances.

The regional features are similar with global features, except that they are computed over a local region with homogeneous texture rather than the whole image. The most widely used features for describing a region include color moment [1], color correlogram [8], wavelet transform texture, and normalized inertia [1]. As a result, an image is represented by a set of vectors and weights each corresponding to a region.

Local invariants such as salient points from which descriptors are derived, traditionally used for stereo matching and object recognition, are being used in image similarity. For example, the algorithm proposed by Lowe [6] constructs a scale space pyramid using Difference-of-Gaussian (DoG) filters and finds the local 3D maxima (i.e., salient point) on the pyramid. A robust Scale Invariant Feature Transform (SIFT) descriptor is computed for each point. An image is thus represented by a set of salient points and 128 dimensional SIFT features, or a histogram of codewords built upon a large visual vocabulary.

Similarity Measures

The similarity measures are based on comparisons between the features associated with images. As shown in Fig. 2, since each type of feature can have its own mathematic formulation, the distance metrics and the corresponding techniques for computing the distances are different.

Considering that an image $\mathbf{I}$ is represented by a single vector $\mathbf{f} = (f(1),...,f(M))$ (e.g., histogram or some distribution) in which each dimension is independent of each other and has equal importance, the most widely adopted similarity is computed based on Minkowski-form distance, defined as

$$\mathbf{D}_{L_p}(\mathbf{I}^0, \mathbf{I}^1) = \left\{ \sum_{i=1}^{M} |f^0(i) - f^1(i)|^p \right\}^{1/p} \quad (1)$$

where $f^0(i)$ and $f^1(i)$ denote i-th feature of image $\mathbf{I}^0$ and $\mathbf{I}^1$, respectively. For $p = 2$, this yields the Euclidean distance. The histogram intersection is a special case of L_1 distance, defined as

$$\mathbf{D}_{HI}(\mathbf{I}^0, \mathbf{I}^1) = \frac{\sum_{i=1}^{M} \min(f^0(i), f^1(i))}{\sum_{i=1}^{M} f^1(i)} \quad (2)$$

It has been shown that histogram intersection is fairly sensitive to the changes of image resolution, occlusion, and viewing point [5]. A distance robust to noise is Jeffery distance (JD) which is based on the Kullback-Leibler divergence (KL) given by

$$\mathbf{D}_{KL}(\mathbf{I}^0,\mathbf{I}^1)=\sum_{i=1}^{M} f^0(i)\log\frac{f^0(i)}{f^1(i)} \tag{3}$$

Although it is often intuited as a distance metric, the KL divergence is not a true metric since it is not symmetric. The JD distance is defined as

$$\mathbf{D}_{JD}(\mathbf{I}^0,\mathbf{I}^1)=\sum_{i=1}^{M}\{f^0(i)\log\frac{f^0(i)}{m_i} + f^1(i)\log\frac{f^1(i)}{m_i}\} \tag{4}$$

where $m_i=\frac{f^0(i)+f^1(i)}{2}$. In contrast to KL divergence, JD distance is symmetric and numerically stable when comparing two empirical distributions. Hausdorff distance [2] is another matching method which is symmetrized by computing additionally the distance with image $\mathbf{I}^0$ and $\mathbf{I}^1$ reversed and choosing the larger one of the two distances

$$\mathbf{D}_H(\mathbf{I}^0,\mathbf{I}^1)=\max\left(\max_i\min_j\ \mathbf{d}(f^0(i),f^1(i)),\ \max_j\min_i\ \mathbf{d}(f^0(i),f^1(i))\right) \tag{5}$$

where $\mathbf{d}(,)$ can be any form of distance such as L_1 and L_2 distances. The Mahalanobis distance metric deals with the case that each dimension of vector is dependent and has different importance, given by

$$\mathbf{D}_M(\mathbf{I}^0,\mathbf{I}^1)=\sqrt{(\mathbf{f}^0-\mathbf{f}^1)^{\mathrm{T}}\mathbf{C}^{-1}(\mathbf{f}^0-\mathbf{f}^1)} \tag{6}$$

where $\mathbf{C}$ is the covariance matrix of the feature vectors. The above distance measures are all derived from a linear feature space which has been noted as the difficulty in measuring perceptual or semantic image distance. Manifold ranking replaces traditional Euclidean distance by the geodesic distance in a non-linear manifold [13]. The similarity is often estimated based on a distance measure $\mathbf{d}(,)$ and a positive radius parameter σ along a manifold

$$\mathbf{D}_{MR}(\mathbf{I}^0,\mathbf{I}^1)=\exp\left\{-\frac{\mathbf{d}(\mathbf{I}^0,\mathbf{I}^1)}{\sigma}\right\} \tag{7}$$

where L_1 distance is usually selected for $\mathbf{d}(,)$.

In a more general situation, an image $\mathbf{I}$ is represented by a set of vectors and weights $\{(\mathbf{f}_i,\omega_i)\}$(i=1, 2,...,M), where M is the number of modalities (e.g., color, texture, or shape). Each vector (also referred to as distribution) corresponds to a specific region or modality and the weight indicates the significance of associating this vector to the others. Note that the weight ω_i can have the same dimension to $\mathbf{f}_i$ indicating that each dimension of the features in a single vector has the same importance to each other, or just a real value indicating the importance of the entire vector. The simplest way to compute the similarity from different modalities is the weighted sum of the similarity from each single vector. The Earth Mover's Distance (EMD) represents a soft matching scheme for features in the form of set of vectors [7]. The EMD "lifts" the distance from individual features to full distributions. The EMD distance is given by

$$\mathbf{D}_{EMD}(\mathbf{I}^0,\mathbf{I}^1)=\frac{\sum_{i=1}^{M^0}\sum_{j=1}^{M^1}s_{ij}\mathbf{d}(\mathbf{f}_i^0,\mathbf{f}_j^1)}{\sum_{i=1}^{M^0}\sum_{j=1}^{M^1}s_{ij}} \tag{8}$$

where $\mathbf{d}(,)$ is the ground distance between two vectors which can be defined in diverse ways depending on the system, s_{ij} minimizes the value of (8) subject to the following constraints:

$$s_{ij}\geq 0,\quad 1\leq i\leq M^0,\quad 1\leq j\leq M^1$$

$$\sum_{j=1}^{M^0}s_{ij}\leq\omega_i^0,\quad 1\leq i\leq M^0$$

$$\sum_{i=1}^{M^1}s_{ij}\leq\omega_j^0,\quad 1\leq j\leq M^1$$

$$\sum_{i=1}^{M^0}\sum_{j=1}^{M^1}s_{ij}=\min\left(\sum_{i=1}^{M^0}\omega_i^0,\sum_{j=1}^{M^1}\omega_j^1\right)$$

when ω_i^0 and ω_j^1 are probabilities, EMD is equivalent to the Mallows distance [2]. Another matching-based distance is the Integrated Region Matching (IRM) distance [2]. The IRM distance uses the most similar highest priority (MSHP) principle to match different modalities or regions. The weights s_{ij} are subject to the same constraints as in the Mallows distance, except that $\mathbf{d}(,)$ is not computed by minimization. Another way to the adjustment of weights ω_i in image similarity is relevance feedback which captures the user's precise needs through iterative feedback and query refinement. The goal of relevance feedback is to find the

appropriate weights to model the user's information need [9]. The weights are classified into *intra-* and *inter-* weights. The intra-weights represent the different contributions of the components within a single vector (i.e., region or modality), while the inter-weights represent the contributions of different vectors. Intuitively, the intra-weights are decided based on the variance of the same vector components in the relevant feedback examples, while the inter-weights are directly updated according to user's feedback in terms of the similarity based on each vector. For the comparison among these distance metrics, please refer to [2] for more details.

In the context of image being represented by a set of salient points and their corresponding local descriptors, image similarity is computed based on the Euclidean distance between each pair of salient points [6]. An alternative way is to represent each image by a bag of codewords which are obtained by unsupervised learning of local appearance [3]. A large vocabulary of codewords is built by clustering a large amount of local features, and then the distribution of these codewords is obtained by counting all the points or patches within an image. Since the image is described by a set of distributions, histogram intersection and EMD defined in (2) and (8) can be employed to compute the similarity.

Key Applications

Content-based Multimedia Information Retrieval/Multimedia Database

The computation of image similarity is the fundamental operation in content-based multimedia information retrieval systems and multimedia database. Given a query image or video, the images or videos reasonably similar to the query are returned based on the given features and distance metric.

Object Recognition

Object Recognition aims to identify an object in a database of images. Typically an object is represented by a set of overlapping regions each represented by a vector computed from the region's appearance. Recognition of a particular object proceeds by matching the descriptor vectors based on image similarity.

Medical/Satellite/Surveillance Applications

In the applications such as medical, satellite image, and surveillance, image similarity is usually used for managing the database and querying or recognizing a particular object. For example, image similarity can be employed to detect the regions with abnormal characteristics in medical diagnose. Moreover, image similarity can support content-based queries on large database of remote sensing images.

Future Directions

As pointed out in [2], the problem of image similarity is the reliance on visual similarity for judging semantic similarity. As directly applying distance metrics to image similarity cannot well model human similarity perception, automatic learning of image similarity with the help of contextual and human information has been explored. For example, when an image is conceived as a bag of instances which correspond to regions, multiple-instance learning (MIL) can be used for learning semantic similarity [1]. Another interesting problem concerns image similarity in human perception system [10]. The mathematical or computational models are needed to accurately assess perceptual similarity by resembling human's perception.

Cross-references

- ▶ Image Database
- ▶ Image Retrieval
- ▶ Similarity in Video
- ▶ Video Retrieval

Recommended Reading

1. Chen Y. and Wang J.Z. Image categorization by learning and reasoning with regions. J. Machine Learn. Res., 5:913–939, 2004.
2. Datta R., Joshi D., Li J., and Wang J.Z. Image retrieval: ideas, influences, and trends of the new age. ACM Comput. Surv., 40(65), 2008.
3. Fei-Fei L. and Perona P. A bayesian hierarchical model for learning natural scene categories. In Proc. IEEE Int. Conf. on Computer Vision and Pattern Recognition, 2005, pp. 524–531.
4. Jing F., Li M., Zhang H.-J., and Zhang B. An efficient and effective region-based image retrieval framework. IEEE Trans. Image Process., 13(5):699–709, May 2004.
5. Long F., Zhang H.-J., and Feng D.D. Fundamentals of Content-based Image Retrieval. Chapter in: Multimedia Information Retrieval and Management – Technological Fundamentals and Applications. D.D. Feng, W.C. Siu, H.-J. Zhang (ed.). Springer, Berlin Heidelberg New York, January 2003.
6. Lowe D.G. Distinctive image features from scale-invariant keypoints. Int. J. Computer Vis., 60(2):91–110, 2004.
7. Rubner Y., Tomasi C., and Guibas L.J. The earth mover's distance as a metric for image retrieval. Int. J. Computer Vis., 40 (2):99–121, 2000.

8. Rui Y., Huang T.S., and Chang S.-F. Image retrieval: current techniques, promising directions and open issues. J. Vis. Commun. Image Rep., 13(10):39–62, 1999.
9. Rui Y., Huang T.S., Ortega M., and Mehrotra S. Relevance feedback: a power tool for interactive content-based image retrieval. IEEE Trans. Circ. Video Tech., 8(5):644–655, September 1998.
10. Santini S. and Jain R. Similarity measures. IEEE Trans. Patt. Analy. Machine Intell., 21(9):871–883, September 1999.
11. Smeulders A.W.M., Worring M., Santini S., Gupta A., and Jain R. Content-based image retrieval at the end of the early years. IEEE Trans. Patt. Anal. Machine Intell., 22(12):1349–1380, 2000.
12. Wang B., Li Z., Li M., and Ma W.-Y. Large-scale duplicate detection for web image search. In Proc. IEEE Int. Conf. on Multimedia and Expo, 2006, pp. 353–356.
13. Zhou D., Bousquet O., Lal T., Weston J., and Scholkopf B. Learning with local and global consistency. In Proc. Advances in Neural Information Processing System, 2003, pp. 321–328.

Image Retrieval

► Image Database

Image Retrieval and Relevance Feedback

Michel Crucianu
National Conservatory of Arts and Crafts, Paris, France

Definition

Relevance feedback is a means of refining a query in an information retrieval system by asking the user to specify how relevant each result of the query is. An image retrieval session relying on relevance feedback is interactive and iterative. The session is divided into several consecutive rounds. At every round, the user provides feedback regarding the current retrieval results, usually by qualifying the returned images as either "relevant" or "irrelevant"; from this feedback, the system attempts to better identify the target of the user and to return improved results. A relevance feedback mechanism must maximize the relevance of the results while minimizing the amount of interaction between the user and the system.

Historical Background

In the early years of content-based image retrieval (CBIR), query by visual example (QBVE) was a prevailing paradigm. To support QBVE, an image retrieval system must first extract, during an off-line phase, a description in terms of low-level features (e.g., distribution of colors, textures, etc.) from every image in the supplied database. Then, when a query image is provided, the system returns the images whose descriptions are the most similar to the description of this query. However, QBVE encountered important difficulties: (i) there is a *semantic gap* between the high-level concept defining what a user is searching for and the existing low-level descriptions, and (ii) a relevant image is seldom available to serve as initial query ("page zero" problem). The introduction of relevance feedback in CBIR was motivated by both difficulties. User feedback provided during consecutive rounds can allow the system to progressively learn a correspondence between the target concept and low-level descriptions, thus bridging the semantic gap. Also, since the user can amend the results at every round, the initial query is less important; many different starting images may allow to identify the target concept.

Relevance feedback is not specific to image retrieval. For other types of content, such as text, music or video, a gap can also be found between automatically extracted descriptions of the content and search criteria that users consider meaningful. Actually, relevance feedback was first introduced for the retrieval of text documents in the seminal work of Salton and van Rijsbergen. These initial proposals inspired the early feedback methods put forward for image retrieval. However, since a user can evaluate the relevance of an image faster than the relevance of a text or of a streaming media item, the appeal of relevance feedback was stronger for image retrieval.

Foundations

Understanding the specific goal of an image retrieval session is important both for defining an appropriate feedback mechanism and for identifying adequate evaluation methods. The earliest and most frequent goal involving relevance feedback consists of finding images that illustrate a target concept the user has in mind [6,10]. The concept can correspond to a perceptive characteristic, but in general has higher-level semantics. The system must solve a ranking problem: the images must be ordered and returned to the user by decreasing relevance. Nevertheless it is accepted that a precise ranking of the relevant images is not required,

mainly because the user herself may be unable to define an optimal ranking; the system should simply rank most of the relevant images before the irrelevant ones.

Recently, relevance feedback was also suggested as a way to delineate the class of images illustrating a concept the user has in mind. The aim is to extend textual annotations of some images in the class to the others [7] or to help librarians perform "mass annotation" of the images [3]. To make this procedure efficient, much less effort should be required for defining the target class than for individually annotating all the images in the class. In this case, the system has to solve a classification problem: the boundary between the images belonging to the target class (the relevant images) and the others (irrelevant images) must be reliably identified. The rate of false positives (i.e., images that do not belong to the class but are assigned to it and, consequently, receive a wrong annotation) is considered to be more important than the overall error rate. However, the boundary does not have to be crisp and a degree of confidence can be associated to the resulting annotation.

Image retrieval with relevance feedback can only be effective if some important assumptions are verified. First, discrimination between relevant and irrelevant images must be possible with the available image descriptors; the "numerical gap," due to the use of inappropriate descriptors, should be avoided. Second, the target concept of the user must be consistent during the consecutive feedback rounds of a retrieval session. Third, since the amount of feedback a user can provide is very limited, the target concept that needs to be identified must have a relatively simple representation in the description space of the images.

Relevance Feedback Methods

To support relevance feedback, an image retrieval system should include two components: a *learner* and a *selector*. At every feedback round, the user marks (or labels) images returned by the system as either relevant or irrelevant. The learner makes use of this information to reestimate the target of the user (see Fig. 1). Given the current estimation of the target, the selector chooses other images for which the user is asked to provide feedback during the next round. The recent evolution of the learners and of the selection criteria is briefly presented in the following.

Learners

To estimate the target of the user, the learner can rely on the training data, consisting of the images marked by the user during consecutive feedback rounds, and on prior knowledge when it is available. The task of the learner is particularly difficult in the context of relevance feedback for several reasons (see also [16]). First, since the interaction with the user during a feedback session is limited, the amount of training data is small, sometimes much smaller than the dimension of the image description space. This highlights the importance of prior knowledge. Then, the target class may have a rather complex shape in the description space and even several disconnected modes; since training data is scarce, this can severely limit the generalization expected from the learner. Another difficulty comes from the strong imbalance in the training data: there are usually fewer positive examples (images considered relevant by the user) than negative examples (irrelevant images). The learner should have a low sensitivity to this imbalance or some remedy must be

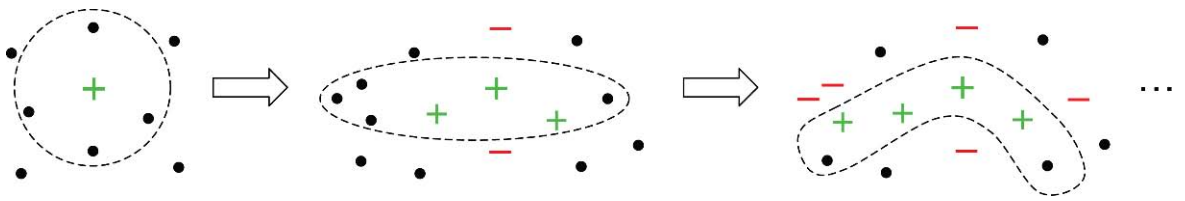


Image Retrieval and Relevance Feedback. Figure 1. Simplified view of some relevance feedback rounds. An initial similarity-based retrieval (leftmost image) returns the four nearest neighbors (points inside the dotted circle) of the query (+), relying on the default metric defined in the image description space. By marking the images returned as either relevant (+) or irrelevant (−), during consecutive rounds, the user allows the system to progressively identify the target class. At every round, the dotted boundary shows the current estimation of the target.

found. Last but not least, to preserve interactivity, both learning from the training data and the evaluation of the remaining images according to the selection criterion should be fast.

Early work on relevance feedback for CBIR directly followed from QBVE, by assuming the existence of an ideal query point and feature weighting that, if found, would allow QBVE to provide the appropriate answer to the user. This motivated the "query point movement" (QPM) approach, for which the task of the learner consists in identifying, at every round, a better query point together with a re-weighting of the individual dimensions of the image description space. Learning sometimes used only the positive examples [14], but usually employed both the positive and the negative examples [12]. These early proposals make the strong assumption that the target class corresponds to a multidimensional Gaussian distribution and some further consider that the covariance matrix of the target class is diagonal. Learning consists then in estimating the parameters of this distribution. This restrictive assumption was removed in [8,11], as a first departure from QPM. In the query expansion scheme put forward in [11], an online clustering of the examples is performed first, then all the other images are evaluated using a nearest-neighbor decision with respect to these clusters. The densities of the positive and of the negative examples are estimated with a Parzen window method in [8]; the images returned to the user are ranked according to the difference between the two densities.

For a large part of the recent work on relevance feedback, the learners employed are kernel methods and especially support vector machines (SVM [13]). With these kernel methods, the data in input space is first mapped to a higher-dimensional feature space using a non-linear transform associated to a reproducing kernel; linear methods in this feature space provide nonlinear solutions in the input space. Learning is usually based on constrained quadratic optimization and the sparsity of the solution is encouraged by various means. Two-class SVM maximize the *margin* between the discrimination boundary and the training data belonging to each class, so the resulting boundary is only defined by those examples that are closest to the boundary (the *support vectors*). One-class SVM model the support of the distribution of data in input space by finding the smallest sphere surrounding the data in feature space (according to one formulation); this sphere is only determined by the outermost examples. Since it is defined in the feature space, a kernel method can be easily applied to different types of data (e.g., vectors, sets, graphs) if an appropriate kernel is found. An SVM has some other properties that make it an interesting choice as a learner for relevance feedback. First, the decision function of an SVM allows both the definition of a boundary and of a ranking. Second, learning with few examples is very fast and, given the sparsity of the solution, computing the value of the decision function for the unmarked images can also be relatively fast. Finally, by relying only on support vectors, two-class SVM are usually less sensitive than density-based learners to the imbalance between positive and negative examples in the training data.

Among kernel methods, recent relevance feedback proposals often make use of two-class SVM to discriminate relevant and irrelevant images (see e.g., [15]). But one-class SVM were also used to model the distribution of the relevant images alone (see [1]). Other kernel methods, such as kernel biased discriminant analysis, Bayes point machines and relevance vector machines, were successfully employed in CBIR with relevance feedback.

Given the small amount of labeled data provided to the learner by user feedback, *semi-supervised* learning received significant attention in this context. Semi-supervised learning attempts to make use of both labeled and unlabeled data in order to find a better model (such as a discrimination boundary or a ranking function). It relies on the assumption that the distribution of unlabeled data does provide valuable information regarding the sought model (e.g., the density of data is lower near the boundary between classes than inside each class). When applied to relevance feedback, this approach has to face two important difficulties. First, the underlying assumption is not necessarily true for a particular image database and target class, and cannot be verified *a priori*. Second, taking unlabeled data into account frequently produces an increase in the computational complexity, which may not be compatible with interactive search.

Selection Criteria

After the learner reestimates the target of the user from the feedback received the selector must provide the user with a new set of images. The ultimate goal of a retrieval session is to present the user with as many relevant images as possible. Accordingly, the earliest

and most frequently employed selection criterion returns those unmarked images that are considered to be the most relevant given the current estimation of user's target. The short description for this criterion is that it returns the "most positive" (MP) images. It has the advantage that the user can receive quite early during the retrieval session several relevant images.

However, to reach optimal results with a minimal amount of interaction, the system should elicit from the user at every round as much information as possible regarding the distinction between relevant and irrelevant images. Existing results concerning *active* learning point out that this is achieved when the examples shown to the user are those expected to remove a maximal amount of uncertainty regarding the target. This selection criterion can be briefly described as returning the "most informative" images (MI). It translates into two complementary conditions for the images being selected: each of them should be ambiguous (given the current estimation of the target class) and any two images should be as dissimilar as possible. A selection criterion based on active learning was first proposed for image retrieval with relevance feedback in [2], for a learner using density estimation. The ambiguousness condition was considered in [15] for SVM learners: the most ambiguous images (MA) are those that are nearest to the current boundary between relevant and irrelevant images. The dissimilarity (or low redundancy) condition was added later (see e.g., [3]).

Temporal Structure of the Session

The MI selection criterion was shown to minimize the number of feedback rounds required for defining the target class of the user. But since MI returns ambiguous images, it is inappropriate for presenting the user a maximum of relevant images. A first solution to this problem was to use MI during several rounds in order to define the class and then switch to MP in order to show the most relevant images; it is nevertheless difficult to know when exactly to replace MI by MP. Another solution is to combine the results of both MI and MP in the images shown to the user at every feedback round; this alternative is better adapted to nonprofessional users, who expect more immediate reward.

The initialization of search can also have an impact on the temporal structure of a retrieval session. In some cases, a relevant starting image can be provided by the user (external query) or found with the help of a visual summary of the image database. If an appropriate starting image is not available, then feedback should be used both for finding a truly relevant image (exploration stage) and then for retrieving further relevant images (exploitation stage). The session begins with some random selection of images. During the exploration stage, the system must be able so show the user images that are increasingly relevant. The user may not see any truly relevant image during several rounds, so she is expected to indicate which among the images returned by the system are more relevant than the others. Support for such an exploration stage also helps when the target class has several distinct modes. While the exploratory behavior was addressed by [2] and, to some extent, by a few QPM proposals, later work mainly focused on the exploitation stage. A competitive method for the exploration stage was recently proposed in [4].

Evaluation of Relevance Feedback in Image Retrieval

User satisfaction is the ultimate measure of the success of image retrieval with relevance feedback. Reliable evaluations or comparisons between alternative relevance feedback methods require large groups of users and real world image retrieval problems. Given the difficulty of setting up large scale experiments with real users, most evaluations are actually performed on specific ground truth databases, by emulating the user. A ground truth usually corresponds to the definition of a set of mutually exclusive image classes, covering an entire database. The emulated user knows the ground truth and is assumed to have a stoic and error free behavior: at every feedback round the emulated user correctly marks as either relevant or irrelevant each of the images returned by the selector. To obtain a reliable evaluation, it is very important to employ several ground truth databases having dissimilar characteristics. It was also argued that more diverse and realistic behaviors should be assigned to the emulated user.

When the aim is to rank the relevant images before the irrelevant ones, the quality of retrieval is usually given by the proportion of relevant images in the top N returned by an MP selector; N is the number of images in the target class of the ground truth database. When the aim is to delineate the target class, the quality of discrimination is $1 - \varepsilon$, where ε is either the overall classification error or the rate of false positives. Usually, a "right" number of feedback rounds can not be fixed *a priori*. To evaluate the performance of a relevance feedback

method, the appropriate quality measure should be recorded during several consecutive feedback rounds.

Key Applications

Interactive multimedia search engines are the original motivation for content-based image retrieval in general and for the introduction of relevance feedback in particular. Such search engines are of high interest both for the general public and for professional users. Scalability and user-friendliness are fundamental requirements in this context.

Assistance in the annotation of content can be an important application mainly directed to professional users. Relevance feedback can support such users in delineating large classes of images in order to annotate at once all the images belonging to a class (mass annotation). In this case, the ability to reach a low classification error or a low rate of false positives is the major concern.

Future Directions

Since the amount of feedback provided during a retrieval session is very small, the system should make the most of all the information sources potentially available. These sources concern the users (e.g., past sessions of other users, user profiles), the images (e.g., structured or unstructured metadata) and the context of retrieval. The system should be able to evaluate and integrate all these sources with the feedback directly provided by the current user.

An important issue that was not extensively studied is the scalability of relevance feedback to very large databases. Scalability is a challenge both for the learner (especially when semi-supervised methods are employed) and for the selector. To avoid evaluating the decision function for all the unmarked images in the database, an index structure is needed. But existing multidimensional or metric index structures and associated *k*NN retrieval methods can not be directly applied, mainly because the queries that have to be processed are not classical point queries. A good example is the use of the MA (or MI) selection criterion with an SVM learner: in this case the images that are nearest to the discrimination boundary (defined by a hyperplane in feature space) should be returned. This scalability issue is addressed by some recent proposals (see [5,9]).

The interaction between users and current systems is rather limited; typically, a user can only mark as relevant or irrelevant each of the shown images. A more advanced interface should bring in more flexibility, by allowing every user to group together or to separate images, to place them in a visual summary, to mix several image retrieval paradigms, to interact online with other users, etc.

Cross-references

► Feature Extraction for Content-Based Image Retrieval
► Image Retrieval
► Relevance Feedback

Recommended Reading

1. Chen Y., Zhou X.S., and Huang T.S. One-class SVM for learning in image retrieval. In Proc. Int. Conf. Image Processing, 2001, pp. 34–37.
2. Cox I.J., Miller M.L., Omohundro S.M., and Yianilos P.N. An optimized interaction strategy for Bayesian relevance feedback. In Proc. IEEE Int. Conf. on Computer Vision and Pattern Recognition, 1998, pp. 553–558.
3. Ferecatu M., Crucianu M., and Boujemaa N. Retrieval of difficult image classes using SVM-based relevance feedback. In Proc. 6th ACM SIGMM International Workshop on Multimedia Information Retrieval, 2004, pp. 23–30.
4. Ferecatu M. and Geman D. Interactive search for image categories by mental matching. In Proc. 11th IEEE Conf. Computer Vision, 2007, pp. 1–8.
5. Heisterkamp D.R. and Peng J. Kernel VA-files for relevance feedback retrieval. In Proc. 1st ACM Int. Workshop on Multimedia Databases, 2003, pp. 48–54.
6. Kurita T. and Kato T. Learning of personal visual impression for image database systems. In Proc. 2nd Int. Conf. Document Analysis and Recognition, 1993, pp. 547–552.
7. Lu Y., Hu C., Zhu X., Zhang H.-J., and Yang Q. A unified framework for semantics and feature based relevance feedback in image retrieval systems. In Proc. 8th ACM Int. Conf. on Multimedia, 2000, pp. 31–37.
8. Meilhac C. and Nastar C. Relevance feedback and category search in image databases. In Proc. Int. Conf. on Multimedia Computing and Systems, 1999, pp. 512–517.
9. Panda N., Goh K.-S., and Chang E.Y. Active learning in very large databases. Multimedia Tools Applicat., 31(3):249–267, 2006.
10. Picard R.W., Minka T.P., and Szummer M. Modeling user subjectivity in image libraries. In Proc. Int. Conf. Image Processing, 1996, pp. 777–780.
11. Porkaew K. and Chakrabarti K. Query refinement for multimedia similarity retrieval in MARS. In Proc. 7th ACM Int. Conf. on Multimedia (Part 1), 1999, pp. 235–238.
12. Rui Y., Huang T.S., Ortega M., and Mehrotra S. Relevance feedback: a power tool in interactive content-based image retrieval. IEEE Trans. Circuits Syst. Video Tech., 8(5):644–655, 1998.
13. Schölkopf B. and Smola A. Learning with Kernels. MIT, Cambridge, MA, 2002.

14. Sclaroff S., Taycher L., and Cascia M.L. Image Rover: a content-based image browser for the world wide web. In Proc. Workshop on Content-Based Access of Image and Video Libraries, 1997, pp. 2–9.
15. Tong S. and Chang E. 1Support vector machine active learning for image retrieval. In Proc. 9th ACM Int. Conf. on Multimedia, 2001, pp. 107–118.
16. Zhou X.S. and Huang T.S. Relevance feedback for image retrieval: a comprehensive review. Multimedia Syst., 8(6):536–544, 2003.

Image Retrieval System

► Image Database

Image Segmentation

Frank Y. Shih
New Jersey Institute of Technology, Newark, NJ, USA

Synonyms

Region segmentation; Pixel classification; Edge detection; Thresholding

Definition

The rapid rate of image analysis field has grown enormously in the past few decades. Image analysis intends to construct explicit, meaningful descriptions of physical objects in images. It can be divided into two parts: low-level image analysis and high-level image analysis. Low-level tasks focus on region-based segmentation, whereas high-level tasks are related to object-oriented representation. Image segmentation, a process of pixel classification, aims to extract or segment objects or regions from the background. Intrinsic images can be generated at the low-level processing, revealing physical properties of the imaged scene. This can often be implemented with parallel computation.

Historical Background

Image segmentation is a critical step to the success of object recognition [12], image compression [2], image visualization [7], and image retrieval [3]. Pal and Pal [13] provided a review on various segmentation techniques. It should be noted that there is no single standard approach to segmentation. Many different types of scene parts can serve as the segments on which descriptions are based, and there are many different ways in which one can attempt to extract these parts from the image. Selection of an appropriate segmentation technique depends on the type of images and applications.

The level of segmentation or subdivision relies on the problem domain being dealt with. For example, in the optical character recognition (OCR), the text is separated from the document image, and further partitioned into columns, lines, words, and connected components. In building character subimages, one is often confronted with touching or broken characters that occur in degraded documents (such as fax, scan, photocopy, etc.). It is still challenging to develop techniques for properly segmenting words into their characters.

There are primarily four types of segmentation techniques: thresholding, boundary-based, region-based, and hybrid techniques. Thresholding is based on the assumption that clusters in the histogram correspond to either background or objects of interest that can be extracted by separating these histogram clusters. In addition to thresholding, many image segmentation algorithms are based on two basic properties of the pixel intensities in relation to their local neighborhood: discontinuity and similarity. Methods based on pixel discontinuity are called boundary-based or edge extraction methods, whereas methods based on pixel similarity are called region-based methods. Boundary-based methods assume that the pixel properties, such as intensity, color, and texture, should change abruptly between different regions. Region-based methods assume that neighboring pixels within the same region should have similar values (e.g., intensity, color, texture).

It is well known that such segmentation techniques – based on boundary or region information alone – often fail to produce accurate segmentation results. Hence, there has been a tendency towards hybrid segmentation algorithms which take advantage of the complementary nature of such information. Hybrid methods combine boundary detection and region growing together to achieve better segmentation [5,6,14]. Note that both results should achieve the foreground and background segmentation coherently.

Foundations

A number of image segmentation techniques, including thresholding, component labeling, locating object

contours by the snake model, and automatic seeded region growing, are described below.

Thresholding

Thresholding provides an easy and convenient way to perform image segmentation based on the different intensities or colors in the foreground and background regions of an image. Not all images can be segmented successfully into foreground and background using simple thresholding. Its validity relies on the distribution of the intensity histogram. If the intensity distribution of foreground objects is quite distinct from the intensity distribution of background, it will be clear to apply thresholding for image segmentation. In this case, one expects to see distinct peaks in the histogram corresponding to foreground objects, such that threshold values can be picked to isolate those peaks accordingly. If such a peak does not exist, it is unlikely that simple thresholding can achieve a good segmentation.

There are several methods of choosing the threshold value λ. For example, the universal thresholding by Donoho et al. [4] sets

$$\lambda = \frac{\sigma\sqrt{2\log n}}{\sqrt{n}} \tag{1}$$

where σ is the standard deviation of the wavelet coefficients and n is the total size of samples. Other possibility is quantile thresholding where λ is statistically set to replace a percentage of the coefficients with the smallest magnitude to zero. Another adaptive method of automatically choosing the best threshold λ consists of four steps: (i) Choose an initial estimate λ; (ii) Calculate the two mean values μ_1 and μ_2 within the two groups of pixels after thresholding at λ; (iii) Calculate the new threshold value $\lambda = (1/2)(\mu_1 + \mu_2)$; (iv) If the new threshold value has a little change (i.e., smaller than a predefined constant), then the threshold selection is done; otherwise, go back to step 2.

Object (Component) Labeling

It is very possible to have more than one object in a scene. All the objects must be individually extracted for the purpose of establishing the object model base. The object-labeling technique is used, so that the array representation of these objects is a multivalued picture, in which the points of each component all have a unique nonzero label and the points of background are all zeros. This technique only requires two raster scans. To label 4-connected components, only the upper and left neighbors are checked. If the 8-connectedness is used, the upper two diagonal neighbors are also included.

Let the value of object points be "1" and of background points be "0." Assume that the 8-connectedness is adopted. Therefore, if all the above four neighbors (i.e., previously scanned neighbors) of the point P are zeros, then P is assigned a new label. If one of four neighbors is 1, then P gets the same label as that neighbor. If two or more of them are 1's, then P gets any one of their labels, and the equivalence table is established by marking the different labels together for later adjustment. Equivalence processing consists in merging the equivalent pair into the same class; i.e., a unique label is assigned to each class. Finally, a second scan is performed to replace each label by the representative of its class. Each component has now been uniquely labeled. After these processes, each individual model object can be fetched by its label.

Locating Object Contours by the Snake Model

In the original snake formulation of Kass et al. [9], the best snake position was defined as the solution of a variational problem requiring the minimization of the sum of internal and external energies integrated along the length of the snake. The corresponding Euler equations, which give the necessary conditions for this minimizer, comprise a force balance equation. The snake model has provided a number of applications in object segmentation, stereo matching, motion tracking, etc. In image processing, the snake model defines a snake as an energy-minimizing spline guided by external constraint forces and influenced by such forces that pull it toward image features such as lines and edges. It is a kind of the active contour model in the way that it locks on nearby edges, localizing them accurately.

There are two parts in the energy function of the snake model. The first part reflects geometric properties of the contour, and the second part utilizes the external force field to drive the snake. The first part serves to impose a piecewise smoothness constraint, and the second part is responsible for putting the snake near the local minimum of energy. The traditional snakes suffer from a great disadvantage that when an object resides in a complex background, the strong edges may not be the object edges of interest.

Therefore, researchers have proposed statistical and variational methods to enrich the energy function and extend its flexibility. They calculated the difference between the target object and the background using statistical analysis, but these limitations come from the priori knowledge requirements, such as independent probability models and template models. Unfortunately, such priori knowledge is usually unavailable unless the captured images are under very constrained settings. A problem with the snake model is that a user needs to place the initial snake points sufficiently close to the feature of interest.

The Traditional Snake Model A snake is a controlled continuity spline that moves and localizes onto a specified contour under the influence of the objective function. Let a snake be a parametric curve: $v(s) = [x(s), y(s)]$, where parameter $s \in [0, 1]$. It moves around the image spatial domain to minimize the objective energy function as defined by

$$E_{snake}(v) = \sum_{i=1}^{n} [\alpha \times E_{cont}(v_i) + \beta \times E_{curv}(v_i) + \gamma \times E_{image}(v_i)], \quad (2)$$

where α, β and γ are weighting coefficients that control the snake's tension, rigidity, and attraction, respectively. The first and second terms are correspondingly the first- and second-order continuity constraints. The third term measures the edge strength (i.e., the image force).

The continuity force E_{cont} encouraging even spacing of points can be calculated as

$$E_{cont}[v_i] = \frac{|\overline{d} - |v_i - v_{i-1}||}{\max_j\{|\overline{d} - |v_i(j) - v_{i-1}||\}}, \quad (3)$$

where $\{v_i(j)|j = 1, 2, ..., m\}$ denotes the snake point v_i's m neighbors, and $\overline{d}$ denotes the average length of all the pairs of adjacent points on the snake contour as given by

$$\overline{d} = \frac{\sum_{i=1}^{n} |v_i - v_{i-1}|}{n}, \quad (4)$$

where $v_0 = v_n$. This term tends to keep the distances between each pair of adjacent vertices equal.

The energy of the second-order continuity E_{curv} is represented by

$$E_{curv}[v_i] = \frac{|v_{i-1} - 2v_i + v_{i+1}|}{\max\{|v_{i-1} - 2v_i + v_{i+1}|\}}. \quad (5)$$

The numerator can be rearranged as

$$v_{i-1} - 2v_i + v_{i+1} = (v_{i+1} - v_i) - (v_i - v_{i-1}). \quad (6)$$

If the ith vertex is pushed toward the midpoint of two adjacent vertices, E_{image} is minimized; i.e., the shape of the contour will remain $\subset^2$ continuity.

The third term, image energy E_{image}, is derived from the image so that it takes on its smaller values at the features of interest, such as boundaries. It considers the gradient (denoted as *grad*) magnitude, leading the active contour toward step edges. It is normalized to measure the relative magnitude as

$$E_{image}[v_i] = \frac{\min\{|grad|\} - |grad_{v_i}|}{\max\{|grad|\} - \min\{|grad|\}}, \quad (7)$$

where min and max denote the minimum and maximum gradients in the v_i's local m-neighborhood, respectively. Note that because the numerator in eq. (7) is always negative, E_{image} can be minimized for locating the largest gradient, which is the edge. In general, the traditional snake model can locate object contours in a simple background. If the background becomes complex, it will fail since the complex background will generate noisy edges to compete with the object edges for attracting the snake.

The Improved Snake Model The following two issues are observed when the snake model fails to locate object contours in complex backgrounds. One is the gray-level sensitivity; i.e., the more abrupt change the gray levels have (e.g., noises), the larger impact on the energy function the snake makes. The other is that the snake mistakenly locates the edges belonging to the background details due to their closeness to the snake point. Figure 1 shows a disk object in a complex background. If the snake points are initialized outside the disk, the snake cannot locate the disk contour accurately due to the disturbances from the background grids. The idea is to push the mean intensity of the polygon enclosed by the snake contour to be as close as to the mean intensity of the target object. The smaller intensity difference between the polygon and the object, the closer the snake approaches the object contour. Therefore, a new energy term, called the *Regional Similarity Energy* (RSE), is established for calculating the gray-level differences to be added into the overall energy [16].

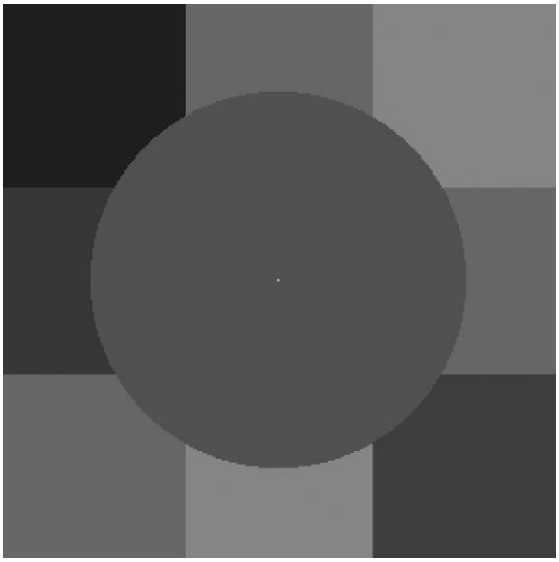


Image Segmentation. Figure 1. A disk object resides in a complex background.

The Gravitation External Force Field and the Greedy Algorithm In this section, the gravitation external force field is introduced and the greedy algorithm [8] is used for the active contour. The concept of gravitation external force field is taken from physics. Two objects attract each other by a force, which is proportional to their mass product and inversely proportional to the distance between their mass centers. Based on this concept, an external energy field, called the *Gravitation Energy Field* (GEF), is developed as given by

$$E_{gravitation} = \int \frac{g(\vec{r})}{\|\vec{r}\|}\vec{r}d\vec{r}, \tag{8}$$

where $\vec{r}$ is a position vector and $g(\vec{r})$ is a first-order derivative. Note that the edge pixels have the local maxima in the first-order derivative. The attractive force enables the snake points to move toward the object. With the gravitation energy field, the active contour can be dragged toward the object even if the snake points are far away. Therefore, the total energy function becomes

$$\begin{aligned} E_{snake} = {} & \alpha E_{cont}[v(s)] + \beta E_{curv}[v(s)] + \gamma E_{image}[v(s)] \\ & + \mu E_{gravitation}[v(s)] + \delta E_{RSE}[v(s)]. \end{aligned} \tag{9}$$

These weighting coefficients can be adjusted according to the user's application. For simplicity, $\alpha = \beta = \gamma = \mu = \delta = 1$.

Experimental Results Both the improved and the traditional snake models are applied on the added salt-and-pepper noise of Fig. 1. The results are shown in Fig. 2. It is observed that the improved model can locate the disk contour, but the traditional model fails. The improved model is suitable for random noise or fixed pattern noise. The banding noise is highly camera-dependent, and is the one which is introduced by the camera when it reads data from the digital sensor. The improved model may not perform well in an image with such noise.

Automatic Seeded Region Growing

Seeded Region Growing (SRG) is one of hybrid methods proposed by Adams and Bischof [1]. It starts with assigned seeds, and grow regions by merging a pixel into its nearest neighboring seed region. Mehnert and Jackway [10] pointed out that SRG has two inherent pixel order dependencies that cause different resulting segments. The first order dependency occurs whenever several pixels have the same difference measure to their neighboring regions. The second order dependency occurs when one pixel has the same difference measure to several regions. They used parallel processing and re-examination to eliminate the order dependencies. Fan et al. [5] presented an automatic color image segmentation algorithm by integrating color-edge extraction and seeded region growing on the *YUV* color space. Edges in *Y*, *U*, and *V* are detected by an isotropic edge detector, and the three components are combined to obtain edges. The centroids between adjacent edge regions are taken as the initial seeds. The disadvantage is that their seeds are over-generated.

Overview of the Improved Seeded Region Growing Algorithm

Figure 3 presents the overview of the improved seeded region growing algorithm [15]. Firstly, the color image is converted from *RGB* to YC_bC_r color space. Secondly, automatic seed selection is applied to obtain initial seeds. Thirdly, the seeded region growing algorithm is used to segment the image into regions, where a region corresponds to a seed. Fourthly, the region-merging algorithm is applied to merge similar regions, and small regions are merged into their nearest neighboring regions.

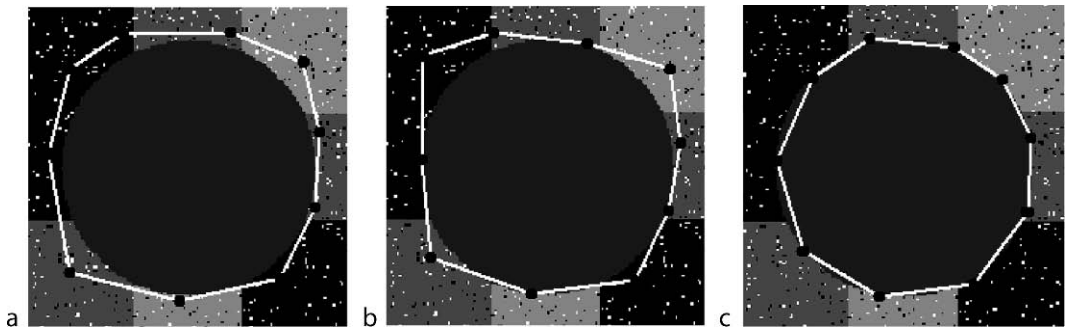


Image Segmentation. Figure 2. (a) The initial snake points, (b) the result by the traditional snake model, (c) the result by the improved snake model.

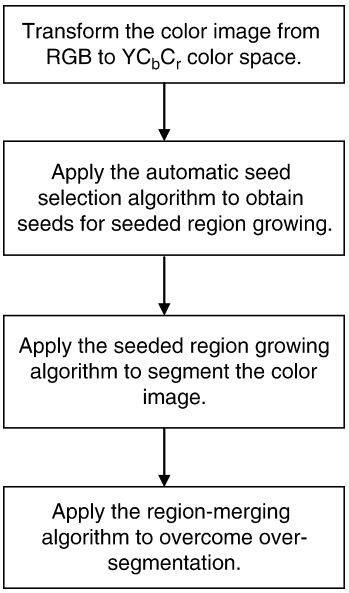


Image Segmentation. Figure 3. Outline of the proposed algorithm.

The Method for Automatic Seed Selection

For automatic seed selection, the following three criteria must be satisfied. First, the seed pixel must have high similarity to its neighbors. Second, for an expected region, at least one seed must be generated in order to produce this region. Third, seeds for different regions must be disconnected.

The similarity of a pixel to its neighbors can be computed as follows. Considering a 3×3 neighborhood, the standard deviations of Y, C_b and C_r components are calculated using

$$\sigma_x = \sqrt{\frac{1}{9}\sum_{i=1}^{9}(x_i - \bar{x})^2}, \tag{10}$$

where x can be Y, C_b, or C_r, and the mean value $\bar{x} = \frac{1}{9}\sum_{i=1}^{9} x_i$. The total standard deviation is

$$\sigma = \sigma_Y + \sigma_{C_b} + \sigma_{C_r}. \tag{11}$$

The standard deviation is normalized to [0,1] by

$$\sigma_N = \sigma/\sigma_{\max}, \tag{12}$$

where $\sigma_{\max}$ is the maximum of the standard deviation in the image. The similarity of a pixel to its neighbors is defined as

$$H = 1 - \sigma_N. \tag{13}$$

From the similarity, the first condition for the seed pixel candidate is defined as follows:

Condition 1: A seed pixel must have the similarity higher than a threshold value.

Secondly, the relative Euclidean distances (in terms of YC_bC_r) of a pixel to its 8 neighbors is calculated as

$$d_i = \frac{\sqrt{(Y - Y_i)^2 + (C_b - C_{b_i})^2 + (C_r - C_{r_i})^2}}{\sqrt{Y^2 + C_b^2 + C_r^2}} \quad i = 1, 2, \cdots, 8. \tag{14}$$

From experiments, the performance of using relative Euclidean distance is better than using normal Euclidean distance. For each pixel, the maximum distance to its neighbors is calculated as

$$d_{\max} = \max_{i=1}^{8}(d_i). \tag{15}$$

From the maximum distance, the second condition for the seed pixel candidate is defined below.

Condition 2: A seed pixel must have the maximum relative Euclidean distance to its eight neighbors which is less than a threshold value.

A pixel is classified as a seed pixel if it satisfies the above two conditions. In order to choose the threshold automatically in condition 1, Otsu's method [11] is used. The threshold is determined by choosing the value that maximizes the discrimination criterion σ_B^2/σ_w^2, where σ_B^2 is the between-class variance and σ_W^2 is the within-class variance. In condition 2, the value 0.05 is selected as the threshold based on these experiments.

Each connected component of seed pixels is taken as one seed. Therefore, the seeds generated can be one pixel or one region with several pixels. Condition 1 checks whether the seed pixel has high similarity to its neighbors. Condition 2 makes sure that the seed pixel is not on the boundary of two regions. It is possible that for one desired region, several seeds are detected to split it into several regions. The over-segmented regions can be merged later in the region-merging step. Figure 4(a) shows a color image, and (B) shows the detected seeds marked in red color. Note that the connected seed pixels are considered as one seed.

The Segmentation Algorithm

Let $A_1, A_2, \ldots, A_i$ denote initial seeds and S_i denote the region corresponding to A_i. The mean of all seed pixels in S_i in terms of Y, C_b and C_r components is denoted as $(\bar{Y}, \bar{C}_b, \bar{C}_r)$. The segmentation algorithm is described as follows:

1. Perform automatic seed selection.
2. Assign a label to each seed region.
3. Record neighbors of all regions in a sorted list T in a decreasing order of distances.
4. While T is not empty, remove the first point p and check its 4-neighbors. If all labeled neighbors of p have a same label, set p to this label. If the labeled neighbors of p have different labels, calculate the distances between p and all neighboring regions and classify p to the nearest region. Then, update the mean of this region and add 4-neighbors of p, which are neither classified yet nor in T, to T in a decreasing order of distances.
5. Perform region merging.

Note that in step 3, T denotes the set of pixels that are unclassified and are neighbors of at least one of the

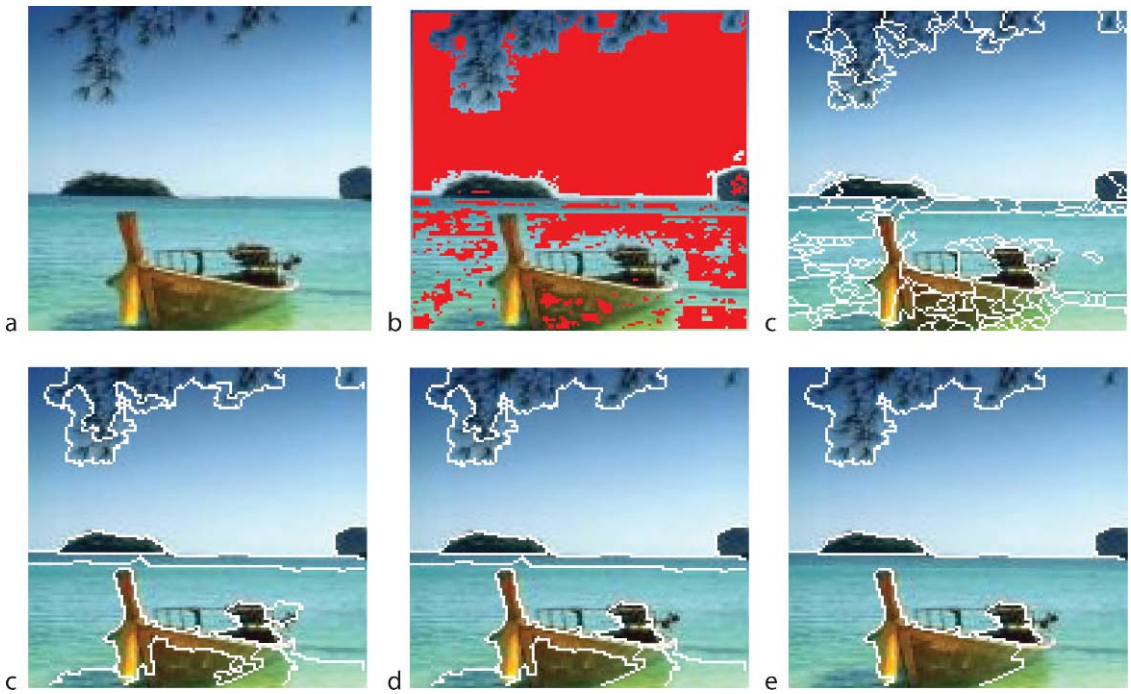


Image Segmentation. Figure 4. (a) Original color image, (b) the detected seeds are shown in red color, (c) seeded region growing result, (d) the result of merging adjacent regions with relative Euclidean distance less than 0.1, (e) the result of merging small regions with size less than 1/150 of the image, and (f) final segmented result.

region. The relative Euclidean distance d_i between the pixel i and its adjacent region is calculate by

$$d_i = \frac{\sqrt{(Y_i - \overline{Y})^2 + (C_{b_i} - \overline{C}_b)^2 + (C_{r_i} - \overline{C}_r)^2}}{\sqrt{Y_i^2 + C_{b_i}^2 + C_{r_i}^2}}, \quad (16)$$

where $(\bar{Y}, \bar{C}_b, \bar{C}_r)$ are the mean values of Y, C_b, and C_r components in that region. In step 4, the pixel p with the minimum distance value is extracted. If several pixels have the same minimum value, the pixel corresponding to the neighboring region having the largest size is chosen. If p has the same distance to several neighboring regions, it is classified to the largest region. Figure 4(c) shows the result of the proposed algorithm, where boundaries of regions are marked in white color.

Key Applications

Object Recognition, Image Compression, Image Visualization, and Image Retrieval.

Cross-references

- ▶ Computer Vision
- ▶ Image Segmentation
- ▶ Pattern Recognition
- ▶ Scene Analysis

Recommended Reading

1. Adams R. and Bischof L. Seeded region growing. IEEE Trans. Pattern Anal. Mach. Intell., 16(6):641–647, 1994.
2. Belloulata K. and Konrad J. Fractal image compression with region-based functionality. IEEE Trans. Image Process., 11 (4):351–362, 2002.
3. Chen Y. and Wang J.Z. A region-based fuzzy feature matching approach to content-based image retrieval. IEEE Trans. Pattern Anal. Mach. Intell., 24(9):1252–1267, 2002.
4. Donoho D., Johnstone I., Kerkyacharian G., and Picard D. Density estimation by wavelet thresholding. Ann. Statist., 24:508–539, 1996.
5. Fan J., Yau D.K., Elmagarmid A.K., and Aref W.G. Automatic image segmentation by integrating color-edge extraction and seeded region growing. IEEE Trans. Image Process., 10(10): 1454–1466, 2001.
6. Haris K., Efstratiadis S.N., Maglaveras N., and Katsaggelos A.K. Hybrid image segmentation using watersheds and fast region merging. IEEE Trans. Image Process., 7(12):1684–1699, 1998.
7. Hartmann S.L. and Galloway R.L. Depth-buffer targeting for spatially accurate 3-D visualization of medical images. IEEE Trans. Med. Imaging 19(10):1024–1031, 2000.
8. Ji L. and Yan H. Attractable snakes based on the greedy algorithm for contour extraction. Pattern Recognit., 35(4):791–806, 2002.
9. Kass M., Witkin A., and Terzopoulos D. Snakes: active contour models. Int. J. Comput. Vis., 1(4):321–331, 1987.
10. Mehnert A. and Jackway P. An improved seeded region growing algorithm. Pattern Recognit. Lett., 18(10):1065–1071, 1997.
11. Otsu N. A threshold selection method from gray-level histogram. IEEE Trans. Syst., Man, Cybern., 9(1):62–66, 1979.
12. Pachowicz P.W. Semi-autonomous evolution of object models for adaptive object recognition. IEEE Trans. Syst. Man Cybern., 24(8):1191–1207, 1994.
13. Pal N.R. and Pal S.K. A review on image segmentation techniques. Pattern Recognit., 26(9):1277–1294, 1993.
14. Pavlidis T. and Liow Y.T. Integrating region growing and edge detection. IEEE Trans. Pattern Anal. Mach. Intell., 12(3): 225–233, 1990.
15. Shih F.Y. and Cheng S. Automatic seeded region growing for color image segmentation. Image Vis. Comput., 23(10): 877–886, 2005.
16. Shih F.Y. and Zhang K. Efficient contour detection based on improved snake model. Pattern Recognit. Artif. Intell., 18(2): 197–209, 2004.

I

Image Standards

▶ Image Representation

Image/Video/Music Search

▶ Semantic Modeling and Knowledge Representation for Multimedia Data

Immersive Data Mining

▶ Visual Data Mining

Implementation Abstraction

▶ Visual Data Mining

Implication of Constraints

Wenfei Fan
University of Edinburgh, Edinburgh, UK

Definition

The implication problem is to decide whether or not a given set of constraints logically implies another constraint. With any constraint (dependency) language $\mathcal{L}$ there are two implication problems associated, which do not coincide in general.

In the traditional logic framework, an instance of a schema **R** is a logical structure that is either finite or infinite, referred to as an *unrestricted instance* of **R**. A set Σ of constraints over **R** *implies without restriction* a constraint φ, denoted by $\Sigma \models_{\text{unr}} \varphi$, if for each unrestricted instance **I** of **R** that satisfies Σ, **I** also satisfies φ. The *unrestricted implication problem* for $\mathcal{L}$ is to determine, given a set Σ of constraints in $\mathcal{L}$ and another constraint φ in $\mathcal{L}$, whether or not $\Sigma \models_{\text{unr}} \varphi$.

In the context of databases, only finite instances are considered and implication analysis lies within finite model theory. A set Σ of constraints over **R** *finitely implies* a constraint φ, denoted by $\Sigma \models_{\text{fin}} \varphi$, if for each finite instance **I** of **R** that satisfies Σ, **I** also satisfies φ. The *finite implication problem* for $\mathcal{L}$ is to determine, given any set Σ of constraints in $\mathcal{L}$ and another constraint φ in $\mathcal{L}$, whether or not $\Sigma \models_{\text{fin}} \varphi$.

In the context of semi-structured data or XML, implication analysis is also conducted in the absence of schema. For XML, a set Σ of constraints *implies* another constraint φ *in the absence of schema* if all XML documents that satisfy Σ also satisfy φ; similarly for semi-structured data.

Historical Background

Logical implication is one of the key issues in dependency theory. It has been studied for, and has proved valuable in, schema normalization, data integrity maintenance and query optimization, among other things. Recently there has also been renewed interest in this line of work, for XML data management, data exchange and data cleaning.

After functional dependencies were introduced by Codd in 1972, the relevance of implication to database theory was first observed by [8]. Since then the problem of logical implication has been extensively studied for a large variety of dependencies, mostly focusing on the following aspects. (i) The complexity of implication analyses (*e.g.,* [3,6,7,9,10,11,15]). In particular, the separation of finite and unrestricted implication was originally investigated in [7,10], and the chase, a powerful decision procedure for implication, was based on the idea of [3] and articulated in [7,15]. (ii) Axiomatization for characterizing logical implication, first studied by Armstrong [5] for functional dependencies, followed by a flurry of research (see, *e.g.,* [12] for a survey). (iii) View dependencies for propagating dependencies from databases to their views, originally studied in [14]. (iv) The impact of interaction between schema (types) and constraints on implication analysis, first studied by [2] for semi-structured data, and then for XML in the presence and in the absence of schema [13]. See [1,12] for comprehensive surveys.

Foundations

The remainder of this entry discusses various aspects of implication: dependencies, unrestricted and finite implication, implication and satisfiability analyses, finite axiomatization, complexity bounds, and view dependencies.

Constraints (Data Dependencies)

Implication analysis has been studied for a variety of integrity constraints (in this entry, integrity constraints and data dependencies are used interchangeably). Most constraints studied for relational databases can be expressed as first-order logic sentences of the following form, referred to as *embedded dependencies*:

$$\forall x_1 \ldots x_m \; (\phi(x_1,\ldots,x_m) \rightarrow \exists y_1 \ldots y_n \; \psi(z_1,\ldots,z_k)),$$

where (i) $\{y_1,\ldots,y_n\} = \{z_1,\ldots,z_k\} - \{x_1,\ldots,x_m\}$; (ii) ϕ is a conjunction of (at least one) relation atoms of the form $R(w_1,\ldots,w_l)$, using all of the variables in $\{x_1,\ldots,x_m\}$, where w_i is a variable for each $i \in [1,l]$; (iii) ψ is a conjunction of either relation atoms or equality atoms $w = w'$, using all of the variables in $\{z_1,\ldots,z_k\}$, where w, w' are variables; and (iv) there exist no equality atoms in ψ using existentially quantified variables.

Embedded dependencies are often classified as follows.

1. Full dependencies are embedded dependencies that have no existential quantifiers, *i.e.,* dependencies of the form $\forall x_1 \ldots x_m \; (\varphi(x_1,\ldots,x_m) \rightarrow \psi(z_1,\ldots,z_k))$, where $\{z_1,\ldots,z_k\}$ is a subset of $\{x_1,\ldots,x_m\}$. Full

dependencies include functional dependencies (FDs), multivalued dependencies (MVDs) and join dependencies (JDs) (see [1,12] for the definitions of FDs, MVDs, JDs and inclusion dependencies).

2. Tuple generating dependencies (TGDs) are embedded dependencies in which the right-hand side ψ is a relation atom. A TGD says that if a certain pattern of entries appears then another pattern must appear. Inclusion dependencies (INDs), MVDs and JDs are examples of TGDs.
3. Equality generating dependencies (EGDs) are embedded dependencies in which the right-hand side ψ is an equality atom. An EGD says that if a certain pattern of entries appears then a certain equality must hold. The best known equality generating dependencies are FDs.
4. Typed dependencies are embedded dependencies for which there exists an assignment of variables to column positions such that (i) variables in relation atoms occur only in their assigned position, and (ii) each equality atom involves a pair of variables assigned to the same position. Typed dependencies include FDs, MVDs and JDs.

Implication analysis has also been studied for integrity constraints on semi-structured data and XML. This entry focuses on relational dependencies only, and refer the interested reader to [4] for a survey on XML constraints.

Unrestricted Implication Versus Finite Implication

Given a set Σ of constraints and another constraint φ, if $\Sigma \models_{\text{unr}} \varphi$, *i.e.,* Σ implies φ without restriction, then obviously $\Sigma \models_{\text{fin}} \varphi$, *i.e.,* Σ finitely implies φ. However, the converse does not necessarily hold. It is possible that $\Sigma \models_{\text{fin}} \varphi$ whereas $\Sigma \not\models_{\text{unr}} \varphi$. That is, the implication problem and finite implication problem may have to be treated separately as different decision problems. Below are two examples.

1. When Σ is a set of FDs and INDs, and φ is either an FD or an IND [9].

   Consider a binary relation $R(A,B)$ with attributes A,B. Let $\Sigma = \{A \rightarrow B, R[A] \subseteq R[B]\}$, and φ be $R[B] \subseteq R[A]$. Then from Σ it follows that for any finite instance I of R that satisfies Σ, $|\pi_{\text{A}}(\text{I})| \geq |\pi_{\text{B}}(\text{I})|$ (by the FD in Σ) and $|\pi_A(\text{I})| \leq |\pi_B(\text{I})|$ (by the IND in Σ), where π is the projection operator in the relational algebra, and $|S|$ denotes the number of distinct tuples in a relation S. Therefore, $|\pi_A(\text{I})| = |\pi_{\text{B}}(\text{I})|$. Since I is finite and $\pi_{\text{A}}(I) \subseteq \pi_B(\text{I})$, it follows that $\pi_B(\text{I}) \subseteq \pi_A(\text{I})$ and I satisfies φ. Conversely, an infinite instance $\{(i+1,i) | i \geq 0\}$ of R satisfies Σ but does not satisfy φ; thus $\Sigma \not\models_{\text{unr}} \varphi$.

   Similarly, let φ be $R[B] \rightarrow R[A]$ then one can verify that $\Sigma \models_{\text{fin}} \varphi$ but $\Sigma \not\models_{\text{unr}} \varphi$.
2. When Σ is a set of TGDs and φ is a TGD (see [7] for a proof).

   A useful technique for proving the decidability of the implication problem for a constraint language $\mathcal{L}$ is to show that the implication and finite implication problems coincide for $\mathcal{L}$, *i.e.,* for any set Σ in $\mathcal{L}$ and another constraint φ in $\mathcal{L}$, $\Sigma \models_{\text{unr}} \varphi$ if and only if $\Sigma \models_{\text{fin}} \varphi$. Indeed, all relational constraints considered so far are definable in first-order logic. For these constraints implication is *r.e.* (recursively enumerable) and finite implication is co-*r.e.*. As a result, if implication and finite implication coincide, then both are recursive, *i.e.,* decidable. For full dependencies, for example, the implication and finite implication problems coincide, and are both decidable.

Implication Versus Satisfiability

Consider Σ and φ defined over a relational schema $\mathbf{R}$. Obviously, $\Sigma \models_{\text{unr}} \varphi$ (resp. $\Sigma \models_{\text{fin}} \varphi$) if and only if the sentence $\wedge\ \Sigma\ \wedge \neg\varphi$ is (resp. finitely) satisfiable, *i.e.,* there exists a (resp. finite) instance of $\mathbf{R}$ that satisfies the sentence. This tells us that there is close connection between (resp. finite) implication and (resp. finite) satisfiability, problems studied in (resp. finite) model theory. For a constraint language $\mathcal{L}$ that is closed under negation (*i.e.,* if $\varphi \in \mathcal{L}$ then so is $\neg\varphi$), the (resp. finite) implication problem is just a special case of the (resp. finite) satisfiability problem.

Embedded dependencies are not closed under negation, and their (finite) implication analyses are quite different from their (finite) satisfiability counterparts. For any set Σ of embedded dependencies defined over a relational schema $\mathbf{R}$, the empty instance $\mathbf{I}\phi$ of $\mathbf{R}$, *i.e.,* an empty database with no tuples, satisfies Σ. For the analysis of (finite) satisfiability, embedded dependencies have the *domain independence* property: when considering whether a database $\mathbf{I}$ satisfies Σ, it suffices to consider the tuples in $\mathbf{I}$ without worrying about the underlying domains of the attributes in $\mathbf{R}$. In contrast, to verify $\Sigma \models_{\text{unr}} \varphi$ (resp. $\Sigma \models_{\text{fin}} \varphi$) one may have to consider all (resp. finite) instances of $\mathbf{R}$, possibly

ranging over all the values in the underlying domains of the attributes in **R**.

Despite this, when studying (finite) implication analyses for certain subclasses of embedded dependencies, it is still possible to capitalize on results on their (finite) satisfiability problems. For example, when Σ and φ are full dependencies, $\wedge\ \Sigma\ \wedge\neg\varphi$ is an $\exists^*\forall^*$ sentence (with equality or not), in a fragment of first-order logic known as the Bernays-Schőnfinkel-Ramsey class. It is known that for the Bernays-Schőnfinkel-Ramsey class, the satisfiability and finite satisfiability problems coincide and are decidable in NEXPTIME (non-deterministic exponential time; it is in fact NEXPTIME-complete in the absence of functions). This yields an upper bound on the implication and finite implication problems for full dependencies.

It is worth mentioning that when it comes to XML, the interaction between schemas (types) and integrity constraints becomes more intriguing than their relational counterparts, and as a result, the finite satisfiability problem becomes much harder. Indeed, for a class $\mathcal{L}_0$ of unary keys and foreign keys for XML, it is undecidable to decide, given a set Σ of constraints in $\mathcal{L}_0$ and a DTD D_0 (document type definition), whether or not there exists a finite XML document that satisfies both Σ and D_0. There are, however, interesting connections between finite satisfiability and finite implication analyses for XML constraints (see [4,13] for detailed discussions).

Finite Axiomatizability

Another important approach to studying (finite) implication of constraints is based on finite axiomatization. A finite axiom system $\mathcal{A}$ for a class $\mathcal{L}$ of constraints consists of finitely many axiom schemes and inference rules. A *proof* of a constraint φ in $\mathcal{L}$ from a set Σ of constraints in $\mathcal{L}$ *using* $\mathcal{A}$ is a finite sequence $\varphi_1,...,\varphi_n$ such that φ_n is φ, and for each $i \in [1, n]$, (i) $\varphi_i \in \Sigma$, or (ii) φ_i is an instance of an axiom scheme in $\mathcal{A}$, or (iii) φ_i follows from preceding constraints in the sequence by one of the inference rules in $\mathcal{A}$. If there exists such a proof, then φ is said to be *provable* from Σ using $\mathcal{A}$, denoted by $\Sigma \vdash_{\mathcal{A}} \varphi$.

For (resp. finite) implication of $\mathcal{L}$, the axiom system $\mathcal{A}$ is *sound* if $\Sigma \vdash_{\mathcal{A}} \varphi$ entails $\Sigma \models_{\text{unr}} \varphi$ (resp. $\Sigma \models_{\text{fin}} \varphi$); it is *complete* if $\Sigma \models_{\text{unr}} \varphi$ (resp. $\Sigma \models_{\text{fin}} \varphi$) entails $\Sigma \vdash_{\mathcal{A}} \varphi$. If $\mathcal{A}$ is both sound and complete then it is called a *finite axiomatization* of (finite) implication of $\mathcal{L}$, which characterizes (finite) implication of $\mathcal{L}$.

The best known example of finite axiomatizations is Armstrong's axioms for functional dependencies (FDs). Recall that an FD is of the form $X \rightarrow Y$, where X and Y are sets of attributes. Armstrong's axiom system consists of:

Reflexivity axiom: $X \rightarrow X$.

Augmentation: If $X \rightarrow Y$ then $XZ \rightarrow YZ$, where XZ denotes $X \cup Z$.

Transitivity: If $X \rightarrow Y$ and $Y \rightarrow Z$ then $X \rightarrow Z$.

Relative to a set U of attributes, an instance of the reflexivity axiom $X \rightarrow X$ is $V \rightarrow V$ when V is a subset of U. An FD φ_i follows from preceding constraints in a proof $\varphi_1,...,\varphi_n$ if there exist a reference rule (either augmentation or transitivity) ρ and a substitution σ from the variables in ρ to subsets of U, such that for each FD in the antecedent of ρ, the corresponding FD obtained via σ is φ_j for some $j < i$. For example, one can easily verify that $\varphi = A \rightarrow C$ is provable from $\Sigma = \{A \rightarrow B, B \rightarrow C\}$ using Armstrong's axioms, *i.e.,* $\Sigma \vdash \varphi$.

For finite implication of $\mathcal{L}$, the existence of a finite axiomatization is a stronger property than the existence of a testing algorithm. Indeed, from the existence of a finite axiomatization for finite implication of $\mathcal{L}$ follows the decidability of the finite implication problem for $\mathcal{L}$ (because the finite implication becomes r.e., and it is co-r.e. for first-order logic). On the other hand, there are dependencies (*e.g.,* join dependencies JDs), such that for their finite implication there is no finite axiomatization, but there is a testing algorithm (via the chase [3,15]).

Finite axiomatizations have been developed for implication and finite implication of a variety of dependencies, including FDs, INDs, MVDs, FDs and MVDs taken together, and for full typed dependencies. There is also a sound and complete axiom system for implication (*not* finite implication) of typed TGDs and EGDs. On the other hand, it is known that there exist no finite sound and complete axiom systems for FDs and INDs taken together.

Complexity of Implication Analyses

A number of complexity results have been established on the implication and finite implication problems for a variety of constraint languages. As remarked earlier, for full dependencies, the implication and finite implication problems coincide and are both decidable. For the following classes of full dependencies, the implication and finite implication have the complexity bounds

given below, in which n is the length of the input constraints Σ and φ:

(i) in $O(n)$-time for FDs [6];
(ii) in $O(n \log n)$-time for MVDs;
(iii) in $O(n^2)$-time for deciding whether an MVD or an FD is implied by a set of typed full dependencies;
(iv) in $O(n^2 \log^2 n)$-time for deciding whether a JD is implied by a set of FDs.

In contrast to the NEXPTIME upper bound on the Bernays-Schőnfinkel-Ramsey class given earlier, there exists an efficient decision procedure for these fragments of full dependencies. On the other hand, several intractability results have also been established:

(v) It is NP-hard to decide whether a set of MVDs implies a JD, and it is NP-complete to decide whether a JD and an FD imply a JD.
(vi) The implication problem for full dependencies is EXPTIME-complete, typed or untyped.

Beyond full dependencies, fewer positive results are known.

(vii) The implication and finite implication problems for INDs coincide and are PSPACE-complete [9].

A well-known undecidability result is the following, proved independently by Chandra, Vardi [11] and by Mitchell:

(viii) For FDs and INDs put together, the implication and finite implication problems differ, and are undecidable.

When it comes to TGDs, the analysis of implication is also beyond reach in practice:

(ix) For TGDs, the implication and finite implication problems are undecidable, even when only typed TGDs are considered.

One of the most powerful tools for testing implication is the chase [3,15]. To determine whether or not Σ implies φ, the basic idea of the chase is to represent φ as a tableau T, and repeatedly apply constraints in Σ to T. This yields a sequence of tableaux, referred to as a chasing sequence of T by Σ, and as a terminal sequence if it is finite and no constraint in Σ can be further applied to it. If a chasing sequence reaches a tableau satisfying a certain condition (depending on what type of φ is), then one can conclude that φ is implied by Σ.

When Σ is a set of full dependencies and φ is a typed dependency, it is known that chasing terminates and has the Church-Rosser property, *i.e.,* different terminal chasing sequences yield the same unique result. Based on this an EXPTIME algorithm can be developed for testing whether $\Sigma \models_{\text{unr}} \varphi$ and $\Sigma \models_{\text{fin}} \varphi$, for full dependencies.

When Σ is a set of embedded dependencies, however, the chase may not terminate. Nevertheless, a decision procedure based on the chase can be developed such that it will give a positive answer if $\Sigma \models_{\text{unr}} \varphi$, and will not terminate if $\Sigma \not\models_{\text{unr}} \varphi$. See [1] for detailed discussions.

View Dependencies

An important application of logical implication and the chase is the analysis of constraint propagation from databases to their views. Let $\mathbf{R}$ be a database schema, Σ a set of constraints over $\mathbf{R}$, V a view definition on $\mathbf{R}$, and φ a constraint defined over the view. Then $\mathbf{R}$: Σ *implies* V: φ, denoted by $\mathbf{R}$: $\Sigma \models V$: φ, if $V(\mathbf{I})$ satisfies φ for each instance $\mathbf{I}$ of $\mathbf{R}$ that satisfies Σ. The *constraint propagation* problem (*a.k.a.* the view dependency inference problem) is to determine, given $\mathbf{R}$, Σ, V and φ, whether or not $\mathbf{R}$: $\Sigma \models V$: φ. The analysis of constraint propagation is useful in, among other things, data exchange, data integration and data cleaning.

For example, when $\mathbf{R}$ consists of a relation schema R with attributes (A, B, C, D), Σ consists of an FD $A, B \rightarrow C$, V is the query $\pi_{A, B, C}(\sigma_{A=1} R)$ in the relational algebra, and φ is $B \rightarrow C$, one can see that $\mathbf{R}$: $\Sigma \models V$: φ.

The constraint propagation problem has been studied for full dependencies and for views defined in the relational algebra, based on an extension of the chase technique. The following complexity bounds are known.

(i) It is undecidable when views V are defined in the relational algebra and the constraints Σ and φ are FDs.
(ii) When $\Sigma \cup \{\varphi\}$ is a set of FDs and MVDs, and views are SPCU queries (selection, projection, Cartesian product and union), the constraint propagation problem is decidable in polynomial time.
(iii) When Σ is a set of FDs and JDs, φ is a JD, and views are SPCU queries, the problem is NP-complete.
(iv) When $\Sigma \cup \{\varphi\}$ is a set of full dependencies and views are SPCU queries, the problem is EXPTIME-complete.

Constraint propagation has also been studied in the context of XML shredding, *i.e.*, for mapping XML data to relations, from XML keys to relational FDs.

Key Applications

Traditional applications of the analysis of constraint implication include schema normalization, data integrity maintenance, storage implementation, and query optimization (see [1]). The prevalent use of the Web has motivated the development of new constraint languages for specifying the semantics of semi-structured data and XML, as well as for capturing the consistency of data. Logical implication has found new applications in XML query optimization, data integration, data exchange and data cleaning.

Future Directions

Several problems remain open for implication analysis of constraints developed for specifying XML semantics and for data cleaning. For example, the exact complexity bounds on the implication problems for certain XML functional dependencies in the presence and in the absence of DTDs are not yet settled. Another topic is the development of efficient algorithms for testing implication of constraints. For a variety of constraint languages the (finite) implication problem is intractable or even undecidable, *e.g.*, for functional and inclusion dependencies taken together. It is important and practical to find effective heuristic algorithms for their implication analyses, ideally with certain performance guarantees. This issue deserves a full treatment.

Cross-references

- ▶ Constraint-Drive Database Repair
- ▶ Data Exchange
- ▶ Database Dependencies
- ▶ Logical Structure
- ▶ Normal Forms and Normalization
- ▶ XML Integrity Constraints

Recommended Reading

1. Abiteboul S., Hull R., and Vianu V. Foundations of databases. Addison-Wesley, Reading, MA, USA, 1995.
2. Abiteboul S. and Vianu V. Regular path queries with constraints. J. Comput. Syst. Sci., 58(3):428–452, 1999.
3. Aho A.V., Beeri C., and Ullman J.D. The theory of joins in relational databases. ACM Trans. Database Syst., 4(3):297–314, 1979.
4. Arenas M., Fan W., and Libkin L. Consistency of XML specifications. In Inconsistency Tolerance. Springer, Berlin, 2005, pp. 15–41.
5. Armstrong W.W. Dependency structures of data base relationships. In Proc. IFIP Congress, 1974, pp. 580–583.
6. Beeri C. and Bernstein P.A. Computational problems related to the design of normal form relational schemas. ACM Trans. Database Syst., 4(1):30–59, 1979.
7. Beeri C. and Vardi M.Y. The implication problem for data dependencies. In Proc. 8th Int. Colloquium on Automata, Languages, and Programming, 1981, pp. 73–85.
8. Bernstein P.A. Synthesizing third normal form relations from functional dependencies. ACM Trans. Database Syst., 1(4):277–298, 1976.
9. Casanova M.A., Fagin R., and Papadimitriou C.H. Inclusion dependencies and their interaction with functional dependencies. In Proc. 1st ACM SIGACT-SIGMOD Symp. on Principles of Database Systems, 1982, pp. 171–176.
10. Chandra A.K., Lewis H.R., and Makowsky J.A. Embedded implicational dependencies and their inference problem. In Proc. 13th Annual ACM Symp. on Theory of Computing, 1981, pp. 342–354.
11. Chandra A.K. and Vardi M.Y. The implication problem for functional and inclusion dependencies is undecidable. SIAM J. Comput., 14(3):671–677, 1985.
12. Fagin R. and Vardi M.Y. The theory of data dependencies – an overview. In Proc. 11th Int. Colloquium on Automata, Languages, and Programming, 1984, pp. 1–22.
13. Fan W. and Libkin L. On XML integrity constraints in the presence of DTDs. J. ACM, 49(3):368–406, 2002.
14. Klug A.C. Calculating constraints on relational expressions. ACM Trans. Database Syst., 5(3):260–290, 1980.
15. Maier D., Mendelzon A.O., and Sagiv Y. Testing implications of data dependencies. ACM Trans. Database Syst., 4(4):455–469, 1979.

Implications of Genomics for Clinical Informatics

Mollie Ullman-Cullere, Eugene Clark, Samuel Aronson

Harvard Medical School – Partners Healthcare Center for Genetics and Genomics, Boston, MA, USA

Synonyms

Biomedical informatics; Bioinformatics; Clinical genetics; Clinical genomics; Medical genetics

Definition

Integration of genetic test results, generated in the clinical laboratory, into the electronic medical record, in a fully structured format enabling enhanced security,

contextual views, clinical decision support, pharmacovigilance, disease management, outcomes and quality assessment.

Historical Background

Clinical genetics got its start in 1948 with the founding of the American Society of Human Genetics, which formalized a scientific approach to the study of human genetics [11]. Traditionally clinical genetics requires practitioners to function as data integrators. Like traditional healthcare, tests are ordered and results returned as interpretive reports delivered in paper form. Understanding the composite picture of the *phenotype* and *genotype* of the patient requires transcribing key signs, symptoms, test values and their clinical interpretation into yet another document, contributing more paper to the aggregate patient record. This process will be repeated numerous times over the course of the patient's lifetime, as new health problems are assessed.

Correcting this problem is similar to overcoming the historical barriers to the electronic health record (EHR) [8]. Data standards need to be created and adopted, testing laboratory infrastructure built to structure genetic test results, interfaces developed to send and receive these data, and EHR's and associated clinical decision support tools and knowledgebases also must be enhanced to accept genetic data and inform clinicians of implications in the context of patient care. This introduces new challenges for the underlying database technologies used in the clinical IT infrastructure.

Foundations

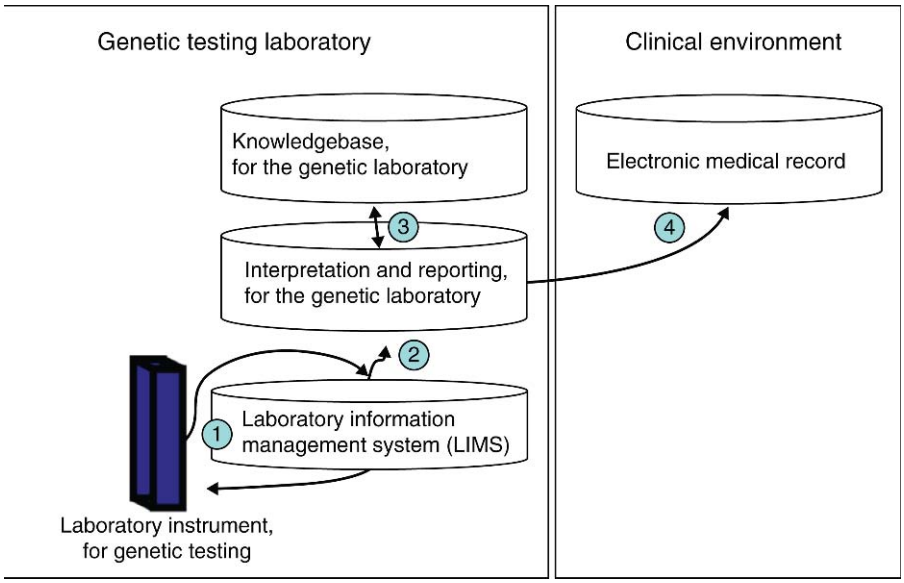
Every person inherits 3 billion base pairs of DNA from each of their parents. Contained in this DNA are regions representing an estimated 22,000 genes. Each of these genes can create RNA that then goes on to create one or more proteins. These proteins participate in complex pathways that perform most of the functions within the body – everything from cell division and food metabolism, to muscle contraction and light perception, just to name a few. When an individual's genomic DNA sequence was compared to the National Center for Biotechnology Information reference nucleotide sequence assembly more than 4.1 million differences were found [6]. Some of these differences, called "DNA variants" are benign in nature, resulting in no measurable phenotypic change. Others result in non-deleterious change, for example, difference in blood type, eye and hair color. While others negatively alter the way protein pathways function within an individual and thereby impact their health. DNA variants, either individually or in combination, are a significant factor, and often an outright cause, of most human disease.

The scope and complexity of this biological genomic infrastructure creates significant knowledge and data management challenges, perhaps the most significant ever encountered. A wide variety of data types such as those associated with DNA, RNA and protein sequences and identified variations have been created. Biological entities such as genes, variants/mutations and proteins will need to be annotated in a manner that provides linkage to public databases for clinical genetic and bioinformatic information, as well as clinical phenotype in the EHR.

Integrating Genetic Data into the Clinical Record

Clinicians and molecular geneticists access genetic information in different ways and for different purposes. This creates new challenges on underlying technologies for supporting new functional requirements. Some key use case scenarios include: establishing the patient context; ensuring data integrity, with patient linkage, throughout the testing process; managing laboratory workflows (often iterative in nature); integrating with instruments; supporting both manual and automated data review; managing quality control and quality assessment; supporting a reporting mechanism; providing decision support for the clinician. In order to meet this challenge a genetic testing laboratory requires: (i) a Laboratory Information Management System (LIMS), to support collection and analysis of raw data, (ii) an interpretation and reporting tool, to assist the geneticist in translating results into clinical implications and report these results in structured form, (iii) a genetic knowledgebase, utilized by the interpretation and reporting tools, to appropriately interpret and structure the results, and (iv) a electronic medical record enhanced to accept and leverage genetic data (see Fig. 1).

Data models, ontologies, and message structure also present a challenge. Properly structuring and annotating genetic data in the EHR is key to effectively displaying, integrating into clinical workflows, and using this data in context of clinical decision support. With this goal in mind, Healthcare Information Technology (HIT) standards are used to create the message between LIMS and EHR's and provide ontological linkage to phenotypic data, while clinical genetic and



Implications of Genomics for Clinical Informatics. Figure 1. IT Infrastructure required for reporting of structured genetic test results into the electronic medical record.

bioinformatic standards are leveraged for structuring of genetic specific data.

Key Applications

Laboratory Information Management Systems for Genetic Laboratories

Raw data files from instruments used in genetic testing contain highly structured data. It is in the process of summarizing, interpreting and translating these results that the geneticist creates a narrative report, communicating these results to the clinician. Structuring genetic data in the EHR first requires LIMS to manage genetic data during the testing process, even as the data is iteratively collected.

Laboratory Information Management Systems support process flows for individual laboratories. At times it is important to support process flows that span laboratories in which case an umbrella LIMS application, an enterprise LIMS superstructure, is required. An enterprise LIMS superstructure, called the Gateway for Integrated Genomics-Proteomics Applications and Data (GIGPAD) [14], was constructed in a collaboration between Harvard Medical School – Partners HealthCare Center for Genetics and Genomics (HPCGG), Partners HealthCare Information Systems department and Hewlett Packard [5]. GIGPAD is responsible for overall process coordination. It also manages all electronic contact between laboratories and external systems and users. GIGPAD supports multiple laboratories. In some cases, the decision was made to support laboratories by purchasing vendor LIMS and integrating them under the GIGPAD umbrella. In other cases, the decision was made to add support for laboratories by constructing custom LIMS within GIGPAD.

GIGPAD leverages a J2EE architecture backed by an Oracle database. The J2EE platform in general, and EJB in particular, is a good match for this use. GIGPAD needs to support a large number of different types of processes. However, these processes often have common elements. The J2EE platform has enabled the construction of object models that facilitate reuse.

Security is a very important consideration for a LIMS or enterprise LIMS superstructure that handles confidential data. J2EE is a good platform for constructing and enforcing object based security although the platform's generic capabilities need considerable enhancement.

Knowledgebases and Reporting Systems for Genetic Laboratories

The field of clinical genetics maintains knowledgebases called Locus Specific Databases (LSDB), in which DNA variants and associated phenotypes are cataloged [1]. However, LSDB's require manual look-up and synthesis to understand the clinical meaning of a variant and do not utilize HIT standards to code phenotype. The

field of bioinformatics utilizes NCBI's dbGaP database to catalog DNA variants and associated phenotype [15]; however, this database contains research data and is not appropriate for clinical care.

In an effort to fill a necessary gap, HPCGG and Partners Healthcare created GeneInsight, a knowledge database that associates genetic variations with clinical annotations using newly extended HIT standards (LOINC, SNOMED, and RxNORM). The variants entered into GeneInsight are validated to ensure that they adhere to the clinical genetics standards for naming variants, using HGVS nomenclature and are valid when compared with locally defined reference sequences. The data stored for each variant includes the DNA change, amino acid change (if applicable), classification (e.g., Pathogenic), source classification (e.g., somatic or germline), and references to other databases including PubMed and dbSNP. In addition, GeneInsight is capable of programmatically deriving the DNA and amino acid change types and flanking sequences.

In addition to variants, GeneInsight also contains records for genes and diseases or conditions. The gene records include references to other databases including NCBI's Gene and PubMed. Reference sequences for the genes are also recorded in GeneInsight and linked to the NCBI's Nucleotide database. Diseases or conditions can represent either a typical disease, such as cystic fibrosis, or a pharmacogenetic condition, such as warfarin metabolism. Medically significant variants are then linked to one or more diseases and/or conditions to establish their clinical context or phenotype.

In order to create the narrative, human readable report (for the clinician) and the machine readable, structured genetic results (for the EHR), a geneticists uses a genetics interpretation and reporting tool which leverages data in a genetic knowledgebase, to create draft reports. Within HPCGG's Laboratory for Molecular Medicine, the Genomic Variant Interpretation Engine (GVIE) is used during the process of reporting genetic test results and co-creates both the narrative report and the structured genetic data for the EHR. GVIE contains definitions of the tests that are run by the HPCGG including the coverage of the test and also defines a series of templates for each disease or condition. The templates are dynamic to allow the automatic insertion of case specific information and can be associated with rules that govern to which cases they are applied. The rules can take into account the specific test that was run, the overall result, the number of variations detected, the classification of those variations, and other information. In addition, the overall result (e.g., Positive) is generated by a separate set of rules that change very infrequently and are driven by the disease's inheritance and the identified variation classification and allele state.

Cerner Corporation has also developed a LIMS and reporting tools for sending structured genetic data into an electronic medical record. Here Cerner uses the Clinical Bioinformatics Ontology (CBO), in lieu of the larger genetic knowledgebase, to report structured genetic variants aligned with NCBI reference sequences [4]. This is a streamlined solution focusing on reporting variants identified during testing.

Integrated Clinical and Genetic Medical Record

A genetically aware clinical medical record effectively integrates highly structured genetic testing results with other laboratory and clinical data, utilizing accepted healthcare informatics standards. These standards include Health Level Seven (HL7) [3] for messaging, Logical Observations Identifiers, Names, Codes (LOINC) [7] as a coding system, and Systematized Nomenclature of Medicine-Clinical Terms (SNOMED CT) [12] and RxNORM [10] as disease and medication terminologies. By using these standards, genetics can be integrated into existing electronic health record databases, leveraging models for storage, retrieval and indexing. Most importantly, these standards aid in the mapping of a patient's genetic information with other clinical data on the patient. As clinical genomics is an emerging field, healthcare informatics standards organizations (including HL7, LOINC, and SNOMED) are in the process of extending their models to meet these needs; therefore, the most up-to-date information can be found on standards organization websites (see Recommended Reading).

Clinical Decision Support

Genomic clinical decision support (CDS) will model existing CDS based on laboratory data. Genetic tests which provide *pharmacogenomic* information will determine how an individual will respond to medication. CDS rules, leveraging this data, would take the form of drug dosage assistance, contraindications, and recommended alternative therapy. Genetic tests which provide *diagnostic* information will feed into problem lists (that can then leverage pre-existing CDS). Genetic tests which identify a patient as being at *increased risk*

for disease will feed into other clinical data used to identify high-risk populations and CDS focused on disease management. The precise instantiation models for genomic CDS are based on preexisting models and remain to be thoroughly tested for identification of gaps.

Cross-references

- ▶ Clinical Data Acquisition, Storage and Management
- ▶ Clinical Data and Information Models
- ▶ Clinical Decision Support
- ▶ Electronic Health Record
- ▶ Storage Management

Recommended Reading

1. Fokkema I.F., den Dunnen J.T., and Taschner P.E. LOVD: easy creation of a locus-specific sequence variation database using an "LSDB-in-a-box" approach. Hum. Mutat., 26(2):63–68, 2005.
2. GeneTests – medical genetics information resource including gene reviews, genetic testing laboratory and clinical directories, as well as educational materials. Available online at: http://www.genetests.org.
3. Health Level Seven (HL7) – focusing on messaging, HL7 is an American National Standards Institute (ANSI) accredited. Available online at: http://www.HL7.org/
4. Hoffman M.A. The genome-enabled electronic medical record. J. Biomed. Inform., 40(1):44–46, 2007.
5. HP Supports "Individualized Medicine" Initiative at Partners, Sept. 30, 2003, Available online at: http://www.hp.com/hpinfo/newsroom/press/2003/030930a.html (retrieved on October 5, 2007).
6. Levy S., Sutton G., Ng P.C., Feuk L., Halpern A.L., Walenz B.P., Axelrod N., Huang J., Kirkness E.F., Denisov G., Lin Y., MacDonald J.R., Pang A.W., Shago M., Stockwell T.B., Tsiamouri A., Bafna V., Bansal V., Kravitz S.A., Busam D.A., Beeson K.Y., McIntosh T.C., Remington K.A., Abril J.F., Gill J., Borman J., Rogers Y.H., Frazier M.E., Scherer S.W., Strausberg R.L., and Venter J.C. The diploid genome sequence of an individual human. PLoS Biol., 5(10):2113–2144, 2007.
7. Logical Observation Identifiers Names and Codes (LOINC) – focusing on the pooling of laboratory results and observations, LOINC is ANSI accredited. Available online at: http://www.regenstrief.org/medinformatics/loinc/
8. McDonald C.J. The barriers to electronic medical record systems and how to overcome them. J. Am. Med. Inform. Assoc., 4(3):213–221, 1997.
9. Online Mendelian Inheritance in Man (OMIM) – a catalog of human genes and genetic disorders. Available online at: http://www.ncbi.nlm.nih.gov/sites/entrez?db=omim.
10. Parrish F., Do N., Bouhaddou O., and Warnekar P. Implementation of RxNorm as a terminology mediation standard for exchanging pharmacy medication between federal agencies. AMIA Annu Symp Proc., 2006:1057, 2006.
11. Rimoin D.L. and Hirschhorn K. History of medical genetics in pediatrics. Pediatr. Res., 56(1):150–9, 2004.
12. Systematized Nomenclature of Medicine-Clinical Terms (SNOMED CT) focusing on clinical terminology, SNOMED is ANSI accredited. Available online at: http://www.ihtsdo.org/
13. The Database of Genotype and Phenotype (dbGaP) – serves as an archive for distribution of clinical study results containing both genotype and phenotype data. Available online at: http://www.ncbi.nlm.nih.gov/sites/entrez?db=gap
14. The Harvard Medical School-Partners HealthCare Center for Genetics and Genomics (HPCGG). Response to the Department of Health and Human Services Request for Information (RFI): Improving Health and Accelerating Personalized Health Care through Health Information Technology and Genomic Information in Population- and Community-based Health Care Delivery Systems. Available online at: http://www.hpcgg.org/News/HPCGG_RFI_Response_1_0.pdf (retrieved on August 14, 2007).
15. Wheeler D.L., Barrett T., Benson D.A., Bryant S.H., Canese K., Chetvernin V., Church D.M., Dicuccio M., Edgar R., Federhen S., Feolo M., Geer L.Y., Helmberg W., Kapustin Y., Khovayko O., Landsman D., Lipman D.J., Madden T.L., Maglott D.R., Miller V., Ostell J., Pruitt K.D., Schuler G.D., Shumway M., Sequeira E., Sherry S.T., Sirotkin K., Souvorov A., Starchenko G., Tatusov R.L., Tatusova T.A., Wagner L., and Yaschenko E. Database resources of the National Center for Biotechnology Information. Nucleic Acids Res., 36:D13–D21, 2008.

Implicit Event

Jonas Mellin, Mikael Berndtsson
University of Skövde, Skövde, Sweden

Definition

In active databases, an implicit event is an event that is implied by an ECA rule definition.

Key Points

ECA rules were developed as an optimization of condition action rules. The performance of rule evaluation was improved by allowing, or even requiring, explicit definition of when rules should be triggered in the form of events. However, it turns out that it is possible to derive events from the condition in a meaningful way in some cases and, thus, there are events that can be implied by the ECA rule definition. For example, if the condition is a logical expression $A = 5 \wedge B = 3$, then a disjunction of the events representing update to the variables can trigger the rule. That is, with implicit events (**IF** $A = 5 \wedge B = 3$ **DO** action) is equivalent to (**ON** $update_A$ **or** $update_B$ **IF** $A = 5 \wedge B = 3$ **DO** action) with explicit events.

Cross-references

- ▶ Atomic Event
- ▶ Composite Event
- ▶ ECA Rules
- ▶ Event
- ▶ Event Detection
- ▶ Event Specification
- ▶ Explicit Event

Imprecise Data

- ▶ Data Uncertainty Management in Sensor Networks

Imprecise Spatial Queries

- ▶ Probabilistic Spatial Queries

Imprecise Time

- ▶ Temporal Indeterminacy

Imputed Data

- ▶ Synthetic Microdata

IMS Data Model

- ▶ Hierarchical Data Model

In Silico Experiment

- ▶ Scientific Workflows

Incoherency Bounds

- ▶ Replica Freshness

Incomplete Information

Gösta Grahne
Concordia University, Montreal, QC, Canada

Synonyms

Uncertain information; Null values; Indefinite information

Definition

Incomplete information arises in relational databases, when a fact (tuple) has to be inserted in a relation, and values for some required columns are missing. For instance, in an employee database, the phone number of one employee might be missing, as might the address of another employee. There are numerous reasons for such missing information, e.g., the insertion was done through a view, or the incomplete tuple originated from another database that does not record these fields. In information integration and data exchange systems incomplete information is rampant. Note that here the null values only represent unknown, existing values. The other main case, namely the one that the column heading is not applicable to the tuple in question, is not covered here, for a treatment, see e.g., [17].

It is easy to *store* missing or incomplete information, by simply using a symbol, say $\perp$, different from the symbols of the domain of the database. As an illustration, consider the following employee database:

Employee	Department	Home Town	Phone no.
Jerry	*Sales*	*New York*	$\perp$
Elaine	*Accounting*	$\perp$	*123*
George	*Services*	*New York*	*456*

In the table above, Jerry's phone number is unknown as is Elaine's home address. The third tuple has complete information. So far the picture is quite uncomplicated. The difficulties arise when *queries* are applied to an incomplete database. For instance, suppose the query asks for all employees living in New York. Should Elaine be included in the answer, as it is not ruled out that she lives in New York? Her unknown hometown could very well be New York. It seems that Elaine *might* be in the answer.

For another example, suppose the relation is decomposed into R(Employee, Department, Home Town) and S(Home Town, Phone no.) Now, joining back R and S, at least the original relation should be recovered. However, it is not clear how to join the R-tuple (*Elaine, Accounting*, $\perp$) with the S-tuple ($\perp$, *123*). Nevertheless, whatever Elaine's home address is, the null-value in the R-tuple must clearly be the same as the null-value in the S-tuple, as they both represent Elaine's unknown home town. Therefore, the tuple (*Elaine, Accounting*, $\perp$, *123*) should *surely* be in the result of the join.

Historical Background

The enormously successful relational model emerged out of algebra and logic, starting with Codd's fundamental 1970's papers. By the end of the decade, the problem of null values was noted and resulted in some early efforts not covered here, see e.g., [17,16]. Around this time Ray Reiter also promulgated the idea of "Proof theoretical" vs. "Model Theoretical" interpretation of the relational model (see references in [12]). Roughly, one could say that in proof theory the database is a set of ground facts in First Order Logic, and the query (also a logical formula) answers are something to deduce from the theory by proof-theoretic means. In "Model Theory" on the other hand, the database is a finite structure, and the queries are algebraic operators applied to the database. Indeed, Reiter proposed a first order interpretation of the null value: Since a regular tuple $(a, b) \in r$ is interpreted as a ground fact $R(a, b)$, it is plausible to interpret $(a, x) \in r$, where x is a null value, as the logical sentence $\exists x : R(a, x)$. Reiter then developed query answering algorithms for his (existentially) extended relational theories [12].

At the same time, relational theory started developing, creating its own apparatus, containing lossless joins, tableaux and the chase, in addition to many other useful tools and concepts, such as the "crown jewel" relational algebra. Working along these lines, T. Imielinski and W. Lipski came out in 1981 with a *VLDB* conference abstract of their landmark paper *Incomplete Information in Relational Databases*, published in *J. ACM* in 1984 [8].

First, Imielinski and Lipski call a relation containing null values a *table* (as opposed to a relation). One assumes two disjoint countably infinite sets of *constants*, denoted *a*, *b*, *c*, *Jerry, Elaine*,...,and of *variables*, denoted *x*, *y*, *z*,.... The table in the introduction, could be written (concisely) as {(*Jerry, Sales, New York, x*),(*Elaine, Accounting, y, 123*), (*George, Services, New York, 456*)}

Second, [8] adopts the *possible worlds* interpretation of incomplete databases: an *incomplete database is a set of complete databases*, one of which is or corresponds to the real world. In other words, the knowledge about the real world, that the incomplete database has is exactly this set. In the sequel (usually infinite) incomplete databases (sets of complete databases) are denoted by $\mathscr{X}, \mathscr{Y}, \ldots$.

Tables represent incomplete databases through valuations, that instantiate the null values to ordinary, known values. Formally, a *valuation* is a mapping from the variables to the constants, that is identity on the constants. Valuations are extended to tuples and tables in the obvious homeomorphic way. Now a database r represents one of the possible worlds of table T, if there is a valuation v, such that $r = v(T)$. For instance, let $T = \{(a,x), (b, y)\}$. Pretend for a moment that domain of constants is finite and only includes a and b. Then T represents $\mathscr{X}$, where $\mathscr{X} = \{\{(a, b), (b, b)\}, \{(a, a), (b, b)\}, \{(a, b), (b, a)\}, \{(a, a), (b, a)\}\}$. Thus $Rep(T)$, the incomplete database $\mathscr{X}$, represented by a table T, is defined as

$$Rep(T) = \{r : r \supseteq v(T), \text{for some valuation } v\}. \quad (1)$$

Note that the *open world* assumption is made, when requiring only $r \supseteq v(T)$. It means that any fact not recorded in all possible databases is considered unknown. Note further that under an open world assumption, a *complete* database r actually represents all supersets of r. Since nothing is surely false, it is clear that querying under the open world assumption cannot include negation. In a *closed world* interpretation, on the other hand, any fact *not* recorded in the database, or not instantiable from a tuple in a table is considered false. For tables T, the closed world interpretation of the possible databases it represents is defined as $Rep_{\text{cwa}}(T) = \{r : r = v(T), \text{for some valuation } v\}$. This means that a tuple that does not belong to any $r \in Rep_{\text{cwa}}(T)$ is considered false.

The main insight when processing a query q on an incomplete database is as follows: since the "real" database r is one within a set $\mathscr{X}$, the "real" answer

should ideally be $q(r)$. However, given only $\mathscr{X}$, what can be captured with query q is the set $q(\mathscr{X}) = \{q(r) : r \in \mathscr{X}\}$. In particular, if $\mathscr{X}$ is represented by a table T, it would be desirable to find a table, denote it $\hat{q}(T)$, such that

$$Rep(\hat{q}(T)) = q(Rep(T)). \quad (2)$$

This is where the technical development can begin.

Foundations

First, note that the open world assumption itself is causing a subtle difficulty when trying to satisfy the commutativity requirement (2). Consider the table T containing a single tuple (a, b). After applying a selection $\sigma_{A=a}$ on T one would (correctly) expect that the tuple (a, b), and nothing else, is in the resulting table $\widehat{\sigma_{A=a}}(T)$. However, (2) is not satisfied, since for instance (c, d) occurs in some $r \in Rep(\widehat{\sigma_{A=a}}(T))$, whereas $(c, d) \notin r$, for all $r \in \sigma_{A=a}(Rep(T))$, as such tuples have been dropped by the selection. Unless the closed world assumption is adopted – in which case clearly $Rep_{\text{cwa}}(\widehat{\sigma_{A=a}}(T)) = \sigma_{A=a}(Rep_{\text{cwa}}(T))$ – condition (2) has to be relaxed.

The key concept for achieving this purpose is the notion of *certain answer*. Let q be a query and $\mathscr{X}$ an incomplete database. The certain answer to q on $\mathscr{X}$ consists of those tuples that are in $q(r)$, for *every* possible database $r \in \mathscr{X}$, that is, the certain answer is $\cap_{r\in\mathscr{X}} q(r) = \cap(q(\mathscr{X}))$.

Given a fixed query language, two incomplete databases are not distinguishable if they give the same certain answer to every query in the language. Formally, let $\mathcal{Q}$ be a query language (the set of all queries expressible in the language). Then incomplete databases $\mathscr{X}$ and $\mathscr{Y}$ are said to be $\mathcal{Q}$-*equivalent*, if $\cap(q(\mathscr{X})) = \cap(q(\mathscr{Y}))$, for all $q \in \mathcal{Q}$. This $\mathcal{Q}$-equivalence is denoted $\mathscr{X} \equiv_{\mathcal{Q}} \mathscr{Y}$. There is another equivalence relation on incomplete databases called *coinitiality*. Now $\mathscr{X}$ is coinitial with $\mathscr{Y}$, in symbols $\mathscr{X} \approx \mathscr{Y}$, if $\mathscr{X}$ and $\mathscr{Y}$ have the same minimal (with regard to subset) elements. Let $\mathcal{Q}_P$ be the set of all queries expressible in *positive* relational algebra (no negation, no inequalities in selection conditions), and let $\mathcal{Q}_{RA}$ be the set of *all* queries expressible in the full relational algebra. It turns out that provably $\mathscr{X} \equiv_{\mathcal{Q}_P} \mathscr{Y}$ iff $\mathscr{X} \approx \mathscr{Y}$, and $\mathscr{X} \equiv_{\mathcal{Q}_{RA}} \mathscr{Y}$ iff $\mathscr{X} = \mathscr{Y}$ [8,4].

Therefore, if the open world assumption if followed, the best approximation of (2) is to weaken it to

$$Rep(\hat{q}(T)) \equiv_{\mathcal{Q}} q(Rep(T)) \quad (3)$$

for some query language $\mathcal{Q}$. This led [8] to the notion of *representation system*. Let $\boldsymbol{T}$ be a class of tables, *Rep* the interpretation function, and $\mathcal{Q}$ a query language. Then the triple $(\boldsymbol{T}, Rep, \mathcal{Q})$ is a *representation system* if for all $T \in \boldsymbol{T}$, and all $q \in \mathcal{Q}$, there is a $\hat{q}(T) \in \boldsymbol{T}$, such that (3) is satisfied. Then the certain answer $\cap\,(q(Rep(T)))$ is clearly equal to $\cap\,(Rep(\hat{q}(T)))$. This means that the certain answer can be extracted from $\hat{q}(T)$, in some cases efficiently, as will be seen below.

The original paper [8] considers the classes consisting of Codd Tables, Naive Tables, and Conditional Tables. Other classes of tables can be found in [2,4,9,14]. Here, the attention is restricted to the original three table classes.

A *Codd table* is a table where each occurrence of a null value is represented by the same symbol (as for example in the table in the first section of this entry). Let for instance the symbol $\perp$ denote a null value. Evaluating a query q (i.e., computation of $\hat{q}(T)$ in (3)) proceeds as in regular relational algebra, with the additional evaluation rules saying that $\perp \neq \perp$, and $\perp \neq a$, for all constants a, as long as the selection conditions are atomic, i.e., of the form $\sigma_{A=a}$ or $\sigma_{A\neq a}$. This is in fact the solution proposed by Codd, and currently implemented in most commercial database management systems. However, the largest query language that Codd tables can support consists of queries expressible in relational algebra using only projection and selection, where, notably, inequalities are allowed in selections. Using any more operators from the relational algebra makes it impossible to satisfy (3), that is, for some such queries q there is no Codd table $\hat{q}(T)$ satisfying (3). The same holds under the closed world assumption. In addition to this, selections can have arbitrary Boolean combinations of atomic selection conditions, but then testing whether a tuple satisfies such a condition becomes coNP-complete. This is due to the fact that any propositional formula can be encoded as a compound selection formula. Note however, that the coNP-completeness depends on the fact that the query is part of the input (expression complexity). For data complexity, the evaluation can be carried out in time polynomial in the size of the table.

A *Naive table* is a table using *distinguishable* nulls, denoted by variables $x,y,\ldots$. Query evaluation uses the rules $x \neq a$, $x \neq y$, $x = x$, for all variables x, y and

constants a. Thus the tuple (*Elaine, Accounting*,$\perp$) would indeed join with the tuple ($\perp$,*123*), resulting in tuple (*Elaine, Accounting*,$\perp$,*123*). These tuples would of course be represented as (*Elaine, Accounting,y*), and (*y,123*). It turns out that if $\mathcal{Q}_P$ is chosen as query language, then $(T_N, Rep, \mathcal{Q}_P)$, where T_N is the class of all Naive tables, indeed forms a representation system. It has also been shown that Naive tables can handle recursion, i.o.w. one can add any positive datalog program as a query, and still be able to $\approx$-represent the result [4]. More generally, one can note that query evaluation, i.e., computation of $\hat{q}(T)$, proceeds by treating the null-values as constants, pairwise distinct, and distinct from all the "real" constants. Thus query evaluation in Naive tables has the same computational complexity as in the regular case. A similar result was also independently proved in [15].

Much has been written about certain answers in the database literature, but not always with complete transparency. Recall that the *certain answer* to a query q on $\mathscr{X}$, are those tuples that are in the answer to q for *every* possible database $r \in \mathscr{X}$. In other words, the certain answer to q on $\mathscr{X}$ is $\cap(q(\mathscr{X}))$. Note that $\hat{q}(T)$ is a Naive table that satisfies (3), for all positive datalog programs, or algebraic expressions in $\mathcal{Q}_P$. Conveniently, the certain answer can be obtained by retaining all variable-free tuples in $\hat{q}(T)$, as it is easily seen that $\mathscr{X} \approx \mathscr{Y}$ implies $\cap\mathscr{X} = \cap\mathscr{Y}$. The certain answer however looses some information from $\hat{q}(T)$. For a simple example, similar to the one in the introduction, let $T = \{(a, x, c)\}$ with schema (A, B, C). The certain answer to both $\pi_{A,B}(Rep(T))$ and $\pi_{B,C}(Rep(T))$ is empty. On the other hand, the certain answer to $\pi_{A,C}(\pi_{A,B}(Rep(T)) \bowtie \pi_{B,C}(Rep(T))) = \{(a, c)\}$. Here $\widehat{\pi_{A,B}}(T) = \{(a, x)\}$, $\widehat{\pi_{B,C}}(T) = \{(x, c)\}$ and $\{(a, x) \widehat{\bowtie} \{(x, c)\} = \{(a, c)\}$. Note how variables can be shared over several tables, above the x the left-hand side of the join is the same as the x on the right-hand side of the join. (On the other hand, for instance $\{(a, x) \widehat{\bowtie} \{(y, c)\} = \emptyset$.) Thus, if the query answer is to be used as a *view* for further querying, the answer should consist of all of $\hat{q}(T)$, not just the variable-free tuples. This aspect has been emphasized in [5]. Furthermore, as Lipski has shown in an all but forgotten paper [11], if query evaluation is to be *uniformly recursive* (such as the one for Naive tables), all tuples in $\hat{q}(T)$ need to be stored in the view.

Conditional tables. Naive tables form a representation system for the positive fragment of relational algebra (and for positive recursion). However, allowing set difference or inequalities in selection conditions in the queries might produce results not expressible as Naive tables. Consider for instance $\sigma_{A\neq a}(Rep(T))$, where $T = \{(a, b),(c, d),(x, e)\}$. A moments reflection will reveal the fact that $\sigma_{A\neq a}(Rep(T)) = \{\{(c, d)\},\{(c, d), (b, e)\},\{(c, d),(d, e)\},...\}$. It is easy to see that there is no Naive table that can represent this set. The problem is that the minimal elements in this set either has one tuple (when $x = a$), or two tuples (when $x \neq a$). The minimal elements in $Rep(T)$, for any (non-redundant) Naive table T, each have the same number of tuples. This problem can be overcome by using a stronger class of tables, albeit on the expense of tractability of query evaluation. This stronger class is called Conditional tables, here denoted T_C. A Conditional table is like a Naive Table, with the addition that each tuple has an associated condition, formed by a Boolean combination of atoms of the form $x = y$, $x \neq y$, $x = a$,$x \neq a$, $a = b$, $a \neq b$, for constants a,b and variables x, y. Note that for two distinct constants a and b, condition $a = b$ is tautologically false, and condition $a \neq b$ is tautologically true. In our previous example, $\sigma_{A\neq a}(Rep(T))$ can be represented by the Conditional table $U = \{(c, d);c \neq a,(x, e);x \neq a)\}$. The incomplete database represented by a conditional table T is defined as

$$Rep(T) = \{r \supseteq \nu(T) : \nu \text{ is a valuation}, \nu(T) = \{\nu(t) : t \in T \text{ and } \nu \text{ makes the condition in } t \text{ true}\}\} \quad (4)$$

For example, if ν is the valuation $x \rightarrow a$, and ν' the valuation $x \rightarrow b$, then, for U as above, $\nu(U) = \{(c, d)\}$ as the condition of the second tuple (x, d); $x \neq a$ becomes false. On the other hand, $\nu'(U) = \{(c, d), (b, e)\}$, since $\nu'(x \neq a)$ becomes $b \neq a$, which is true. Set difference is treated in a similar manner, for instance, subtracting $\{(c, b),(e, b)\}$ from $\{(x, b)\}$ results in the conditional table $\{(x, b);x \neq c \wedge x \neq e\}$. All in all, it holds that $(T_C, Rep, \mathcal{Q}_{P^{\neq}})$, is a representation system. Here $\mathcal{Q}_{P^{\neq}}$ denotes the query language obtained from $\mathcal{Q}_P$, by allowing inequalities in selection conditions. If set difference is to be incorporated, the closed world assumption has to be adopted. In this case it holds that $(T_C, Rep_{cwa}, \mathcal{Q}_{RA})$, is a representation system, actually satisfying the stronger condition (2). Positive datalog recursion can also be added both under the open and the closed world assumption.

For the computational complexity of query evaluation, recall that query evaluation in the system $(T_N, Rep, \mathcal{Q}_P)$ is polynomial in the number of tuples in T. Conditional tables are more complex. For all $q \in \mathcal{Q}_{RA}$ and conditional tables T, the conditional table $\hat{q}(T)$ can be computed in polynomial time. However, the conditions in table $\hat{q}(T)$ can have a convoluted structure. Thus testing whether a tuple t is in the certain answer, i.e., if $t \in \cap(Rep(\hat{q}(T)))$ is a coNP-complete problem. This result holds even if T is a Codd table, in which case a query $q \in \mathcal{Q}_{RA}$ is needed to get the lower bound (here the fact that $\hat{q}(T)$ is not a Codd table is ignored). On the other hand, for an arbitrary table $T \in T_C$, the query can be identity. In other words, the set $\cap(Rep(T))$ has a coNP-complete membership test. These, and further complexity results can be found in [2].

The theory of representation systems has also been extended to incorporate dependencies, see [8,4]. Let $\mathcal{X}$ be an incomplete database, and Σ a set of equality and weakly acyclic tuple generating dependencies. Denote by $\Sigma(\mathcal{X})$ the set of all minimal relations s, such that $s \supseteq r$, for some $r \in \mathcal{X}$, and s satisfies all dependencies in Σ. It has been shown that for all Naive tables T, there is a Naive table $\Sigma(T)$, such that $Rep(\Sigma(T)) \approx \Sigma(Rep(T))$. If conditional tables are used, the coinitiality can be replaced by equality.

The relationship between tables and constraint databases is explored in [13]. The relationship between tables and probabilistic databases, and between tables and data provenance is elegantly captured in the semiring framework of [7].

Key Applications

Three important applications of incomplete information will be discussed here. The first one is the problem of *view updates*. In this scenario, a view is defined by a query q, but the view is virtual, and only the database is materialized. As queries almost never are one-to-one functions, when a tuple t is inserted through a virtual view $q(r)$, it has to be translated to an insertion of a tuple, say $\hat{t}$, into r, s.t. $q(r \cup \{\hat{t}\}) = q(r) \cup \{t\}$. There is also a further requirement, that essentially guarantees the "minimality" of the change caused be the insertion of $\hat{t}$. Since there in general are several, sometimes infinitely many, insertions $\hat{t}$ that qualify, there actually is a set of possible databases $\{r \cup \{\hat{t}\} : \hat{t}\ \textit{qualifies}\}$. For example, for $q = R \bowtie S$, where $R(A, B)$ and $S(B, C)$, the deletion of a tuple (a, b, c) from a view $q(r, s)$, can be accomplished by either deleting the tuple (a, b) from r, or deleting the tuple (b, c) from s. This can easily be achieved if the database is stored as a conditional table, by simply replacing the tuples (a, b) in r, and (b, c) in s, with the conditional tuples (a, b); $x = 1$, and (b, c); $x \neq 1$, where x is an arbitrary fresh variable.

The views in the view update scenario are virtual. If views are materialized, they can be used to speed up query processing. The case where all views are materialized, and the database virtual is the classic information integration scenario. The views represent data sources, and the database schema represents the integrated schema that queries are formulated over. Intuitively, one can imagine that the integrated database exists, the view-defining queries are executed, and the views are accordingly populated. After this, the integrated database disappears. It is easy to see that a set of materialized views represent a set of possible databases, each satisfying the requirement that the current content of the views can be derived from it. Note that there is an open world assumption, as the requirement is that the view tuples are a subset (not necessarily proper) of $q(r)$, for all view definitions q and possible databases r. The closed world assumption would require equality, not just subset. For query answers, the certain answer is usually desired. The certain answer means in this context the set of tuples that are in the query result, for every possible database. To answer queries over the integrated schema, one can either rewrite the query in terms of the view-schemas, or reconstruct a representation of all the possible databases, and evaluate the original query on this representation. It perhaps speaks for the robustness of the concept of conditional tables, that they can do the job of representing the set of possible databases, whenever the views are queries in $\mathcal{Q}_{P^{\neq}}$, or in $\mathcal{Q}_{RA}$ if the closed world assumption is adopted [1,6].

The last application is that of *data exchange*. The setting is a peer-to-peer system, where each peer has a relational database and wants to exchange tuples with other peers. The schemas of any two peers, say p_1 and p_2 are usually different from each other, so the data has to be converted when exchanged. This is achieved by defining a mapping from p_1 to p_2. This mapping could for instance be expressed as an equation $q_1(p_1) \subseteq q_2(p_2)$. When we export data from p_1 to p_2, we evaluate q_1 on p_1, and make sure that the resulting tuples are in $q_2(p_2)$. Since the tuples have to be inserted into p_2, it is

easy to see that this is similar to the familiar problem of view updates, and thereby also incomplete information. For a very simple example, suppose the schema of p_1 is $R(A, B)$, and the schema of p_2 is $S(A, C)$. Consider then the equation $\pi_A(R) \subseteq \pi_A(S)$. If r contains a tuple (a, b), it is clear, that in order to satisfy the equation, a tuple (a, x) has to be in s. It has for example been shown [10], that if $q_1 \in \mathcal{Q}_P$ and q_2 only uses projection, then given null-free instances of p_1 and p_2, there is a naive table p_2', containing p_2, representing all minimal solutions to the equation, that is $q_1(p_1) \subseteq q_2(r)$, for all $r \in Rep(p'_2)$, and the equation is not satisfied by any proper subsets of these r's.

Future Directions

It is clear that new applications, new data models, and new query languages will encounter the problem of incomplete information. In order to not "re-invent the wheel," any solution should build on the foundation laid in [8]. A step in this direction has for instance been taken in [3].

Recommended Reading

1. Abiteboul S. and Duschka O.M. Complexity of answering queries using materialized views. In Proc. 17th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems, 1998, pp. 254–263.
2. Abiteboul S., Kanellakis P.C., and Grahne G. On the representation and querying of sets of possible worlds. Theor. Comput. Sci., 78(1):158–187, 1991.
3. Abiteboul S., Segoufin L., and Vianu V. Representing and querying XML with incomplete information. ACM Trans. Database Syst., 31(1):208–254, 2006.
4. Grahne G. The Problem of Incomplete Information in Relational Databases. Springer, 1991.
5. Grahne G. and Kiricenko V. Towards an algebraic theory of information integration. Inf. Comput., 194(2):79–100, 2004.
6. Grahne G. and Mendelzon A.O. Tableau Techniques for Querying Information Sources through Global Schemas. In Proc. 7th Int. Conf. on Database Theory, 1999, pp. 332–347.
7. Green T.J., Karvounarakis G., and Tannen V. Provenance semirings. In Proc. 26th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems, 2007, pp. 31–40.
8. Imielinski T. and Lipski W. Incomplete information in relational databases. J. ACM, 31(4):761–791, 1984.
9. Imielinski T. and Vadaparty K.V. Complexity of query processing in databases with OR-objects. In Proc. 8th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems, 1989, pp. 51–65.
10. Libkin L. Data exchange and incomplete information. In Proc. 25th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems, 2006, pp. 60–69.
11. Lipski W. Jr. On relational algebra with marked nulls. In Proc. 3rd ACM SIGACT-SIGMOD Symp. on Principles of Database Systems, 1984, pp. 201–203.
12. Reiter R. A sound and sometimes complete query evaluation algorithm for relational databases with null values. J. ACM. 33(2):349–370, 1986.
13. Revesz P.Z. Introduction to Constraint Databases. Springer, 2002.
14. Sarma A.D., Benjelloun O., Halevy A.Y., and Widom J. Working models for uncertain data. In Proc. 22nd Int. Conf. on Data Engineering, 2006, p. 7.
15. Vardi M.Y. Querying logical databases. J. Comput. Syst. Sci. 33 (2):142–160, 1986.
16. Vassiliou Y. Null values in data base management: a denotational semantics approach. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1979, pp. 162–169.
17. Zaniolo C. Database relations with null values. In Proc. 1st ACM SIGACT-SIGMOD Symp. on Principles of Database Systems, 1982, pp. 27–33.

Inconsistent Databases

Leopoldo Bertossi
Carleton University, Ottawa, ON, Canada

Definition

An inconsistent database is a database instance that does not satisfy those integrity constraints that have been declared together with the schema of the database.

Historical Background

Already in the classical and seminal paper by E. F. Codd [5] on the relational data model it is possible to find the notions of integrity constraint and consistency of a database. The idea of *consistent query answering*, consisting in characterizing and computing semantically correct answers to queries in inconsistent databases, was explicitly introduced in [1].

Foundations

A database can be seen as a model, i.e., as a simplified, abstract description, of an external reality. In the case of relational databases, one starts by choosing certain predicates of a prescribed arity. The *schema* of the database consists of this set of predicates, possibly *attributes*, which can be seen as names for the arguments of the predicates, together with an indication of the domains where the attributes can take their values. Having chosen the schema, the representation of the

external reality is given in terms of relations, which are extensions for the predicates in the schema. This set of relations is called an *instance* of the schema.

For example, relational database for representing information about students of a university might be based on the schema consisting of the predicates *Students(StNum, StName)* and *Enrollment(StName,Course)*. The attribute *StNum* is expected to take numerical values; *StName*, character string values; and *Course*, alphanumeric string values. In Fig. 1 there is a possible instance for this schema.

In order to make the database a more accurate model of the university domain (or to be in a more accurate correspondence with it), certain conditions are imposed on the possible instances of the database. Those conditions are intended to capture more meaning from the outside application domain. In consequence, these conditions are called *semantic constraints* or *integrity constraints* (ICs). For example, a condition could be that, in every instance, the student name functionally depends upon the student number, i.e., a student number is assigned to at most one student name. This condition, called a *functional dependency* (FD), is denoted with $StuNumber \rightarrow StuName$, or $Students : StuNumber \rightarrow StuName$, to indicate that this dependency should hold for attributes of relation *Students*. Actually, in this case, since all the attributes in the relation functionally depend on *StuNum*, the FD is called a *key constraint*.

Integrity constraints can be declared together with the schema, indicating that the instances for the schema should all satisfy the integrity constraints. For example, if the functional dependency $Students : StuNumber \rightarrow StuName$ is added to the schema, the instance in Fig. 1 is consistent, because it satisfies the FD. However, the instance in Fig. 2 is *inconsistent*. This is because this instance does not satisfy, or, what is the same, violates the functional dependency (the student number *101* is assigned to two different student names.

Functional dependencies are particular cases of integrity constraints. It is also possible to consider with the schema a *referential integrity constraint* that requires that every student (number) in the relation *Enrollment* appears, associated with a student name, in relation *Students*, the official "table" of students. This is denoted with $Enrollement[StNum] \subseteq Students[StNum]$. If this IC is considered in the schema, the instance in Fig. 1 is inconsistent, because student *105* does not appear in relation *Students*. However, if only the referential constraint were in the schema, the instance in Fig. 2 would be consistent.

In can be seen that the notion of consistency is relative to a set of integrity constraints. When a database is said to be inconsistent, it is meant that the particular instance of the database at hand is inconsistent.

The two particular kinds of integrity constraints presented above and also other forms of ICs can be easily expressed in the language of predicate logic. For example, the FD above can be expressed by the symbolic sentence

$$\forall x \forall y \forall z((Students(x, y) \wedge Students(x, z)) \rightarrow y = z), \quad (1)$$

Students	StuNum	StuName
	101	john bell
	102	mary stein
	104	claire stevens
	107	pat norton

Enrollment	StuNum	Course
	104	comp150
	101	comp100
	101	comp200
	105	comp120

Inconsistent Databases. Figure 1. A database instance.

Students	StuNum	StuName
	101	john bell
	101	joe logan
	104	claire stevens
	107	pat norton

Enrollment	StuNum	Course
	104	comp150
	101	comp100
	101	comp200

Inconsistent Databases. Figure 2. Another instance.

whereas the referential constraint above can be expressed by

$$\forall x \forall y (Enrollment(x, y) \rightarrow \exists z Students(x, z)). \quad (2)$$

Notice that this language of predicate logic is determined by the database schema, whose predicates are now being used to write down logical formulas. We may also use "built-in" predicates, like the equality predicate. Thus, ICs can be seen as forming a set Σ of sentences written in a language of predicate logic.

A database instance can be seen as an *interpretation structure* D for the language of predicate logic that is used to express ICs. This is because an instance has an underlying domain and (finite) extensions for the predicates in the schema. Having the database instance as an interpretation structure and the set of ICs as a set of symbolic sentences is crucial, and makes it possible to simply apply the notion of satisfaction of a formula by a structure of first-order predicate logic [6]. In this way, the notion of satisfaction of an integrity constraint by a database instance is a precisely defined notion. The database instance D is consistent if and only if it satisfies Σ, which is commonly denoted with $D \models \Sigma$.

Since it is usually assumed that the set of ICs is consistent as a set of logical sentences, in databases the notion of consistency becomes a condition on the database instance. Thus, this use of the term "consistency" differs from its use in logic, where consistency characterizes a set of formulas.

Inconsistency is an undesirable property for a database. In consequence, one attempts to keep it consistent as it is subject to updates. There are a few ways to achieve this goal. One of them consists in declaring the ICs together with the schema, and the database management system (DBMS) will take care of the database maintenance, i.e., of keeping it consistent. This is done by rejecting transactions that may lead to a violation of the ICs. For example, the DBMS should reject the insertion of the tuple (*101*, *sue jones*) into the instance in Fig. 1 if the FD (1) was declared with the schema (as a key constraint). Unfortunately, commercial DBMSs offer limited support for this kind of database maintenance.

An alternative way of keeping consistency is based on the use of triggers (or active rules) that are stored in the database. The reaction to a potential violation is programmed as the action of the trigger: if a violation is about to be produced or is produced, the trigger automatically reacts, and its action may reject the violating transaction or compensate it with additional updates, to make sure that at the end, consistency is reestablished. Consistency can also be enforced through the application programs that interact with the DBMS. However, the correctness of triggers or application programs with respect to (with regard to) ensuring database consistency is not guaranteed by the DBMS.

It is the case that, for whatever reasons, databases may become inconsistent, i.e., they may violate certain ICs that are considered to be relevant to maintain for a certain application domain. This can be due to several reasons, e.g., poorly designed or implemented applications that fail to maintain the consistency of the database, or ICs for which a DBMS does not offer any kind of support, or ICs that are not enforced for better performance of application programs or DBMSs, or ICs that are just assumed to be satisfied based on knowledge about the application domain and the kind of updates on the database. It is also possible to have a legacy database on which semantic constraints have to be imposed; or more generally, a database on which imposing new constrains depending on specific needs, e.g., user constraints, becomes necessary.

In the area of data integration the satisfaction of desirable ICs by a database is much more difficult to achieve. One can have different autonomous databases that are separately consistent with regard to their own, local ICs. However, when their data is integrated into a single database, either material or virtual, certain desirable global ICs may not be satisfied. For example, two university databases may use the same numbers for students. If their data is put together into an integrated database, a student number might be assigned to two different students.

When trying to use an inconsistent database, the application of some *data cleaning* techniques may be attempted, to cleanse the database from data that participates in the violation of the ICs. This is done sometimes. However, data cleaning is a complex and non-deterministic process; and it may also lead to the loss of information that might be useful. Furthermore, in certain cases like virtual data integration, where the data stays at the autonomous data sources, there is no way to change the data without ownership of the sources.

One might try to live with inconsistent databases. Actually, most likely one will be forced to keep using it, because there is still useful information in it. It is also likely that most of the information in it is

Students1	StuNum	StuName
	101	john bell
	104	claire stevens
	107	pat norton

Students2	StuNum	StuName
	101	joe logan
	104	claire stevens
	107	pat norton

Inconsistent Databases. Figure 3. Two repairs.

somehow consistent. Thus, the challenge consists in retrieving from the database only information that is consistent. For example, one could pose queries to the database at hand, but expecting to obtain only answers that are semantically correct, i.e., that are consistent with the ICs. This is the problem of *consistent query answering* (CQA).

The notion of consistency of a database is a holistic notion, that applies to the entire database, and not to portions of it. In consequence, in order to pursue this idea of retrieving consistent query answers, it becomes necessary to characterize the consistent data in an inconsistent database first. The idea that was proposed in [1] is as follows: the consistent data in an inconsistent data is the one that is invariant under all possible way of restoring the consistency by performing minimal changes on the initial database. That is, no matter what minimal consistency restoration process is applied to the database, the consistent data stays in the database. Each of the consistent versions of the original instance obtained by minimal changes is called a *minimal repair*, or simply, a *repair*.

It becomes necessary to be more precise about the meaning of minimal change. In between, a few notions have been proposed and studied (cf. [2–4] for surveys of CQA). Which notion to use may depend on the application. The notion of minimal change can be illustrated using the definition of repair given in [1]. First of all, a database instance D can be seen as a finite set of ground atoms (or database tuples) of the form $P(\bar{c})$, where P is a predicate in the schema, and $\bar{c}$ is a finite sequence of constants in the database domain. For example, *Students(101, john bell)* is an atom in the database. Next, it is possible to compare the original database instance D with any other database instance D' (of the same schema) through their symmetric difference $D\Delta D' = \{A \mid A \in (D \setminus D') \cup (D' \setminus D)\}$.

Now, a repair of an instance D with regard to a set of ICs Σ is defined as an instance D' that is consistent, i.e., $D' \models \Sigma$, and for which there is no other consistent instance D'' that is closer to D than D', i.e., for which it holds $D\Delta D'' \subsetneq D\Delta D'$. For example, the database in Fig. 2 has two repairs with regard to the FD (1). They are shown in Fig. 3 and are obtained each by deleting one of the two conflicting tuples in relation *Students* (relation *Enrollment* does not change).

Having defined the notion of repair, a *consistent answer* from an instance D to a query $Q(\bar{x})$ with regard to a set Σ of ICs is defined as an answer $\bar{c}$ to Q that is obtained from every possible repair of D with regard to Σ. That is, if the query Q is posed to each of the repairs, $\bar{c}$ will be returned as a usual answer to Q from each of them.

For example, if the query $Q_1(x, y) : Students(x, y)$, asking for the tuples in relation *Students*, is posed to the instance in Fig. 2, then *(104, claire stevens)* and *(107, pat norton)* should be the only consistent answers wrt the FD (1). Those are the tuples that are shared by the extensions of *Students* in the two repairs. Now, for the query $Q_2(x) : \exists y Students(x, y)$, i.e., the projection on the first attribute of relation *Students*, the consistent answers are *(101)*, *(104)* and *(107)*.

There might be a large number of repairs for an inconsistent database. In consequence, it is desirable to come up with computational methodologies to retrieve consistent answers that use only the original database, in spite of its inconsistency. Such a methodology that works for particular syntactic classes of queries and ICs, was proposed in [1]. The idea is to take the original query Q that expects consistent answers, and syntactically transform it into a new query Q', such that the *rewritten query* Q', when posed to the original database, obtains as usual answers the consistent answers to query Q. The essential question is, depending on the language in which Q is expressed, what kind of language is necessary for expressing the rewriting Q'. The answer to this question should also depend on the kind of ICs being considered.

The idea behind the rewriting approach presented in [1] can be illustrated by means of an example. The consistent answers to the query $Q_1(x, y) : Students(x, y)$ above with regard to the FD (1) can be obtained by posing the query $Q'(x, y) : Students(x, y) \wedge \neg\exists z (Students(x, z) \wedge z \neq y)$ to the database. The new query collects as normal answers those tuples where the value of the first attribute is not associated to two

different values of the second attribute in the relation. It can be seen that the answer set for the new query can be computed in polynomial time in the size of the database.

In this example, a query expressed in first-order predicate logic was rewritten into a new query expressed in the same language. It has been established in the literature that, for complexity-theoretic reasons, a more expressive language to do the rewriting of a first-order query may be necessary. For example, it may be necessary to do the rewritings as queries written in expressive extensions of Datalog [2–4].

If a database is inconsistent wrt referential ICs, like the instance in Fig. 1 and the constraint in (2), it is natural to restore consistency by deleting tuples or inserting tuples containing *null values* for the existentially quantified variables in the ICs. For example, the tuple (*105*, *comp120*) could be deleted from *Enrollment* or the tuple (*105*, *null*) could be inserted in relation *Students*. This requires a modification of the notion of repair and a precise semantics for satisfaction of ICs in the presence of null values [2,4].

Key Applications

Key applications of *consistent query answering* (CQA) are still missing. Applications to virtual data integration look promising, and also applications to data cleaning.

Future Directions

There are many open problems and research directions, among them, and most prominently, the development of key applications of CQA. A more precise characterization of the languages that are needed for doing CQA using query rewriting is also missing. It also becomes necessary to shed more light on the right kind of repair semantics to use depending on the application. CQA in a dynamic setting, when the databases is subject to updates, has not been investigated much. Integrity constraints and consistency issues for the relational model of data have been investigated for many years. However, there are other data models, e.g., spatial databases, for which much research of this kind is still necessary.

Cross-references

- ▶ Active Databases
- ▶ Data Cleaning
- ▶ Logical Data Integration
- ▶ Logics and Databases
- ▶ Null Values
- ▶ Relational Theory

Recommended Reading

1. Arenas M., Bertossi L. and Chomicki J. Consistent query answers in inconsistent databases. In Proc. 18th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems, 1999, pp. 68–79.
2. Bertossi L. Consistent query answering in databases. ACM SIGMOD Rec., 35(2):68–76, 2006.
3. Bertossi L. and Chomicki J. Query answering in inconsistent databases. In Logics for Emerging Applications of Databases. Springer, Berlin, 2003, pp. 43–83.
4. Chomicki J. Consistent query answering: five easy pieces. In Proc. 11th Int. Conf. on Database Theory, 2007, pp. 1–17.
5. Codd E.F. A relational model of data for large shared data banks. Commun. ACM, 13(6):377–387, 1970.
6. Enderton H. A Mathematical Introduction to Logic, 2nd edn. Academic Press, New York, NY, USA, 2001.

Incremental Computation of Queries

Guozhu Dong[1], Jianwen Su[2]
[1]Wright State University, Dayton, OH, USA
[2]University of California-Santa Barbara, Santa Barbara, CA, USA

Synonyms

Incremental view maintenance

Definition

A view on a database is defined by a query over the database. When the database is updated, the value of the view (namely the answer to the query) will likely change. The computation of the new answer to the query using the old answer is called *incremental query computation* or *incremental view maintenance*. Incremental computation is typically performed by identifying the part in the old answer that need to be removed, and the part in the new answer that need to be added. Incremental computation is desirable when it is much more efficient than a re-computation of the query. Efficiency can be measured by computation time, storage space, or query language desirability/availability, etc. Incremental computation algorithms could use auxiliary relations (in addition to the query answer), which also need to be incrementally computed.

Two query languages can be involved for the incremental query computation problem. One is used for defining the view to be maintained, and the other for describing the incremental computation algorithm. For relational databases, the two languages can be relational algebra, SQL, nested relational algebra, Datalog, SQL embedded in a host programming language, etc.

Historical Background

Reference [14] was an early paper on incremental query computation of relational algebra queries. Reference [8] improved the query rewriting algorithm of [14] to ensure minimality of the changes to a query result in response to updates to base relations. Other articles in the literature (see [9]) considered the incremental computation of queries involving SQL aggregation, duplicates, etc.

Reference [7] was an early paper on incremental query computation of recursive (Datalog, including transitive closure) queries using relational calculus (or equivalently relational algebra) queries. Reference [3] provided relational-calculus queries for maintaining the transitive closure of acyclic graph (binary relation) after an edge insertion or deletion. Reference [6] discussed how to incrementally compute Datalog queries after tuple insertion. References [4,13] provided relation-calculus queries for maintaining the transitive closure of undirected graph after edge insertion or deletion. Reference [1] rewrote the maintenance queries in SQL. Reference [12] provided relational-calculus queries for maintaining the shortest paths in undirected graphs after edge insertion/deletion. Reference [11] studied the power of incremental query computation using SQL as maintenance queries. A number of theoretical results on the power of the arity of the auxiliary relations have been reported. Reference [5] gave a survey of results on the incremental computation of recursive queries up to the year 2000.

Reference [9] contains a collection of papers on incremental query computation/view maintenance.

Foundations

For brevity and concreteness, the discussion below will be restricted to relational databases, although incremental query computation also applies to other kind of databases. Moreover, the discussion is divided into two parts, one part for the incremental computation of relational-algebra queries, and the other part for the recursive queries (including Datalog and transitive closure). Algorithms for incrementally computing other kinds of queries are not covered in this entry.

Incremental query computation is not merely a means to avoid expensive re-computation. Sometimes, incremental computation is a way to do things that could not be done otherwise. For example, the transitive closure query cannot be directly expressed in relational algebra but it can be expressed in relational algebra in the incremental setting.

Some notations are now defined. Let $R_1,...,R_m$ be the base relations of a relational database D. An update to the database consists of $2m$ sets of tuples: for each relation R_i it contains two disjoint sets, R_i^+ and R_i^-, of tuples, where R_i^+ is the set of tuples to be added to R_i and R_i^- is the set of tuples to be deleted from R_i. Let D^{old} denote the old database and R_i^{old} denote the old instance of R_i before the update, and D^{new} denote the new database and R_i^{new} denote the new instance of R_i after the update. It is assumed that the update is minimal in the sense that $R_i^+ \cap R^{\text{old}} = \emptyset$ and that $R_i^- \subseteq R^{\text{old}}$.

The same notation will be used to denote the "update to the query." Let Q be a query. Then Q^- is used to denote the set of tuples to be deleted from the old answer, and Q^+ is used to denote the set of tuples to be added to the old answer, in order to get the new answer; in formula: $Q(D^{\text{new}}) = (Q(D^{\text{old}}) - Q^-) \cup Q^+$.

Incremental Computation of Relational-Algebra Queries

The relational algebra queries are expressions built from relation names using the following operations: selection (σ_p), projection (Π_A), Cartesian product ($\times$), union ($\cup$), intersection ($\cap$), difference ($-$), and join ($\bowtie$). Here p is a predicate or condition, and A is a set of attributes. A relational-algebra query over the database is a relational algebra expression.

The 16 rules [8] in Table 1 can be used to generate the queries for computing Q^+ and Q^-. These rules also ensure that the "update to the query" is minimal in the sense that $Q^+ \cap Q^{\text{old}} = \emptyset$ and $Q^- \subseteq Q^{\text{old}}$.

Observe that one also needs to store the answer to all intermediate queries in order to use the rules; these are auxiliary relations that also need to be maintained.

Incremental Computation of Recursive Queries

The framework used in the incremental computation of recursive queries is quite similar to the one for relational algebra queries as discussed above. However, there are two notable differences. One is that the

Incremental Computation of Queries. Table 1. Rules for incremental computation of relational algebra queries

Query Q	Rewriting Rule for Getting Q^-	Query Q	Rewriting Rule for Getting Q^+
R_i	R_i^-	R_i	R_i^+
$\sigma_p(S)$	$\sigma_p(S^-)$	$\sigma_p(S)$	$\sigma_p(S^+)$
$\Pi_A(S)$	$\Pi_A(S^-) - \Pi_A(S^{new})$	$\Pi_A(S)$	$\Pi_A(S^+) - \Pi_A(S^{old})$
$S \times T$	$(S^- \times T^{old}) \cup (S^{old} \times T^-)$	$S \times T$	$(S^+ \times T^{new}) \cup (S^{new} \times T^+)$
$S \cup T$	$(S^- - T^{new}) \cup (T^- - S^{new})$	$S \cup T$	$(S^+ - T^{old}) \cup (T^+ - S^{old})$
$S \cap T$	$(S^- \cap T^{old}) \cup (S^{old} \cap T^-)$	$S \cap T$	$(S^+ \cap T^{new}) \cup (S^{new} \cap T^+)$
$S - T$	$(S^- - T^{old}) \cup (S^{old} \cap T^+)$	$S - T$	$(S^+ - T^{new}) \cup ((S^{old} - S^-) \cap T^-)$
$S \bowtie T$	$(S^- \bowtie T^{old}) \cup (S^{old} \bowtie T^-)$	$S \bowtie T$	$(S^+ \bowtie T^{new}) \cup (S^{new} \bowtie T^+)$

query definition and the incremental computation may use different query languages, and the other is that the incremental computation may use auxiliary relations that are not explicitly involved in the original query.

Maintenance algorithms can be specified in different (maintenance) languages such as relational calculus, SQL, nested relational algebra, Datalog, or even a host programming language. The choice of a maintenance language can be influenced by the practical constraints imposed by real systems, and it can also be influenced by efficiency issues, as some languages are more efficient or more optimizing than others. For example, it is desirable to maintain the transitive closure query using relational algebra (or equivalently, first-order logic) queries, since the latter are available in all relational database systems and they are more efficient to evaluate. Another desirable language for maintaining recursive queries is SQL.

With the help of auxiliary relations, one can maintain queries that cannot be maintained otherwise. Moreover, the arity of the auxiliary relations bounds the amount of information to be kept and makes a difference on what can be maintained.

Sometimes the incremental computation algorithms can deal with certain special types of updates. Example types of updates include single-tuple insertions, single-tuple deletions, insertions/deletions of sets satisfying certain conditions, or combinations of these.

The following describes the relational algebra (or first-order logic) queries for maintaining the transitive closure of acyclic graphs [2,3]. The incremental queries can handle both tuple insertions and tuple deletions, and they do not use auxiliary relations.

Let G represent the input graph (directed) and TC the transitive closure of G. So a tuple (x,y) is in the relation G if and only if there is a directed edge from the node x to the node y in the input graph, and a tuple (x,y) is in the relation TC if and only if there is a directed path from the node x to the node y in the input graph. An edge insertion is allowed only if this insertion does not lead to cycles in the new graph.

Suppose an edge (a,b) is inserted. Then TC is maintained as follows. Essentially, the new transitive closure is obtained by adding to the old transitive closure the following: (i) all new paths constructed by adding the new edge (a,b) to the back of existing paths ending at a, (ii) all new paths constructed by adding the new edge (a,b) to the front of existing paths starting at b, (iii) all new paths constructed by inserting the new edge (a,b) between an existing path ending at a and an existing path starting at b, and (iv) the new edge itself. New paths added by rules (1), (2), and (3) correspond to paths of type $x \rightarrow a$, $b \rightarrow y$, and $x \rightarrow y$, respectively. These rules cover all new paths because only one occurrence of the new edge is necessary in every new path.

Suppose an existing edge (a,b) is deleted. TC can be maintained as follows. Let $S_{ab} = \{(x,y) | TC^{old}(x,a) \wedge TC^{old}(b,y)\}$ be the set of all paths (x,y) in the old TC which go through (a,b). The letter S is for *suspicious* – it is doubtful whether these paths should belong to the new TC. Let $G^{new} = G^{old} - \{(a,b)\}$ and $T_{ab} = (TC^{old} - S_{ab}) \cup G^{new}$. Each pair in T_{ab} is definitely in the new TC. (The letter T is for *trusty.*) Surprisingly, the new TC can be completely reconstructed from T_{ab} using several joins and projections given by the following formula:

$$T_{ab} \cup (T_{ab} \circ T_{ab}) \cup (T_{ab} \circ T_{ab} \circ T_{ab})$$

where $R_1 \circ R_2$ is defined as $\{(x,y) | \exists u(R_1(x,u) \wedge R_2(u,y)\}$ and R_1 and R_2 are binary relations.

So the new transitive closure contains (i) all *trusty* paths, (ii) all paths constructed by concatenating two consecutive *trusty* paths, and (iii) all paths constructed by concatenating three consecutive *trusty* paths.

To maintain the transitive closure of undirected graphs in relational calculus after edge deletion, one also needs to maintain some auxiliary queries. The auxiliary relations include a total order on the nodes of the graph, a spanning forest of the graph, and a ternary relation indicating whether a node is on a path between two other nodes in the spanning forest. The details can be found in [4,5], which also discuss other queries.

While the maintenance queries for the transitive closure of undirected graphs and acyclic graphs do not need to use arithmetic operations, the maintenance queries for shortest paths [12] need to use both + and $<$ on numbers.

Key Applications

Incremental query computation is useful for (i) maintaining materialized views, (ii) efficient checking and monitoring of integrity constraints, and (iii) the efficient management of triggers in active databases. Incremental query computation is also related to dynamic descriptive complexity theory [10]. Incremental computation of transitive closure has also been used in formal verification.

Future Directions

It is still open whether the transitive closure of arbitrary directed graphs can be maintained using relational calculus queries after edge deletions.

Cross-references

- ► Active Databases
- ► Database Trigger
- ► FOL modeling of integrity constraints (dependencies)
- ► Incremental View Maintenance

Recommended Reading

1. Dong G., Libkin L., Su J., and Wong L. Maintaining transitive closure of graphs in SQL. Int. J. Inf. Technol., 5(1):46–78, 1999.
2. Dong G. and Pang C. Maintaining transitive closure in first-order after node-set and edge-set deletions. Inf. Process. Lett., 62 (3):193–199, 1997.
3. Dong G. and Su J. Incremental and decremental evaluation of transitive closure by first-order queries. Inf. Comput., 120(1):101–106, July 1995.
4. Dong G. and Su. J. Arity bounds in first-order incremental evaluation and definition of polynomial time database queries. J. Comput. Syst. Sci., 57(3):289–308, December 1998.
5. Dong G. and Su J. Incremental maintenance of recursive views using relational calculus/SQL. ACM SIGMOD Rec., 29 (1):44–51, 2000.
6. Dong G., Su J., and Topor R. Nonrecursive incremental evaluation of datalog queries. Annals Math. Artif. Intell., 14:187–223, 1995.
7. Dong G. and Topor. R. Incremental evaluation of datalog queries. In Proc. 4th Int. Conf. on Database Theory, 1992, pp. 282–296.
8. Griffin T., Libkin L., and Trickey H. An improved algorithm for the incremental recomputation of active relational expressions. IEEE Trans. Knowl. Data Eng., 9(3):508–511, 1997.
9. Gupta A. and Mumick I.S. (eds.). Materialized views: techniques, implementations, and applications. MIT, MA, USA, 1999.
10. Immerman N. Descriptive Complexity. Springer, New York, NY, USA, December 1998.
11. Libkin L. and Wong L. On the power of incremental evaluation in SQL-like languages. In Proc. 7th Int. Workshop on Database Programming Languages, 1999, pp. 17–30. See also: SQL can maintain polynomial-hierarchy queries. Technical report, Institute of Systems Science Singapore, 1997.
12. Pang C., Dong G., and Kotagiri R. Incremental maintenance of shortest distance and transitive closure in first-order logic and SQL. ACM Trans. Database Syst., 30(3):698–721, 2005.
13. Patnaik S. and Immerman N. Dyn-FO: a parallel dynamic complexity class. J. Comput. Syst. Sci., 55(2):199–209, Oct 1997.
14. Qian X. and Wiederhold G. Incremental recomputation of active relational expressions IEEE Trans. Knowl. Data Eng., 3(3):337–341, 1991.

Incremental Crawling

Kevin S. McCurley
Google Research, Mountain View, CA, USA

Synonyms

Spidering; Crawler

Definition

Part of the success of the World Wide Web arises from its lack of central control, because it allows every owner of a computer to contribute to a universally shared information space. The size and lack of central control presents a challenge for any global calculations that operate on the web as a distributed database. The scalability issue is typically handled by creating a central repository of web pages that is optimized for large-scale calculations. The process of creating this repository

consists of maintaining a data structure of URLs to fetch, from which URLs are selected, the content is fetched, and the repository is updated. This process is called *crawling* or *spidering*.

Unfortunately, maintaining a consistent shadow repository is complicated by the dynamic and uncoordinated nature of the web. URLs are constantly being created or destroyed, and contents of URLs may change without notice. As a result, there will always be URLs for which the content is not present in the repository, as well as URLs whose content is different from the copy in the repository. Many new URLs can only be discovered by recrawling old URLs whose content has now changed to include links to new URLs. In order to minimize the impact of these inconsistencies, URLs should periodically be prioritized and revisited. The process of prioritizing and revisiting URLs is usually referred to as *incremental crawling*. The primary issues in incremental crawling center around defining metrics for performance, both for the quality of the repository and the resources required to build and maintain the repository.

Historical Background

In the early days of the world wide web, it quickly became apparent that documents could be treated as living objects that could be changed at will. Thus the notion of a URL was born as a "resource locator" rather than a document ID. Early attempts to build search engines largely ignored this problem, assuming that if a document was worth retrieving later, then it would remain relatively stable at its URL. Thus early attempts to crawl and index the web simply started with a set of seed URLs, and iteratively crawled pages and extracted new URLs to be crawled. After a period of time, this repository would be used as a "snapshot" of the web to build a keyword index. If a reasonable seed set is used and URLs are appropriately prioritized, the resulting snapshot was useful for constructing a search engine or performing aggregate analysis.

The snapshot approach works well for web pages that are created and remain unchanged, but web usage has evolved quite a bit since the early days. Web authors learned that fresh content would bring repeat readers, and readers have increasingly migrated toward entertainment and news, for which freshness is increasingly important. Thus in some sense the more dynamic parts of the web are those that have the highest readership and therefore the greatest social impact. As a result, incremental crawling strategy has become increasingly important.

Foundations

In order to address the issues surrounding incremental crawling, it is important to first define the goals of the shadow repository. The primary application that has been implemented thus far have been search engines that allows users to find pages that match simply expressed information needs. A prerequisite for success of a search engine is that it reference the best content that is of interest to a majority of users at any given time. For this, the repository should be as complete as possible and as fresh as possible, since pointing users at pages that don't fulfill their information need will result in a bad user experience.

Another application that can use a shadow repository is the discovery of plagarism or copyright-protected content on the web. A third application might seek to track historical evolution of the web, for historical studies. In this case the repository should not only contain the most recent content for a URL, but all versions as they evolve over time. It is easy to see that performance metrics for these applications will vary somewhat. Moreover, these applications will differ in the the degree of cooperation that can be expected from contributors to the web.

Metrics for Incremental Crawling

There is no uniformly accepted notion of how to measure the inconsistency between the repository and the actual web. Moreover, even if there was universal agreement on the proper metric for consistency, it may be difficult to measure, and any such measure would be time-dependent since the web is constantly changing. Three possible metrics that are most obvious are:

Coverage: count the number of pages that exist on the web but are not present in the respository at any given time.

Freshness: count the number of documents that are present in the repository, but whose content was changed after insertion into the repository. Alternatively, the metric may incorporate a measure of the size of the change for individual pages.

Age: the average age of documents in the repository (e.g., time since they were last updated).

For any particular application, these metrics may all be adjusted to incorporate weights of individual pages to

reflect their relative importance for the application. Thus for a search engine, documents that are dominated by thousands of other documents for every possible query should probably should be weighted lower than documents that are often found by users of the search engine. The quality of an incremental crawling strategy must also be evaluated on the basis of resources that it consumes in order to maintain a given level of metric.

Incentives and Cooperation

It should be noted that a crawler consumes resources both of the party maintaining the repository as well as the creators of content on the web. If a crawler becomes too aggressive in tracking the changes to URLs, it could quickly overwhelm small sites and degrade the value of the web site to the users of the web. At the same time, it is relatively simple to create a web site that issues dynamic content in response to any request by a crawler, essentially creating an excess of URLs that could theoretically be fetched, archived, and indexed by the crawler. Web sites may also engage in other forms of mischief, such as accepting connections and either responding slowly or with noncomformant content.

Thus both content providers and repository maintainers are threatened by potentially unconstrained resource requirements. As the web has evolved, a number of standards and standard practices have emerged that mitigate these threats, and a form of economic equilibrium has emerged to allow repositories to be maintained.

The situation is complicated by the fact that the goals of content producers and crawlers are sometimes not aligned to each other. Content providers are generally motivated by the desire to shape public opinion through their information, or their desire to drive traffic for commerce. In a competitive market, multiple content providers may find themselves in competition with each other, competing for the attention of humans. As such, they can be expected to engage in a variety of practices that will improve their share of readership.

By contrast, consider the goals of a commercial search engine. Search engines make money from advertising, and in order to maintain the traffic, they need to serve the aggregated needs of users. Even if the search engine had a completely accurate copy of the web, there is still the task of deciding which results to show, and this is where the goals of individual content providers may conflict with those of the search engine (and users). Content providers generally want their content to be indexed, but they also want their content to be shown to users ahead of their competitors. They can be expected to act in their own self interest.

One of the activities that some sites engage in is to create link farms, which are designed to enhance the ranking of pages in search engines. In order for this to be effective, the pages of link farms must be crawled and incorporated into the ranking of the search engine. This is an extreme example of the more general phenomenon that content providers and search engines may disagree on what should be crawled and indexed.

Cache Consistency

Inconsistency of a web shadow repository is similar to any other cache consistency problem, and has been studied extensively in the literature. Traditional approaches to cache consistency include time-to-live (TTL), client polling, and invalidation protocols. Most of the literature on caching has neglected the issue of discovery, which is a critical feature of crawling and one reason why pages need to be revisited (to find new links). Due to the uncoordinated nature of the web, most of the effort has gone into client polling approaches, although elements of the other approaches have also been experimented with. The HTTP protocol [5, section 13] contains a variety of optional features that can help with TTL approaches if they are properly implemented. Examples include the `Cache-Control`, `Expires`, `Last-Modified`, `If-Modified-Since`, `If-Unmodified-Since`, and `Vary` header fields. The cache-consistency protocol of the HTTP protocol is designed to facilitate human interactive browsing, and is not designed for batch processing. Moreover, relatively few web sites have taken the care to implement the existing protocols correctly.

In 2005 Google published an improved mechanism for conveying TTL cache consistency messages, known as sitemaps [7]. This followed an earlier effort called Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH), and was released under a creative commons license. The purpose of the sitemaps initiative was to provide a more efficient polling mechanism, where web sites could provide information in an XML format to specify a list of valid URLs for crawling, along with optional last-modified dates, change frequencies, and priorities. Theoretically this scales up to sites with fifty million URLs, but has only weak consistency guarantees incorporated into it. In addition to conveying

information about expiration and priority, the protocol also specifies a way for sites to notify individual search engines of the existence of their sitemap, either through placement in a robots.txt file or through a ping notification message [7]. Sitemaps can also be useful in eliminating intra-site duplicates, which consume resources of search engines and degrade the quality of indexing of sites.

In theory, sitemaps can greatly improve the efficiency of maintaining a shadow repository, but they have their own problems due to previously mentinoed misalignment of incentives between content providers and search engines. Sitemaps are subject to various kinds of abuse, and cannot be completely trusted by the crawler to give an accurate view of a web site.

Resource Management

The resources consumed by incremental crawling can be significant. In one study in 2002 [9], it was estimated that 40% of Internet traffic is due to web crawlers retrieving pages. Similar figures have been observed by others, though the impact is probably higher for low-traffic sites than it is for high-traffic sites. From the point of view of the crawler, it is not obvious how to optimally design a crawl strategy in order to achieve a given level of one of a metrics such as freshness.

One tempting strategy is to adjust the crawl frequency for a page in proportion to the rate of change for the page. Thus if a page changes freqently and substantially, it would receive a higher level of crawling. It is perhaps counterintuitive that this may not be optimal. In the extreme case, a page may in fact change every time it is accessed, which means that no matter how often you access it, you will never have the "up to date copy." Accessing a page too frequently would also violate the standard "politeness" policy of web crawlers.

Another potential policy would be to revisit pages with the same frequency, ignoring the change rate of individual pages. This is evidently wasteful of resources, but Cho and Garcia-Molina [1] showed both experimentally and theoretically that this outperforms the proportional approach. The optimal strategy interpolates between the two, with a visitation schedule that increases monotonically with the change rate, but penalizes pages that change too often.

Theoretical results of this type depend upon an underlying mathematical model of how the web changes, and should be carefully examined when applying it to a restricted subset of the web, and should be re-evaluated in the future should the social forces that shape the web somehow change. Such models should reflect the rate at which new information is produced, the distribution of such production among individual websites, and the rate at which information decays or is revised. Such models are essential for deciding how to balance the resources dedicated to discovery of new content vs. the confirmation that existing content has not changed.

Key Applications

To date, the primary application that has made use of crawling is search engines. The details of their incremental crawling strategies remain unpublished, as they are based on the perceived economic value of the repository. As the web has grown, search engines have become increasingly important for users, allowing them to express an information need in fairly precise terms, and quickly navigate directly to documents on the topic of interest. From the point of view of a search engine, the dynamic nature of the web presents special problems, since a search engine will typically depend upon deep analysis of documents and links between documents. Documents whose content changes on a regular basis present a challenge for indexing, since the freshness of a search engine is a critical measure of utility for users.

Search engines are not the only applications that can make use of incremental crawling. Others include systems for prefetching proxies, notifications and alerts, mirroring and archiving, and business and political intelligence.

Future Directions

Most of the current strategies for incremental crawling are heavily dependent upon polling, and are therefore fairly inefficient. An approach built around notification and invalidation would clearly be more efficient, but there appears to be little economic incentive to implement them. Existing mathematical models for the growth and change rates of the web are fairly simplistic, treating all web pages as equal and failing to recognize the inherent organizational structure of web sites and the different purposes for which pages are created. Finally, as new applications emerge for web shadow repositories, it is natural to expect new requirements to emerge.

Experimental Results

See [1].

Cross-references

- ▶ Caching
- ▶ Data Broadcasting, Caching and Replication in Mobile Computing
- ▶ Focused Web Crawling
- ▶ Replication
- ▶ Web Crawler Architecture

Recommended Reading

1. Cho J. and Garcia-Molina H. Effective page refresh policies for web crawlers. ACM Trans. Database Syst., 28(4):390–426, 2003.
2. Coffman E.G., Liu Z., and Weber R.R. Optimal robot scheduling for Web search engines. J. Schedul., 1:15–29, 1998.
3. Dikaiakos M.D., Stassopoulou A., and Papageorgiou L. An investigation of web crawler behavior: characterization and metrics. Comput. Commun., 28:880–897, 2005.
4. Edwards J., McCurley K.S., and Tomlin J. An adaptive model for optimizing performance of an incremental web crawler. In Proc. 10th Int. World Wide Web Conference, 2001, pp. 106–113.
5. Fielding R., Gettys J., Mogul J., Frystyk H., Mastinter L., Leach P., and Berners-Lee T. Hypertext Transfer Protocol – HTTP/1.1, RFC 2616 http://www.w3.org/Protocols/rfc2616/rfc2616.html.
6. Podlipnig S. and Böszörmenyi L. A survey of Web cache replacement strategies, ACM Comput. Surveys, 35(4): 374–398, 2003.
7. Sitemap protocol specification. http://www.sitemaps.org/protocol.php.
8. Wang J. A survey of web caching schemes for the Internet. ACM SIGCOMM Comput. Commun. Rev., 29(5): 36–46, 1999.
9. Yuan X., MacGregor M.H., and Harms J. An efficient scheme to remove crawler traffic from the Internet. In Proc. 11th Int. Conf. on Computer Communications and Networks, 2002, pp. 90–95.

Incremental Maintenance of Views with Aggregates

Himanshu Gupta
Stony Brook University, Stony Brook, NY, USA

Definition

Views are SQL or relational expressions over the given data sources. In a data warehouse, view expression generally involve the aggregate operator. In order to keep *materialized* (precomputed and stored) views up to date, it is necessary to maintain the views in response to the changes at the sources. *Incremental maintenance* of a view involves propagating the changes at the source onto the view so that the view reflects the changes. Incrementally maintaining a view can be significantly cheaper than recomputing the view from scratch.

Historical Background

Incrementally maintaining a view can be significantly cheaper than recomputing the view from scratch, especially if the size of the view is large compared to the size of the changes [1,2,9]. The problem of incremental maintenance of views has been studied extensively, and several algorithms have been proposed over the years [3,4,6–12]. Most works (except for [7,12]) either handled view expressions without aggregate operators [3,4,11], or *aggregate views* (view expressions with the aggregate operator as the last operator) [8,9,10]. Quass in [12] attempts to maintain general view expressions involving aggregate operators, but the expressions obtained are inefficient and very complicated. Recently, Gupta and Mumick [7] developed the change-table technique for incremental maintenance of general view expressions involving aggregate and outerjoin operators. Change table of a particular view is applied to the view using a special refresh operator. The change-table technique can be looked upon as a generalization of the technique of summary tables developed by [9] for maintenance of aggregate views. In contrast to the most other techniques which propagate data in terms of insertions and deletions through a view expression, the change-table technique propagates data (in terms of change-tables) as well as action (in terms of parameters of the refresh operation) through the given view expression.

Foundations

The first step is to explain the framework developed in [3,11] for deriving incremental view maintenance expressions and relate it to the change-table technique. Let a database contain a set of relations $\mathcal{R} = \{R_1, R_2,...,R_n\}$. A *change transaction* t is defined to contain the expression $R_i \leftarrow (R_i \dot{-} \nabla R_i) \uplus \Delta R_i$, for each relation R_i, where ∇R_i are the deletions from R_i, and ΔR_i are the insertions into R_i. Let V be a bag-algebra expression defined on a subset of the relations in $\mathcal{R}$. The *refresh-expression* $New(V, t)$ ([3] uses the notation $pre(t, V)$ instead) is used to compute the new value of V. Griffin and Libkin in [3] define the expression $New(V, t)$ to be:

$$New(V, t) = (V \dot{-} \nabla(V, t)) \uplus \Delta(V, t).$$

So, the goal in deriving view maintenance expressions for a view V is to derive two functions $\nabla(V, t)$ and $\Delta(V, t)$ such that for any transaction t, the view V can

be maintained by evaluating $(V \dot{-} \nabla(V,t)) \uplus \Delta(V, t)$. In order to derive $\nabla(V, t)$ and $\Delta(V, t)$, [3] gives *change propagation equations* that show how deletions and insertions are propagated up through each of the relational operators. The work of [3] was extended to include aggregate operators by Quass in [12].

The change-table technique can be thought of as introducing a new definition for New(V, t). The expression New(V, t) is defined for general view expressions as

$$\text{New}(V, t) = (V \; \textit{REFRESH} \; \square(V, t)),$$

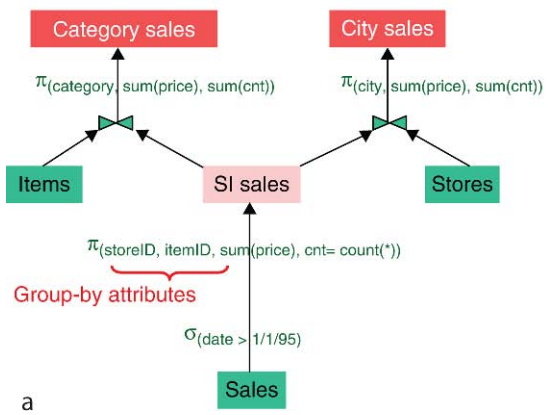
where $\square(V, t)$ is the "change-table" for the transaction t and *REFRESH* is the "refresh" operator used to apply the changes in the change table to its view. The above new definition of New(V, t) is motivated from the following observation. In the case of general view expressions involving aggregate operators, it is usually more efficient to propagate the change tables beyond an aggregate operator, instead of propagating insertions and deletions. Propagation of a change table is particularly efficient when the change table depends only on the changes to the base relation (self-maintainability [5]), while the insertions and deletions depend on the old value of the view. As shown in the example below, if the aggregate node is not materialized, the computation of insertions and deletions could be very expensive. The above new definition of New(V, t) also means that in order to obtain a complete technique it is necessary to define a general refresh operator, show how to generate a change table, and how to propagate a change table through various operators. The reader is referred to [7] for the above details.

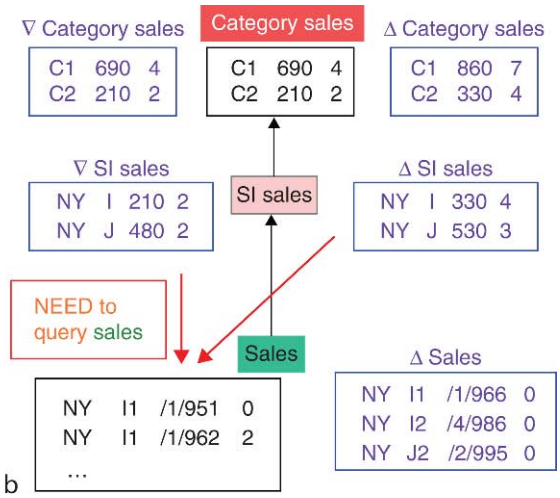
Illustrative Example. Consider the classic example of a warehouse containing information about stores, items, and day-to-day sales. The warehouse stores the information in three base relations viz. `stores`, `items`, and `sales` having the following schemas.

`stores` (storeID, city, state)
`items` (itemID, category)
`sales` (storeID, itemID, date, price)

For each store location, the relation `stores` contains the storeID, the city, and the state in which the store is located. For each item, the relation `items` contains its itemID and its category. An item can belong to multiple categories. The relation `sales` contains detailed information about sales transactions. For each item sold, the relation `sales` contains a tuple storing the storeID of the selling store, itemID of the item sold, date of sale, and the sale price.

Views. Consider the views `SISales`, `CitySales`, and `CategorySales` defined over the base relations as shown in Fig. 1a. The view `SISales` computes for each storeID and itemID the total price of items sold after 1/1/95. The view `SISales` is an intermediate view used to define the views `CitySales` and `CategorySales`. The view `CitySales` stores, for each city, the total number and dollar value of sales of all the stores in the city. The view `CategorySales` stores the total sale for each category of items. All the above described views consider only those sales that occur after 1/1/95. The views `CitySales` and `CategorySales` are stored (materialized) at the data warehouse and this is represented below by the keyword "MATERIALIZED" (The keyword "MATERIALIZED"



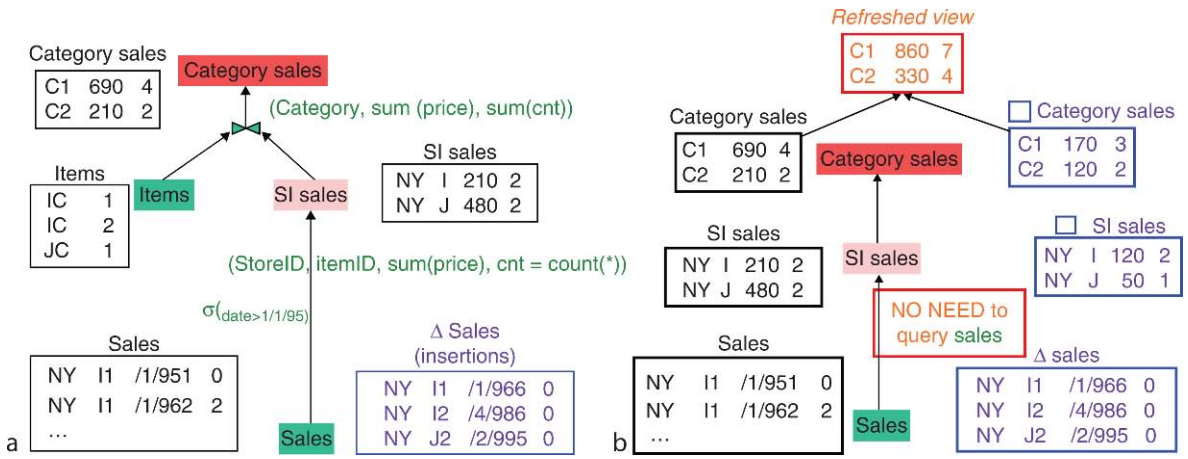


is not supported by SQL, but has been introduced in this article) in the SQL definitions of the views. It is desirable to maintain these materialized views in response to insertions to the base relation `sales` for the instance shown in Fig. 1b.

```
CREATE VIEW SISales AS
SELECT storeID, itemID, sum(price) AS
SumSISales, count(*) AS NumSISales
FROM sales
WHERE date>1/1/95
GROUP BY storeID, itemID;
CREATE MATERIALIZED VIEW CitySales AS
SELECT city, sum(SumSISales) AS SumCi-
Sales, sum(NumSISales) AS NumCiSales
FROM SISales, stores
WHERE SISales.storeID = stores.storeID
GROUP BY city;
CREATE MATERIALIZED VIEW CategorySales
AS
SELECT category, sum(SumSISales) AS
SumCaSales, sum(NumSISales) AS
NumCaSales
FROM SISales, items
WHERE SISales.itemID = items.itemID
GROUP BY category;
```

Insertion-Deletion Technique. Griffin and Libkin in [3] update view expressions by recursively computing insertions and deletions for each of the subexpressions in the view expression in response to changes at the base relations. Quass in [12] extends the techniques in [3] by including aggregate operators. Of the maintenance approaches that propagate insertions and deletions, only [12] provides techniques to maintain general view expressions involving aggregate operators. In the example, the insertions to `sales`, Δ`sales`, result in insertions (Δ`SISales`) and deletions (∇`SISales`) to the view `SISales`, which is an aggregate view over the base relation `sales`. The expressions that compute Δ`SISales` and ∇`SISales`, as derived in [12], are quite complex (see [7]). As `SISales` is not materialized, the maintenance expressions for `SISales` essentially recompute the aggregate values of the affected tuples in `SISales` from the base relation `sales`. Using the propagation equations from [12], one can propagate Δ`SISales` and ∇`SISales` upwards to obtain expressions for ∇ `CitySales`, Δ`CitySales`, ∇`CategorySales`, and Δ`CategorySales`. Figure 2a illustrates the [12] technique for updating `CategorySales` in response to insertions into `sales`. As emphasized in the figure, the computation of Δ`SISales` and ∇`SISales` require querying the base relation `sales`, because the intermediate view `SISales` is *not* materialized.

Change-Table Technique. The change-table technique proposed in [7] is as follows. Instead of computing and propagating insertions and deletions beyond an aggregate node `SISales`, a change table is computed and propagated for `SISales`. (A change table is a general form of summary-delta tables introduced in



Incremental Maintenance of Views with Aggregates. Figure 2. (a) Insertion-deletion approach ([12]), and (b) The change-table approach [8].

[9]). Propagation of change tables yields very efficient and simple maintenance expressions for general view expressions. The change table cannot be simply inserted into or deleted from the materialized view. Rather, the change table must be applied to the materialized view using a special "refresh" operator. Denote the change table of a view V by $\square V$, and a refresh operator by *REFRESH*. In practice, there are certain parameters passed with the *REFRESH* operator, but for simplicity ignore the parameters here.

For our example, start with computing the change table $\square$SISales that summarizes the net changes to SISales. For this first level of aggregates, the expression that computes $\square$SISales is similar to that derived in [9]. The change table $\square$SISales is computed from the insertions and deletions into sales by using the same generalized projection (aggregation) as that used for defining SISales. More precisely,

$$\square\text{SISales} = \pi_{storeID, itemID, SumSISales=sum(price), NumSISales=sum(\_count)}(\Pi_{storeID, itemID, price, \_count=1}(\sigma_p(\Delta\text{sales})) \uplus \Pi_{storeID, itemID, price=-price, \_count=-1}(\sigma_p(\nabla\text{sales})))$$

where p is (date > 1/1/95)

Figure 2b presents an instance of the base relation sales and the table Δsales, which is the set of insertions into sales. For the given tables, Figure 2b also shows the computed table $\square$SISales. Next propagate the change table $\square$SISales upwards to derive expressions for the change tables $\square$CitySales and $\square$CategorySales.

$$\square\text{CitySales} = \pi_{City, SumCiSales=sum(SumSISales), NumCiSales=sum(NumSISales)}(\square\text{SISales} \bowtie \text{stores})$$

$$\square\text{CategorySales} = \pi_{category, SumcaSales=(SumSISales), NumcaSales=sum(NumSISales)}(\square\text{SISales} \bowtie \text{items})$$

Figure 2b shows the change table $\square$CategorySales for the instance of the base table items in Fig. 1b. The change table $\square$CitySales can be similarly computed. The new propagated change tables are then used to refresh their respective materialized views CitySales and CategorySales using the refresh equations below. The details of the refresh equations are in [7].

CitySales = CitySales *REFRESH* $\square$CitySales, and
CitySales = CategorySales *REFRESH* $\square$CategorySales

As SISales is not materialized, it does not need to be refreshed. Also, as emphasized in Fig. 2b, it is not necessary to query the base relation sales for updating CitySales or CategorySales, which results in huge savings. The refresh operation is illustrated by showing how the CategorySales view is refreshed. Figure 2b shows the materialized table CategorySales for the given instance of base tables. For each tuple $\square v$ in $\square$CategorySales, one looks for a matching tuple in CategorySales using the join condition CategorySales.category = $\square$CategorySales.category (specified in one of the parameters of *REFRESH*). For example, the tuple <C1,170,3> in $\square$CategorySales matches with the tuple v = <C1,690,4> of CategorySales. The tuple <C1,170,3> in $\square$CategorySales means that three more sales totaling \$170 have occurred for C1 category. The total number of sales for C1 is now 7 for a total amount of \$860. To reflect the change, the tuple v is updated to <C1,860,7> by adding together the corresponding aggregated attributes (specified in another parameter of *REFRESH*).

Cost Comparison. It is shown in [7] that for typical sizes of base tables and changes to the base tables, the number of tuple accesses (reads and writes) incurred by the change-table technique is an order or magnitude less than that by the insertion-deletion propagation technique.

Cross-references

- ▶ Data Warehouse
- ▶ View Maintenance
- ▶ Views

Recommended Reading

1. Blakeley J.A. and Martin N.L. Join index, materialized view, and hybrid hash join: A performance analysis. In Proc. 6th Int. Conf. on Data Engineering, 1990, pp. 256–263.
2. Colby L., Kawaguchi A., Lieuwen D., Mumick I., and Ross K. Supporting multiple view maintenance policies. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1997, pp. 405–416.
3. Griffin T. and Libkin L. Incremental maintenance of views with duplicates. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1995, pp. 316–327.
4. Griffin T., Libkin L., and Trickey H. A correction to "incremental recomputation of active relational expressions" by Qian and

Wiederhold. Technical report, AT&T Bell Laboratories, Murray Hill, NJ, 1994.
5. Gupta A., Jagadish H., and Mumick I.S. Data integration using self-maintainable views. In Advances in Database Technology, Proc. 5th Int. Conf. on Extending Database Technology, 1996, pp. 140–144.
6. Gupta A., Jagadish H.V., and Mumick I.S. Maintenance and self-maintenance of outerjoin views. In Proc. 3rd Workshops on Next Generational Inf. Tech. and Syst., 1997.
7. Gupta H. and Mumick I.S. Incremental maintenance of aggregate and outerjoin expressions. Inform. Syst., 31(6), 2006.
8. Gupta A., Mumick I., and Subrahmanian V. Maintaining views incrementally. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1993, pp. 157–166.
9. Mumick I., Quass D., and Mumick B. Maintenance of data cubes and summary tables in a warehouse. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1997, pp. 100–111.
10. Palpanas T., Sidle R., Cochrane R., and Pirahesh H. Incremental maintenance for non-distributive aggregate functions. In Proc. 28th Int. Conf. on Very Large Data Bases, 2002, pp. 802–813.
11. Qian X. and Wiederhold G. Incremental recomputation of active relational expressions. IEEE Trans. Knowl. Data Eng., 337–341, 1991.
12. Quass D. Materialized Views in Data Warehouses. PhD thesis, Stanford University, Department of Computer Science, 1997. Chapter 4. Preliminary version appears as Maintenance Expressions for Views with Aggregation in the ACM Workshop on Materialized Views, 1996.

Incremental k-Distance Join

► Closest-Pair Query

Incremental Maintenance of Recursive Views

► Maintenance of Recursive Views

Incremental View Maintenance

► Incremental Computation of Queries

Indefinite Information

► Incomplete Information

Index Creation and File Structures

Steven M. Beitzel[1], Eric C. Jensen[2], Ophir Frieder[3]
[1]Telcordia Technologies Piscataway, NJ, USA
[2]Twitter Inc., San Fransisco, CA, USA
[3]Georgetown University, Washington, DC, USA

Synonyms

Indexing; Inverted indexes

Definition

A core element of modern information retrieval systems is the document index. The index is a set of data structures that are constructed from a source document collection with the goal of allowing an information retrieval system to provide timely, efficient response to search queries. The process of index creation typically involves reading and processing the source document collection, parsing the text in each individual document and extracting the necessary features to allow for retrieving and ranking that document in response to a user query. Additionally, indexing systems often use dimension reduction, compression, and other related techniques to drastically reduce the storage footprint of the source collection in its indexed form. Document indexes are frequently stored in a set of file structures that are conducive to rapid retrieval and ranking by an information retrieval system in response to a query.

Historical Background

As document collections have grown in size, the need to search them in an efficient and effective fashion has grown accordingly. At the time of this writing, modern, moderately-sized document collections measure in the hundreds of gigabytes, with search engines indexing tens of billions of pages [2]. Given the sheer scale of the search problem, researchers have invested a considerable amount of effort in recent years on developing efficient file structures for indexing and storing large document collections on disk. With the advent of the web (and the subsequent rapid explosion of searchable digital documents) during a time in which disk space and processing power were substantially more expensive than they are now, indexing approaches typically focused on advanced techniques for data compression to reduce the storage footprint for an index (thoroughly discussed in [3]), and on aggressive dimension reduction, such as the removal

of very frequent terms ("stopwords"), or very short documents, with the goal of reducing the total number of objects of interest tracked by the index created over the collection. As processing speeds have increased and persistent storage has gotten cheaper, indexing systems have focused on more complex algorithms intended to capture a richer set of information about the documents in the collection.

In addition to extracting key features (terms, frequencies) from the source documents and constructing data structures for storage and retrieval, indexing systems often tailor their file structures to accommodate the specific needs of the information retrieval system. For example, this may involve special extensions to allow for high-performance updates to the index, distributed indexing techniques for very large collections or high-performance indexing, or the collection of special metadata that are specific to the search tasks at hand (i.e., localized search, news search, expert search, etc). Many recent research efforts have focused on these extensions to core indexing strategies. An excellent survey can be found in [4].

Foundations

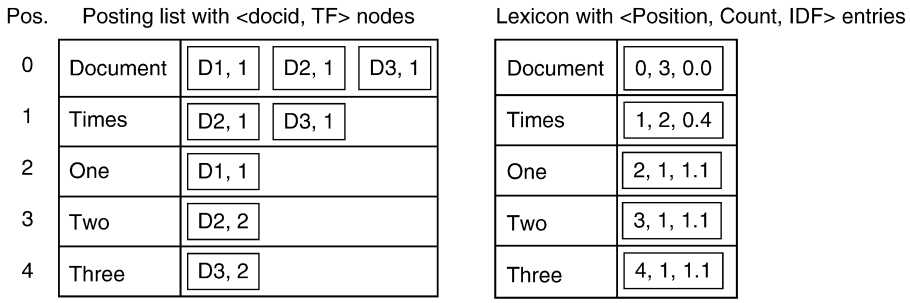
At a high level, indexing a collection of documents involves three main steps. First, the document collection must be parsed. This involves reading in each document in the collection and converting the source text of each document into a single unified form that is readable by other components in the indexing process. This step typically resolves the formatting differences between different types of source documents (i.e., HTML web pages, Microsoft Word™ documents, PDF™ files, E-Mail messages, raw text, etc). For efficiency purposes, it is often possible and desirable to perform the document parsing step in a massively parallel fashion since most documents can be parsed in a completely independent fashion.

The second major step in the indexing process involves extracting key features from the source text of each document that will be used in the retrieval process. In practice, the precise set of features is dependant on the retrieval strategy of the overall system, but distinct terms and their associated frequencies in each document are almost always included.

The final step involves sorting any intermediate representations and compiling them into a unified set of final data structures for storing on the disk. A common type of index structure that is frequently referenced in the literature is the *inverted index* [1], which represents a reasonably static document collection as a posting list, where each element in the list maps a distinct term in the collection to a set of documents containing that term. Furthermore, this list can be sorted in various ways depending on the optimization strategies imposed by the underlying search engine (sorting by descending term frequency is a common approach, but there are many others). Each node in the posting list entry (which represents a document containing the term at the head of the posting list entry) may contain extra information about that term as it pertains to the document in question. A common example of this is the term's position within the document, which is valuable for proximity search, as well as presenting results with *keyword-in-context,* meaning that result summaries will display query terms highlighted with surrounding text to give the user an idea of how their keywords are used in the result document.

In many systems, the posting list is supported by a secondary data structure known as a lexicon, which contains an entry for each distinct term in the collection along with a set of metadata that help the retrieval algorithms score target documents for relevance, including statistics like the inverse document frequency (IDF) of the term and the size of that term's posting list. Additionally, for efficiency purposes, the lexicon may include offset addresses into the inverted file, which is often stored as a random access file on the disk. Other efficiency gains may be observed if the lexicon is sufficiently small to fit in main memory, which eliminates the need for extra disk-bound I/O.

For an example of inverted index construction, consider a small collection with three documents. Document one contains the text "This is document one." Document two reads "This is document two, times two" and document three reads "This is document three, times three." The indexing system must read and parse these three documents, extract their terms, and build the posting list and lexicon. As mentioned above, a common technique used to reduce the size of the index is to remove very frequent terms that add little overall information to a document's content (typically called "stopwords"). For this example, assume that the terms "this" and "is" are considered stopwords and removed accordingly. Thus, at the end of the parsing process, five distinct terms in this example collection remain: "document", "times", "one", "two", and "three". Each distinct



Index Creation and File Structures. Figure 1. Example inverted index with Lexicon.

term is represented as an entry in the posting list with the associated lexicon, as shown in Fig. 1.

As discussed above, each element in the posting list represents a distinct term in the collection, and each term references a list of documents containing that term along with the term frequency (TF) of that term in the document. The Lexicon contains a listing of all distinct terms along with their position in the posting list, the total number of posting list nodes for the term, and the inverse document frequency (IDF) of that term in the collection, which is defined as the log of the ratio of total documents to the number of documents containing the term [1]. It is easy to see how the retrieval process could operate using this data structure, simply looking up each query term in the lexicon, using the position information there to retrieve the posting list entry for that term, and using the term frequency, IDF, and any other available statistics to rank each document for inclusion in the final results presented to the user.

Key Applications

Indexing processes and index file structures are required for virtually all modern Information Retrieval systems, given the size and complexity of modern document collections. In addition to indexing large, relatively static collections of documents, there is also research that addresses the construction and maintenance of dynamic and distributed indexes, which represent a significant area of focus as the amount of searchable data continues to grow and the need to search becomes more and more pervasive and decentralized.

Cross-references

- ▶ Lexical Analysis of Textual Data
- ▶ Stoplists
- ▶ Text Index Compression

Recommended Reading

1. Grossman D. and Frieder O. Information retrieval: algorithms and heuristics. 2nd Edn. Springer, 2004.
2. The size of the World Wide Web: www.worldwidewebsize.com – Retrieved March, 2008.
3. Witten I.H., Moffat A., and Bell T.C. Managing gigabytes: compressing and indexing documents and images. 2nd Edn. Morgan Kaufmann, 1999.
4. Zobel J. and Moffat A. Inverted files for text search engines. ACM Comput. Surv., 38:(2), 2007.

Index Join

Jingren Zhou
Microsoft Research, Redmond, WA, USA

Synonyms

Index join; Index loop join; Index nested loop join

Definition

The index join is a variant of the nested loop join in database systems using index. The join predicate can be either an equality predicate or a range predicate. The algorithm starts with reading the *outer* relation R. For each tuple $\mathcal{R} \in R$, instead of scanning the entire *inner* relation S, an index on S is used to find matching tuples and add them to the result.

Key Points

An index on S is applicable for an index join if one join attribute is the leading indexed key of the index. If the join predicate is an equality predicate, an index lookup is performed for each outer tuple. If the join predicate is a range predicate, for each outer tuple, an index seek is performed to locate the first matching tuple, followed by an index scan for the rest matching tuples. Compared

with the nested loop join, the index join saves disk I/Os for reading the entire inner relation *S* multiple times.

Cross-references

► Evaluation of Relational Operators
► Parallel Join Algorithms

Recommended Reading

1. Mishra P. and Eich M.H. Join processing in relational databases. ACM Comput. Surv., 24(1):63–113, 1992.

Index Loop Join

► Index Join

Index Nested Loop Join

► Index Join

Index Sequential Access Method (ISAM)

► Tree-based Indexing

Index Structures for Biological Sequences

Tamer Kahveci
University of Florida, Gainesville, FL, USA

Definition

Biological sequence databases are mainly composed of DNA, RNA, and protein sequences. DNA and RNA sequences are polymers of nucleotides, whereas proteins are polymers of amino acids. A database of biological sequences contains a set of biological sequences of the same type. The length of each sequence varies from less than a hundred to several hundred million bases. An index structure on a database of biological sequences helps in identifying sequences in that database that are similar to a given query sequence quickly. The definition of similarity depends on two orthogonal parameters; similarity function and the length of the similarity of interest.

The simplest similarity function is the edit distance, which measures the number of substitutions, insertions, and deletions needed to transform one sequence to the other. More complex functions involve variable gap penalties and substitution scores based on how frequent substitutions are observed in nature. The length of the similarity can be either the entire sequence (global alignment) or a subsequence of the database and the query sequence (local alignment). Depending on how the similarity is defined using these parameters, the similarity may or may not be metric. The index structure used for the biological sequence database needs to be suitable to accommodate the underlying similarity measure.

Historical Background

Levene identified the nucleotide bases in 1919. He found that the nucleotides can form a chain as the phosphate in a nucleotide can create bond with other nucleotide. Watson and Crick discovered the structure of the DNA in 1953 using X-ray diffraction. The efforts to create a database of DNA sequences, named GenBank, has started in 1979. By 1983, there were around 2,000 sequences in GenBank. Since then, the size of genome databases is increasing exponentially [2]. Statistics show that the size of GenBank has doubled every 15 months. In August 2005, GenBank contained over 100 billion bases. This rapid growth in the size of the biological sequence databases coupled with the costly distance measures made the use of index structures essential for this data type.

One of the earliest index structures used for biological sequences is the hash table [10]. Hash tables are often used to index *k*-grams (subsequences of length *k*) of the sequences in the database. This index structure is used to find exact matches between the *k*-grams of a given query sequence and that of the database sequences. Popular biological database search tools such as FASTA [10] and BLAST [1] employed hash tables to index *k*-grams. These tools created one hash table entry for each unique *k*-gram. Thus, the number of hash table entries is exponential in *k*. This limits the value of *k* to a small number. Another drawback of the hash table is that the value of *k* has to be predetermined at the time of index construction.

Therefore, the same sized k-grams need to be used for all the queries. Pol and Kahveci incorporated randomization to enable variable k value for hash table construction [11].

Suffix trees and suffix arrays constitute another set of commonly used index structures for biological sequences. Suffix trees were first proposed by Weiner [14] under the name *position tree*. Unlike the hash table, this structure index all the suffixes of all the sequences in the database. McCreight proposed a space efficient technique for the construction of the suffix trees [9]. Later, Ukkonen developed an on-line construction method [12]. A variety of other suffix tree implementations have also been proposed such as implicit suffix tree [12], string B-tree [4] and suffix array [8]. AVID [3] use the suffix tree structure to query biological sequences.

Hash tables and suffix trees can identify exact matches. The tools that use them perform additional processing to identify approximate matches. Reference-based index structures enable approximate searches. The VP-tree (Vantage Point tree) [15] indexes the data in general metric space with the help of references. Omni method [5] proposed to select the references near the convex hull of the database, far away from each other. Venkateswaran et al, later showed that multiple references should be used to index a database sequence, and the references should be a combination of similar and distant sequences [13].

The Sequence Search Tree (SST) [6] maps each sequence to a set of vectors in a high dimensional integer space. It then builds an index on these vectors. The Multi Resolution String (MRS) index structure [7] also maps the subsequences into a high dimensional integer space. Unlike SST, the mapping of the MRS index structure allows computation of a lower bound to the distance. This way MRS avoids false dismissals.

Foundations

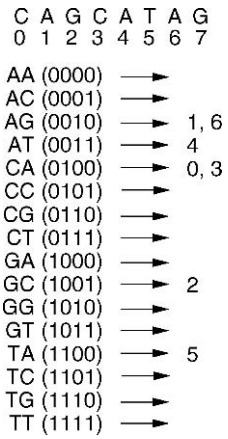
Hash Tables

One of the most widely used index for sequence search is the *hash table* (also called the *lookup table*). Hash table enables quick lookup for a set of pre-specified sequences. Usually, in the bioinformatics literature, the hash table is built on the k-grams, where k is predefined.

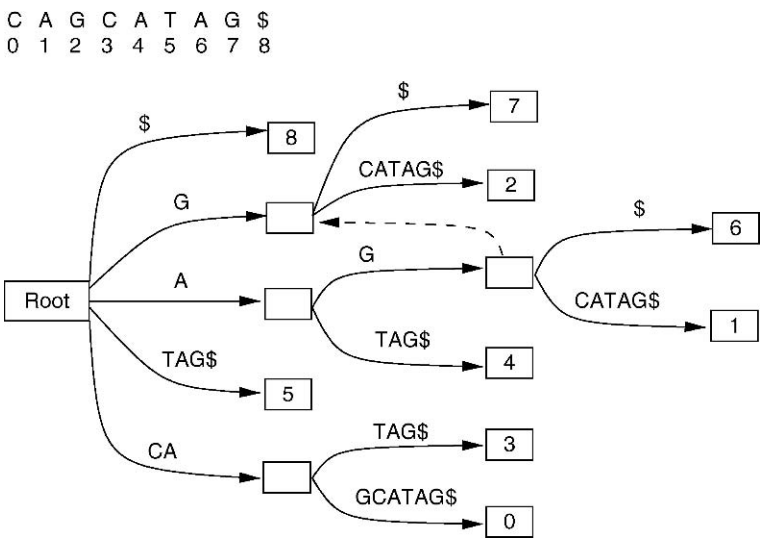
The hash table of a sequence is built using two parameters: alphabet and k. Assume that the alphabet contains σ letters. The hash table encodes the ith letter in the alphabet as the binary representation of $i - 1$ using $\lceil log_2\sigma \rceil$ bits. For example, the letters in the DNA alphabet {A, C, G, T} are encoded as A = 00, C = 01, G = 10, and T = 11. The hash key of a k-gram is the concatenation of the binary representations of the letters that constitute that sequence. For example, the DNA sequence GGCA is represented as 10100100. Figure 1 shows the hash table built on a DNA sequence for $k = 2$. In this example, the hash table contains 4^2 entries since $\sigma = 4$. The total number of pointers to database is the $N - k + 1$, where N is the number of letters of the sequence indexed.

Suffix Trees

Suffix tree is a tree structure where each path from the root to a leaf node denotes a suffix of the sequence that it indexes. Thus, there is a bijection between the leaf nodes and the suffixes. In order to be able to create this bijection unambiguously, suffix tree increases the length of each of the database sequences by one letter by appending a unique "stop" character at the end of the sequence. This character is not contained in the alphabet that constitutes the database sequences. Figure 2 shows the suffix tree built on sequence CAGCATAG. The letter, \$, marks the end of the sequence. Every path from the root node to a leaf node through the solid arrows defines a suffix of this sequence. The labels on the edges of each such path show the



Index Structures for Biological Sequences. Figure 1. A DNA sequence and the hash table constructed on it when $k = 2$. The numbers in parenthesis show the binary code of the corresponding sequence.



Index Structures for Biological Sequences. Figure 2. Suffix tree built on the sequence CAGCATAG. The dashed arrow is a suffix link. The letter, $, marks the end of the sequence.

contents on that suffix. The numbers at the leaf nodes show the starting position of the suffix denoted by that leaf. The dashed arrows in the figure are the suffix links. There is a suffix link from an internal node u to another internal node v if u and v are labeled with suffixes $c\alpha$ and α respectively for a given letter c. Suffix links enable faster construction of suffix trees.

Suffix trees are notorious for their excessive memory usage. Although the space complexity is $O(n)$, the constant in the big-Oh may be large. The size of the suffix tree depends on the alphabet and the distribution of the letters in the database sequence. The literature reports the size of the suffix tree as 10–70 bytes per letter in the database.

C A G C A T A G
0 1 2 3 4 5 6 7

6	AG
1	AGCATAG
4	ATAG
0	CAGCATAG
3	CATAG
7	G
2	GCATAG
5	TAG

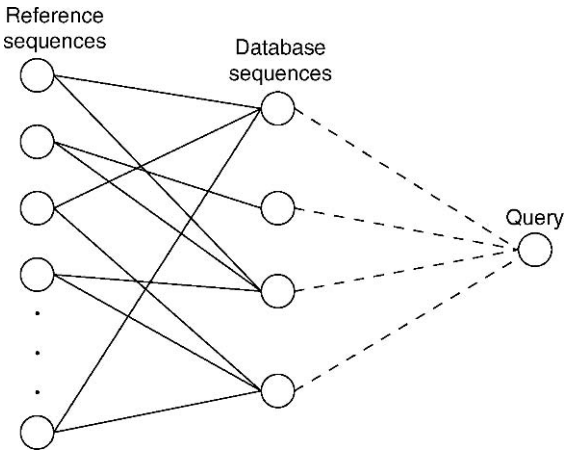
Index Structures for Biological Sequences. Figure 3. Suffix array built on the sequence CAGCATAG and the corresponding suffixes indexed by the entries of this index.

Suffix Arrays

Suffix arrays reduces the space consumption of suffix trees to five bytes per letter at the expense of increased search time complexity. The suffix array of a sequence is an array of integers that shows the alphabetical order of all suffixes of that sequence. Figure 3 shows the suffixes of CAGCATAG and the suffix array constructed on it.

Reference-Based Indexing

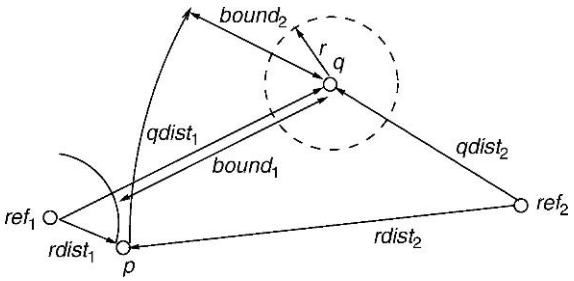
Reference-based index structures work when the distance measure is metric. The distance measure is metric when the distance function is metric and the entire sequences are aligned. An example to metric distance functions is the edit distance. Reference-based index structures select a small number of sequences, referred to as the set of *reference sequences*. Often, these references are selected from the database sequences for simplicity. However, it is possible select sequences that are not in the database as references. The structure pre-computes the distances between the references and the database sequences. For a given a query sequence, the query algorithm first computes the distance from each of the references to the query sequence. It then computes upper and lower bounds to the distance between the query sequence and the database sequences with the help of the pre-computed distances



Index Structures for Biological Sequences. Figure 4. Reference based indexing. The *small circles* represent database, reference and query sequences. Each database sequence is indexed by two references. The *solid lines* denote which references are used for each database sequence. The distances between the sequences connected by *solid line* are precomputed. The *dashed lines* denote the distance between the query and the references. These distances are computed on the fly as the query arrives.

in the index quickly. Thus, without any further comparisons, sequences that are too close to or too far away from a reference may be removed from the candidate set with the help of the triangle inequality. Figure 4 shows a reference based index where each database sequence is indexed using two references.

Figure 5 illustrates reference-based indexing in a hypothetical two-dimensional, space. Here, the database sequences are represented by points. The distance between two points in this space corresponds to the underlying distance between the two sequences (e.g., if the points denote sequences, $rdist_1$ between the points ref_1 and p corresponds to the edit distance between the sequences represented by them). In reference-based indexing, the distances between the sequence p and references ref_1 and ref_2 are pre-computed. Let $rdist_1$ and $rdist_2$ be the two pre-computed distances, respectively. Given a query q with range r, the first step is to compute reference-to-query distances $qdist_1$ and $qdist_2$. A lower bound for the distance between sequences q and p with reference ref_1 is computed as $bound_1 = |qdist_1 - rdist_1|$ using the triangle inequality. Similarly, $bound_2$ gives a lower bound for the distance between



Index Structures for Biological Sequences. Figure 5. Two dimensional illustration of reference-based indexing. Here ref_1 and ref_2 are references. The query region is given by its center q and radius r. $qdist_1$ and $qdist_2$ are query-to-reference distances. $rdist_1$ and $rdist_2$ are distances from the reference to the data p. $bound_1$ and $bound_2$ are the bounds obtained using references.

q and p with reference ref_2. Since $bound_1 > r$ with ref_1 as the reference, sequence p can be pruned from the candidate set of q.

Vector Space Indexing

This set of index structures first map the sequences to a vector space (typically in a multi-dimensional integer space) and build index structure on this space. An example to this is the MRS (Multiple Resolution String) index.

Let s be a sequence from the alphabet $\Sigma = \{\alpha_1, \alpha_2,...,\alpha_\sigma\}$. Let n_i be the number of occurrences of the character α_i in s for $1 \leq i \leq \sigma$. The vector space mapping of s is computed as $f(s) = [n_1, n_2,...,n_\sigma]$. The vector, $f(s)$ is called the *frequency vector*, of s. For example, let s = AGCTTTTCATTCTGAC be a DNA sequence. The frequency vector of s is $f(s) = [3,4,2,7]$ ([#As,#Cs,#Gs,#Ts]), since the DNA alphabet contains the letters A, C, G, and T.

The MRS index structure stores a sequence of Minimum Bounding Rectangles (MBRs) at different resolution levels in the index structure. Resolutions are represented using window size, which are powers of two.

In order to obtain the MBRs of a sequence at resolution 2^i, a window of length $w = 2^i$ is placed at the leftmost point of that sequence. Later, this window is slid by one letter until it reaches to the end of that sequence. Each placement of this window produces a subsequence. The frequency vectors of all those windows are computed. First, the minimum box, called

Minimum Bounding Rectangle (MBR), that covers the frequency vector of the first subsequence is computed. This box is later extended to cover more frequency vectors until the box capacity is reached. Box capacity is an integer that denotes the maximum number of frequency vectors that an MBR can contain. Typically, this number is set to 1,000 to achieve good performance result. Once the box capacity is reached, a new MBR is created to cover the next subsequences. This process continues until all subsequences are transformed. Note that only the lower and higher end points of the MBRs along with the starting locations of the first subsequence contained in that MBR are stored for each MBR.

Key Applications

The ability to query large biological sequence databases is needed in nearly all areas of molecular biology, pharmacology, plant sciences and horticulture. Two example applications are as follows.

Repeat identification. DNA sequences contain large amounts of repeating patterns. Identifying these patterns is essential for many purposes. For example, repeat copy numbers can help in determining ancestral relationship, which is often needed to identify victims, criminals, parenthood, whether a race horse is pure breed, etc. Furthermore, identifying these repeats is needed for more accurate sequence assembly and for high quality primer production. Identifying repeats require comparing the sequences in a repeat library and the target sequence (library-based repeat identification) as well as a self comparison of the target sequence (de-novo repeat identification). The size of the target sequence and the repeat library often necessitates the use of an index structure.

Shotgun sequencing. One of the commonly used technologies to identify the letters in large chromosomes is called *shotgun sequencing*. This technology first produces multiple copies of a given long DNA sequence. It then chops these sequences into short fragments from (almost) random locations using restriction enzymes. It then identifies the sequences that have less than 1,000 letters using high throughput sequencing machines. This process produces a bag of short subsequences of the target DNA sequences. The challenge is then to reassemble the original long DNA sequence from these short fragments. This problem requires an all-to-all sequence comparison for repeat and overlap detection. The massive size of the database makes it essential to use an index structure.

Data Sets

GenBank http://www.ncbi.nlm.nih.gov/Genbank/
PDB http://www.rcsb.org/pdb/
SwissProt http://ca.expasy.org/sprot/

Cross-references

► Biological Sequence
► Query Languages and Evaluation Techniques for Biological Sequence Data

Recommended Reading

1. Altschul S., Gish W., Miller W., Meyers E.W., and Lipman D.J., Basic Local Alignment Search Tool. J. Mole. Biol., 215 (3):403–410, 1990.
2. Benson D., Karsch-Mizrachi I., Lipman D., Ostell J., Rapp B., and Wheeler D. GenBank. Nucleic Acids Res., 28(1):15–18, 2000.
3. Bray N., Dubchak I., and Pachter L. AVID: a global alignment program. Genome Res., 13(1):97–102, 2003.
4. Ferragina P. and Grossi R. The string B-tree: a new data structure for string search in external memory and its applications. J. ACM, 46(2):236–280, 1999.
5. Filho R.F.S., Traina A.J.M., Caetano Traina J., and Faloutsos C. Similarity search without tears: The OMNI family of all-purpose access methods. In Proc. 17th Int. Conf. on Data Engineering, 2001, pp. 623–630.
6. Giladi E., Walker M., Wang J., and Volkmuth W. SST: an algorithm for finding near-exact sequence matches in time proportional to the logarithm of the database size. Bioinformatics, 18 (6):873–877, 2002.
7. Kahveci T. and Singh A. An efficient index structure for string databases. In Proc. 27th Int. Conf. on Very Large Data Bases, 2001, pp. 351–360.
8. Manber U. and Myers E. Suffix arrays: a new method for on-line string searches. SIAM J. Comput., 22(5):935–948, 1993.
9. McCreight E. A space-economical suffix tree construction algorithm. J. ACM, 23(2):262–272, 1976.
10. Pearson W. and Lipman D. Improved tools for biological sequence comparison. In Proc. Natl. Acad. Sci., 85:2444–2448, 1988.
11. Pol A. and Kahveci T. Highly scalable and accurate seeds for subsequence alignment. In Proc. IEEE Int. Conf. on Bioinformatics and Bioengineering, 2005.
12. Ukkonen E. On-line Construction of Suffix-trees. Algorithmica, 14:249–260, 1995.
13. Venkateswaran J., Lachwani D., Kahveci T., and Jermaine C. Reference-based indexing for metric spaces with costly distance measures. VLDB J. 17(5):1231–1251, 2008.
14. Weiner P. Linear pattern matching algorithms. In Proc. IEEE Symposium on Switching and Automata Theory, 1973, pp. 1–11.
15. Yianilos P. Data structures and algorithms for nearest neighbor search in general metric spaces. In Proc. 4th Annual ACM - SIAM Symp. on Discrete Algorithms, 1993, pp. 311–321.

Index Tuning

Philippe Bonnet[1], Dennis Shasha[2]
[1]University of Copenhagen, Copenhagen, Denmark
[2]New York University, New York, NY, USA

Definition

Index Tuning is concerned with when and how to construct an index.

Historical Background

When relational models were first introduced, customers complained that vendors had introduced them to slowed down customer applications in order to make the customers buy more hardware. Vendors and researchers responded to the performance challenge by incorporating existing data structures (notably B-trees and hash structures) and improving many new ones (notably bit vectors and multi-dimensional indexes). The decision of which data structures to create was left to the user. This entry concerns that decision.

Foundations

An *index* is a data structure plus a method of arranging the data tuples in the table (or other kind of collection object) being indexed. The syntax for constructing indexes is discussed elsewhere. This entry presents the tuning considerations.

Two data structures are most often used in practice for indexes: B+-trees and Hash structures. Of these, B+-trees are used the most often. The folk wisdom holds that if one doesn't know which kind of indexes to put on a column or set of columns and scanning is too slow, then one should use a B+-tree. When performing accesses based on equality, however, hash structures perform better.

The main tuning consideration for *B+-trees* is to avoid having too many levels. Because an access to disk secondary memory costs a few milliseconds if it requires a seek (as index accesses will) and even an access to flash memory may require switching blocks (a relatively expensive operation), the performance of a B+-tree depends critically on the number of nodes in the average path from root to leaf – the number of levels (the root will tend to be in RAM, but the other levels may or not be, and the farther down the tree the search goes, the less likely it is for the nodes to be in RAM). One technique that database management systems use to minimize the number of levels is to make each interior node have as many children as possible (1,000 or more for many B+-tree implementations). The maximum number of children a node can have is called its *fan-out*. Because a B+-tree node consists of key-pointer pairs, the larger the key is, the lower the fan-out.

For example, a B+-tree with a million records and a fan-out of 1,000 requires three levels (including the level where the records are kept). A B+-tree with a million records and a fan-out of 10 requires seven levels. If one increases the number of records to a billion, the numbers of levels increase to four and ten respectively. This is why accessing data through indexes on large keys is slower than accessing data through small keys on most systems (the exceptions are those few systems that offer good compression).

Hash structures, by contrast, store key–value pairs based on a pseudorandomizing function called a *hash function*. The hash function can be thought of as the root of the structure. Given a key of any size, the hash function returns a location that contains either a page address (usually on disk) or a directory location that holds a set of page addresses. That page either contains the key and associated record or is the first page of a linked list of pages, known as an *overflow chain* leading to the record(s) containing the key. (One can keep overflow chaining to a minimum by allocating enough hash buckets such that the space in the hash buckets is at least twice the space required to hold the data.) In the absence of overflow chains, hash structures can answer equality queries (e.g., find the employee with a certain Social Security number) in one disk access, making them the best data structures for that purpose.

The data structure portion of an index has pointers at its leaves to either data pages or data records. If there is at most one pointer from the data structure to each data page, then the index is said to be *sparse*. If there is one pointer to each record in the table, then the index is said to be *dense*.

If records are small compared to pages, then there will be many records per data page and the data structure supporting a sparse index will usually have one fewer level than the data structure supporting a dense index. This means one less disk (or flash block) access if the table is large. By contrast, if records are almost as large as pages, then a sparse index will rarely have better disk access properties than a dense index.

The main virtue of dense indexes is that they can support certain read queries within the data structure itself in which case they are said to *cover* the query. For example, if there is a dense index on the keywords of a document retrieval system, a query can determine the number of records containing some term, e.g., "derivatives scandals," without accessing the records themselves (Count information is useful for that application, because queriers frequently reformulate a query when they discover that it would retrieve too many documents). A secondary virtue is that a query that makes use of several dense indexes can identify all relevant tuples before accessing the data records. Instead, one can form intersections and unions of pointers to data records or of record identifiers.

A *clustering index* on an attribute (or set of attributes) *X* is an index that puts records close to one another if their *X*-values are *near* one another. What "near" means depends on the data structure. On B-trees, two *X*-values are near if they are close in their sort order. For example, they are close numerically or alphabetically. In hash structures, two *X*-values are near only if they are identical. Index-organized tables are clustering indexes where *X* is the primary key.

Sparse indexes must be clustering, but clustering indexes need not be sparse. In fact, clustering indexes are sparse in some systems (e.g., SQL Server, ORACLE hash structures) and dense in others (e.g., ORACLE B-trees, DB2). Because a clustering index implies a certain table organization and the table can be organized in only one way at a time, if there is a clustering index on the sequence of attributes *Y*, then any other clustering index on attributes *X* must have the property that *X* is a prefix of *Y* or *Y* is a prefix of *X*.

A *nonclustering index* (sometimes called a *secondary* index) is an index on an attribute (or sequence of attributes) *Y* that puts no constraint on the table organization. The table can be clustered according to some other attribute *X* or can be organized as a heap, as discussed below. A nonclustering index must be dense, so there is one leaf pointer per record. There can be many nonclustering indexes per table.

A *heap* is the simplest table organization of all. In the basic implementations, records are ordered according to their time of entry. That is, new insertions are added to the last page of the data structure. In this case, inserting a record requires a single page access.

Nonclustering indexes are very useful if they cover a query but can also be useful if the query retrieves significantly fewer records than there are pages in the file. The word "significant" needs explanation: a table scan can often save time by reading many pages at a time, provided the table is stored on contiguous tracks. Therefore, if a query requires most records in a table, a scan may be 2–10 times faster than an index read.

Decision support applications often entail querying on several, perhaps individually unselective, attributes. For example, "Find people in a certain income range who are male, live in California, buy boating equipment, fish, drive a sports car, and work in the computer industry." Each of these constraints is unselective in itself, but together form a relatively small result. The best all-around data structure for such a situation is the *bitmap*. A bitmap is a collection of vectors of bits. The length of each vector equals the length of the table being indexed and has a 1 in position i if the ith record of the table has some property. For example a bitmap on state in the United States would consist of 50 vectors, one for each state. The vector for California would have a 1 in its ith position if record i pertains to a person from California.

Key Applications

Indexes are necessary in any application that handles large amounts of data. Which indexes to choose can have an enormous impact on performance. An index may reduce the time to execute a query from hours to a few seconds in one application, yet increase batch load time by a factor of 80 in another application. Add them with care.

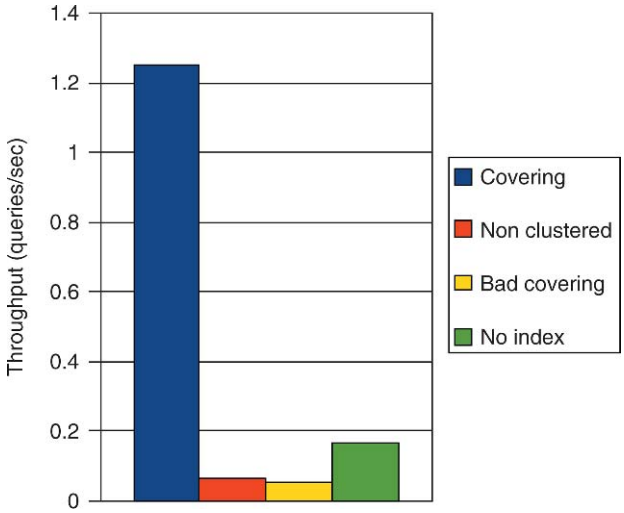
Experimental Results

Covering Experiment

This experiment illustrates two of the issues discussed above: (i) the potential benefit of covering index, and (ii) the trade-off between scans and non-clustered indexes. This experiment considers the table ACCOUNT with 25 attributes, and the following query:

select home_street, last_name from ACCOUNT where first_name='first_name12';

The ACCOUNT table has three different indexes: (i) a covering index on first_name, home_street and last_name, (ii) another covering index on home_street, first_name and last_name, and (iii) a non-clustered index on first_name. The experiment runs the same



Index Tuning. Figure 1. Covering experiment.

query using these three different indexes as well as with no index. The experiment runs those queries on MySQL 6.0 with a cold buffer (i.e., the database cache is empty and IO are required to complete the query). Figure 1 traces the results.

First, the covering index provides the best performance. The reason is that the query can be answered using the index. There is no need to dereference the index leaf pointers to access the underlying table. Second, the order of attributes in the covering index is crucial. In this query, the attribute first_name is used in the WHERE clause. A covering index has the most beneficial effect when the prefix attributes in the index key are those attributes that appear in the where clause. Third, scan sometimes wins over a non clustered index. The reason is that the few random IOs that are necessary to complete the query (the data is generated so that first_name has a 1% selectivity) take more time than the sequential IOs required to scan the whole table.

URL to Code and Data Sets

Index experiments: http://www.databasetuning.org/?sec=index

Cross-references

- ▶ B+-Tree
- ▶ Bitmap Index
- ▶ Hash-based Indexing

Recommended Reading

1. Celko J. Joe Celko's SQL for Smarties: Advanced SQL Programming (3rd edn.). Morgan Kaufmann, San Fransisco, CA, 2005.
2. Kimball R. and Ross M. The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling (2nd edn.). Wiley, New York, NY, 2002.
3. Shasha D. and Bonnet P. Database Tuning: Principles, Experiments and Troubleshooting Techniques. Morgan Kaufmann, San Fransisco, CA, 2002.
4. Tow D. SQL Tuning. OReilly, Sebastopol, CA, 2003.

Indexed Sequential Access Method

Alex Delis[1], Vassilis J. Tsotras[2]
[1]University of Athens, Athens, Greece
[2]University of California-Riverside, Riverside, CA, USA

Synonyms

Indexed sequential file; ISAM file; ISAM

Definition

An indexed sequential access method is a *static, hierarchical*, disk-based index structure that enables both (single-dimensional) range and membership queries on an ordered data file. The records of the data file are stored in *sequential* order according to some data attribute(s). Since ISAM is static, it does not change its structure if records are added or deleted from the data

file. Should new records be inserted into the data file, they are stored in an *overflow* area. Deleted records are removed from the file (leaving empty space).

Historical Background

Although transparent for the user of a DBMS, access methods play a key role in database performance. A major performance goal of a DBMS is to minimize the number of I/Os (i.e., blocks or pages transferred) between the disk and main memory. One way to achieve this goal is to minimize the number of I/Os when answering queries. Note that many queries reference only a small portion of the records in a database table. For example the query: "*find the employees who reside in Santa Monica, CA*" references only a fraction of the records in the *Employee* relation. It would be rather inefficient to have the database system sequentially read all the pages of the *Employee* file and check the residence field of each employee record for the name '*Santa Monica*'. Instead the system should be able to locate the pages with '*Santa Monica*' employee records directly. To allow such fast access, additional disk-resident structures called *indices* (or *access methods*) are designed per database relation. One of the first such methods developed was the *index sequential access method* (ISAM). ISAM was developed at IBM in late 1960s [3] and it is essentially the predecessor to the widely used B+-tree index. A major difference between ISAM and the B+-tree [1] is that instead of overflowing pages, the B+-tree introduces page splitting. The ISAM was later replaced by IBMs *virtual storage access method* (VSAM) [4] which introduced the notion of splitting (data or index pages) when there is not enough space for inserting a new record.

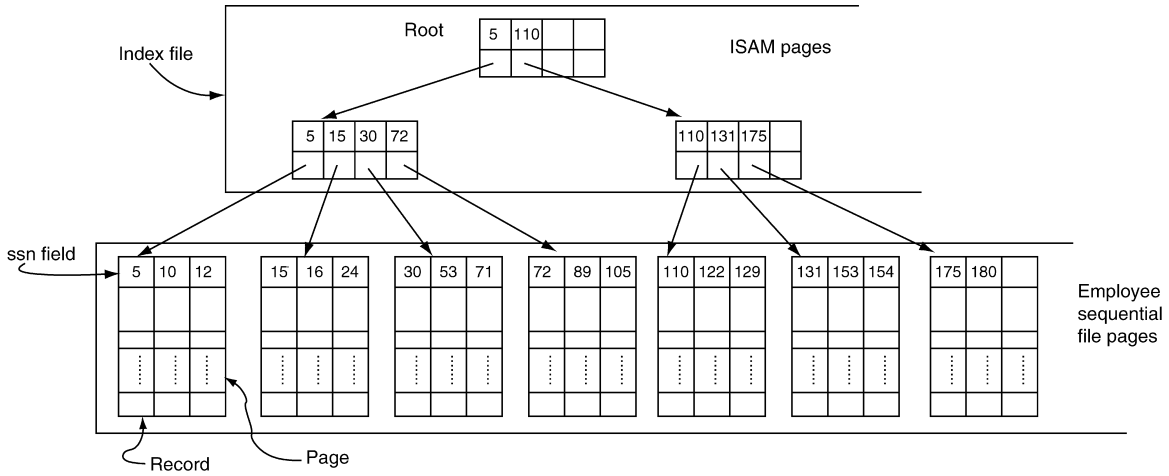
Foundations

The ISAM structure contains three separate storage areas: the data file, the index file and the overflow area. For simplicity, assume that the data file is an *Employee* relation, ordered according to the social security number or *ssn* attribute. Moreover, assume that this relation is stored sequentially on the disk, following the logical order of the *ssn* attribute. If the *Employee* file has n records and one page can hold B *Employee* records, the total number of pages in this file is $O(n/B)$. Note that each file page is full of *Employee* records except possibly the last page. Moreover, given the sequential storage of the file, each page can easily access the next page of the file in *ssn* order (it is simply the next physical page on the disk).

A straightforward way to build an index on the *Employee* file is to create a new (much smaller) file that contains one representative record from each *Employee* file page. The records in this new file are of the form: $<search\_value, ptr>$ where *ptr* is a pointer to an *Employee* file page (a *page-id* number uniquely identifying the page on the disk) and *search_value* is the smaller *ssn* recorded in that page. Since these records are smaller in size than the *Employee* file records, each page of the new file will contain many of them. If the *Employee* file is large, the new file will spread over a number of pages (this number is clearly bounded by $O(n/B^2)$). However, since the new file is also an ordered file (it has two attributes and is ordered according to the *search_value* attribute) it can be indexed by another (even smaller) level of index pages, and so on. This process continues until the creation of an index layer that consists of a single page. As a result a multi-way, tree-structured index is created whose nodes correspond to pages (see Fig. 1). It is worth pointing out that all index pages from possibly multiple index levels are resident in the index file area of the ISAM.

The ISAM organization is a single-dimensional (as opposed to multi-dimensional) index. It supports searches on the attribute (or collection of attributes) on which the data file is ordered. For example, searching the *indexed sequential access method* for a given *ssn* K (i.e., a membership query) is simple. The search starts from the root page where the record with the largest *search_value* that is less or equal to K is located. The search then continues to the page in the next index level, pointed by this record, until a page of the *Employee* file is reached. If K is found among the *ssn* values of that *Employee* page, the appropriate record is returned as answer to the query. If K is not found the answer is empty. It is easy to see that this search takes $O(logB(n/B))$ page accesses (I/Os) as this is the height (in pages) of the tree. The reader should note that the logarithm is base B, the size of the page, since this is a multi-way tree where each node has $O(B)$ fan-out.

Range queries (as in: *find the Employee records with ssn in the range* [25, 100]) are addressed similarly. A search is first performed for the *ssn* defining the lower part of the range (in the above query example this would be $ssn = 25$). This look-up will lead to an appropriate *Employee* record located in some file page. Records with higher *ssn* values within this page



Indexed Sequential Access Method. Figure 1. An indexed sequential access method.

I

are accessed until a record with *ssn* larger than the upper limit of the query range is found. If the upper limit of the query range is higher than the highest *ssn* in this page, the next page of the file is accessed and so on (recall that the file is stored sequentially). The search stops when an *Employee* page is found that contains a record with *ssn* larger than the query range.

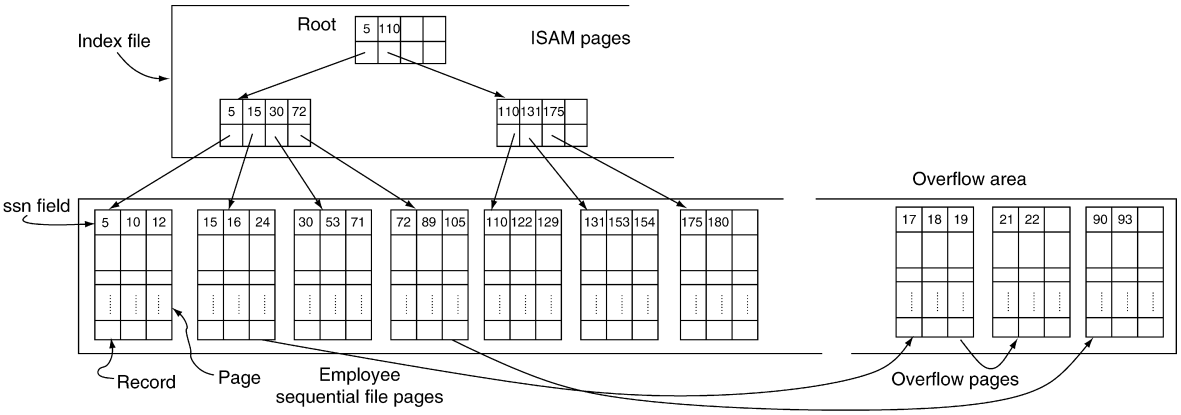
If a denotes the answer size to a range query (number of *Employee* records satisfying the query range predicate), ISAM answers a range query in $O(logB(n/B)+a/B)$ I/Os. Note that the logarithmic part is spent to find the *Employee* page with the first record that satisfies the query predicate (if any) and the $O(a/B)$ part corresponds to accessing the rest of the *Employee* pages that contain answer.

While the use of the index greatly facilitates query time, there is of course a space overhead, since the access method itself uses pages to store its records. However, this overhead is minimal. The number of pages used by the index structure is still bounded by $O(n/B)$. This is because the first level uses at most $O(n/B^2)$ pages, the second at most $O(n/B^3)$ and so on.

An interesting observation is that an indexed sequential access method imitates binary search on a disk-based environment. However, given that at each node of the index a whole page is accessed, there are $O(B)$ choices (instead of just 2 in the binary search) at each node.

The main advantages of the ISAM organization are its simplicity, small space overhead and fast query time. The structure however is *static*. If new records are added in the Employee file they are handled in an *overflow* file. Since there is no empty space in the data file, overflow pages are created to store the new records. Such pages are typically chained to the page where a record should have been stored (see Fig. 2). Various proposals exist on how to handle the overflow file [2,4,5,6]. Nevertheless, the structure of the index does not change as the size of the data file changes. This eventually affects query time. The overflow file can be merged periodically with the main *Employee* file, at which time the index needs to be recreated. Similarly, if records are deleted in the original *Employee* file, pages may be left containing very few records which affects both storage and query time. These problems are solved by the B+-tree, which is a *dynamic* indexing scheme [1].

There are two main differences between ISAM and B+-tree: firstly, when a new page is created in the B+-tree, space is left to accommodate future insertions. In practice, a newly created page starts half empty so that it can store many new records before a structural re-organization is needed. If the page becomes full of records and a new record is directed to it, the page is split into two pages (that are half full). Secondly, a page in the B+-tree is not allowed to become scarce of records (unless it is the tree's root page). As a result, when a page is accessed, it is guaranteed to contain a minimum number of records. If due to deletions a pages record occupancy falls below the threshold (half the page size) the page is merged with another page so that the combination has enough records. Note that leaving pages half empty imposes additional space overhead for the B+-tree than the ISAM; however it



Indexed Sequential Access Method. Figure 2. The indexed sequential access method with overflows.

results into a very effective dynamic height-balanced indexing scheme.

Finally, ISAM can be considered as the tree-based index alternative to the static external hashing. Both schemes are static and overflow areas are used for additional records. Their major difference is that ISAM can perform both range and membership queries, while static external hashing is designed only for membership queries.

Key Applications

ISAM has been used in early database management systems as an index method to provide fast access to range and membership queries. It was later replaced by the VSAM structure [4] which introduced the notion of page splitting. Finally, the B+-tree was proposed as a dynamic indexing structure [1] and is now the standard access method in most relational database systems.

Cross-references

- ▶ B+-Tree
- ▶ Indexing
- ▶ Membership Query
- ▶ Range Query

Recommended Reading

1. Bayer R. and McCreight E. Organization and maintenance of large ordered indexes. Acta Inf., 1(3):173–183, 1972.
2. Behymer J.A., Ogilive R.A., and Merten A.G. Analysis of indexed sequential and direct access file organizations. In Proc. 1974 ACM SIGFIDET (SIGMOD) Workshop on Data Description, Access and Control, 1974, pp. 186–212.
3. IBM Corporation. IBM System/360 Operating System Data Management Services, February 1972. Second Edition, C26-3746-1.
4. Keehn D.G. and Lacy S.O. VSAM Data set design parameters. IBM Syst. J. 13(3):186–212, 1974.
5. Larson P. Analysis of index-sequential files with overflow chaining. ACM Trans. Database Syst., 6(4):671–680, 1981.
6. Mullin J.K. An improved index sequential access method using hashed overflow. Commun. ACM, 15(5):301–307, May 1972.
7. Wong K.F. and Strauss J.G. An analysis of ISAM performance improvement options. Manag. Datamat. J., 4(3):95–107, 1975.

Indexed Sequential File

▶ Index Sequential Access Method

Indexing

▶ Biomedical Scientific Textual Data Types and Processing
▶ Index Creation and File Structures
▶ Physical Database Design for Relational Databases

Indexing and Similarity Search

Michail Vlachos
IBM T.J. Watson Research Center, Hawthorne, New York, USA

Synonyms

Data organization; Hierarchical data organization; Space segmentation; Space partitioning

Definition

Indexing refers to the process of efficient data organization. It is closely related to similarity search because it allows such costly operations over a large dataset of objects to be efficiently sped up. Indices (or indexes) are hierarchical structures that direct the search to the most promising part of the database, hence eliminating from examination a large portion of objects. One can make the analogy with phone books, where all entries are recorded in sorted alphabetical order; therefore search involves only the lookup at the relevant portion of the book.

Historical Background

Traditional indexing structures include the B-trees. However, B-trees organize the data based on a single attribute/feature. Many of todays multimedia data contain hundreds or thousands of features. As an example, a small B&W image of 50 × 50 pixels contains 2,500 points/features. In order to accommodate objects that contain more attributes, extensions to the indexing schemes have been presented. Such indexes include kd-trees [3], grid files [9], as well as R-trees [7] and its variants [2], and the various incarnations of metric trees (VP-trees [12], M-trees [6]). The ultimate goal of a successful index is to divide object space into areas with approximately equal density using a set of heuristics.

Indexes work on simplified representations of the original data, since they do not perform well for raw data with high dimensionalities. This phenonenon is known as the "curse of dimensionality", which means that for higher dimensionalities, the pruning power of the indices diminishes exponentially, and all sequences are eventually retrieved from the disk. Therefore, when an index is utilized, it is necessary to perform some data compression, also called dimensionality reduction, because this data compaction will boost the index performance.

After the raw data are compacted from the original dimensionality n to some lower dimensionality d, the compressed d-dimensional objects/points are stored in the index structure in order to expedite the search process.

Foundations

An index can facilitate the fast similarity search over the database of objects. Therefore, a user is posing a query object q against a database and is seeking the most similar (or k most similar) objects to the query q, for a given similarity measure that assesses the affinity between a pair of objects.

Index structures attribute their search efficiency to two factors:

1. *Effective space partitioning*. Similar objects are grouped at the same portion of the index, such that for a given query large parts of the database can be eliminated from examination.
2. *Usage of simplified versions of the data objects*. This shrinks significantly the index size, compared to the storage requirements of the original database. Therefore, search operations on the index are much faster than on the original uncompressed data.

Consider a large database of objects, over which fast search needs to be enabled. If objects are high-dimensional, they will first be simplified in order to be stored in an index. Techniques like PCA, Fourier or Wavelet transform, etc., can be utilized at this step. For this example, in order to enhance visualization, the database objects correspond to images. The images can be simplified and represented using two features: the "number of red pixels" and the "number of blue pixels". Using these features, each image will be represented as a point on a two-dimensional space. Given these projected dimensions, two sets of images depicting underwater and wildlife images will be clustered as shown in Fig. 1. The way this new space is partitioned and searched, depends on the specifics of the utilized index.

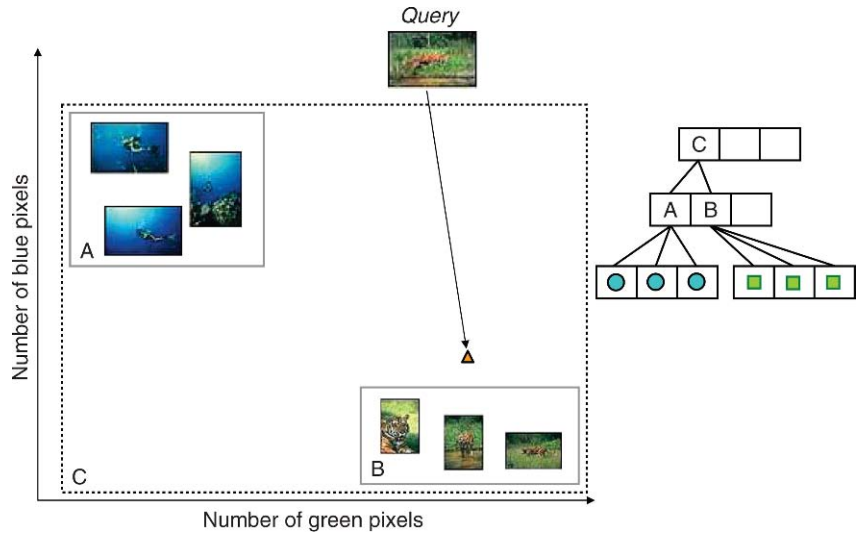
Key Applications

The following sections explain in more detail the inner-workings of widely used instances of index structures, such as the R-trees (space partitioning index) and the VP-trees (metric index).

R-Trees

The R-tree structure represents an extension of the B-tree for multiple dimensions. It has been proposed in 1984 by Antonin Guttman [7] and since then it has been utilized extensively in the database and data-mining fields.

A d-dimensional R-tree is a hierarchical structure of rectangles, with leaves and intermediate nodes. The leaves store d-dimensional hyper-rectangles



Indexing and Similarity Search. Figure 1. *Left*: Database of images and a potential two-dimensional representation. *Right*: The corresponding R-tree is a hierarchical structure of rectangles.

(or d-dimensional points) and a pointer to the object they describe. The intermediate nodes store a hyper-rectangle that completely contains the rectangles of their child-nodes, as well as pointers to the child-nodes.

In Fig. 1 a potential R-tree structure is depicted, based on the previously discussed example. Suppose now, that a query q is posed against the database of images. The query is mapped into a new point in the projected space (shown as a triangle in Fig. 1). For a *range search* one needs to find the overlapping rectangles within the requested search radius. for a *k-NN search* [10,11] one can define a `MINDIST` operator between a point and a rectangle and start traversing the tree according to the most promising path, while recording the remaining paths in a priority queue. The tree traversal can be either depth-first or breadth-first.

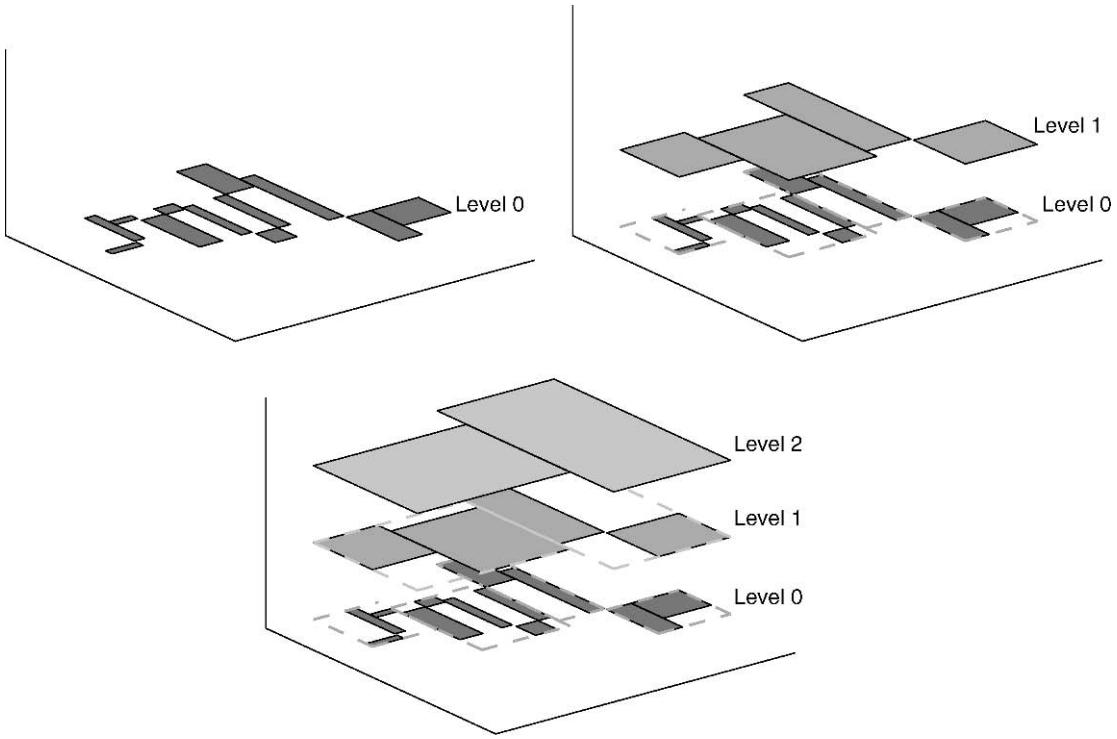
For example, if the user is seeking the 3-NN of the query q, then rectangle C will be pushed into the priority queue (see Fig. 1). It will be popped out and its children (A and B) will be pushed in and sorted according to their minimum distance to q. Subsequently, rectangle B will be examined first, because it is the closest one to the query. Its objects will be retrieved and their true distance to q shall be calculated. The search will end here, because the third closest neighbor found so far, has distance smaller than *MINDIST* (q, *Rectangle*(A)). Therefore, none of the objects of rectangle A need to be retrieved from disk.

Figure 2 demonstrates another example of the hierarchical structure of an R-tree.

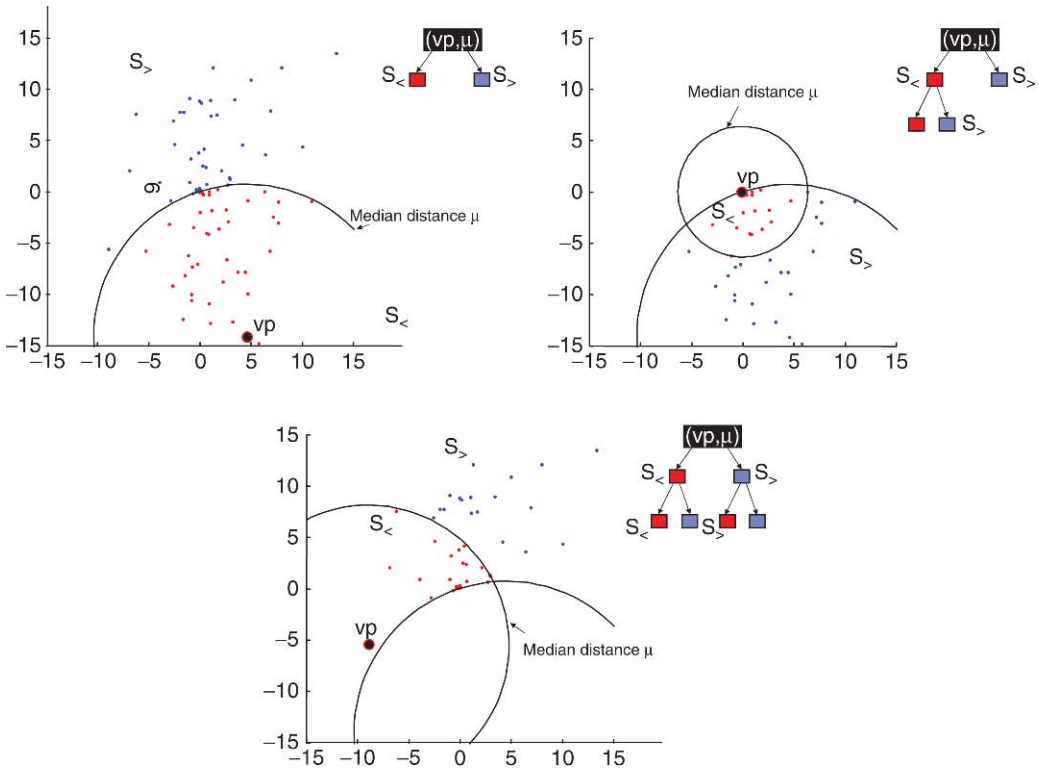
VP-Trees

According to the relevant bibliography, the pruning power of space partitioning indexing structures like R-trees, and its variants, degrades for data dimensionalities larger than 5–8. This means that at high dimensionalities the index will retrieve the majority of objects from the disk.

Metric trees exhibit better performance at larger dimensionalities (e.g., up to 20–25 dimensions). Metric trees utilize the distances between objects in order to create the tree and direct the search process. A popular instance of metric-trees are the VP-trees [4,5,12]. VP-trees partition the space based on distances to selected *vantage points* of the dataset. A tree containing the objects is constructed by recursively partitioning the dataset points into two distinct sets based on the median distance μ to the vantage point/object; the points that are closest to the vantage point ($S_<$) are stored on the left subtree, and those that are further away from the median distance ($S_>$) are directed on the right subtree. The process is repeated recursively and a different vantage point is selected for each of the remaining subsets. Figure 3



Indexing and Similarity Search. Figure 2. Bottom-up hierarchical construction of the two-dimensional R-tree.



Indexing and Similarity Search. Figure 3. Illustration of the creation of a VP-tree.

illustrates this process on two-dimensions for clarity (each point essentially represents one object).

After the tree is constructed and a query is posed, one only has to examine the proper subset based on the position of the query. Only if the query lies close to the median distance, both subsets need to be examined, otherwise one of them is discarded from examination.

Cross-references

- ▶ Curse of Dimensionality
- ▶ Data Partitioning
- ▶ Dimensionality Reduction
- ▶ Multimedia Data Indexing

Recommended Reading

1. Agrawal R., Faloutsos C., and Swami A. Efficient Similarity Search in Sequence Databases. In Proc. 4th Int. Conf. on Foundations of Data Organization and Algorithms, 1993, pp. 69–84.
2. Beckmann N., Kriegel H.-P., Schneider R., and Seeger B. The r*-tree: An efficient and robust access method for points and rectangles. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1990, pp. 322–331.
3. Bentley J. Multidimensional divide and conquer. Commun. ACM, 23(4):214–219, 1980.
4. Bozkaya T. and Özsoyoglu M. Distance-based indexing for high-dimensional metric spaces. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1997, pp. 357–368.
5. Chee Fu A.W., Chan P.M., Cheung Y.L., and Moon Y. Dynamic VP-tree indexing for N-nearest neighbor search given pair-wise distances. VLDB J., 9(2): 154–173, 2000.
6. Ciaccia P., Patella M., and Zezula P. M-tree: An efficient access method for similarity search in metric spaces. In Proc. 23th Int. Conf. on Very Large Data Bases, 1997, pp. 426–435.
7. Guttman A. R-trees: A dynamic index structure for spatial searching. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1984, pp. 47–57.
8. Keogh E., Chakrabarti K., Pazzani M., and Mehrotra S. Locally adaptive dimensionality reduction for indexing large time series databases. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 2001, pp. 151–162.
9. Nievergelt J., Hinterberger H., and Sevcik K.C. The grid file: An adaptable, symmetric multikey file structure. ACM Trans. Database Syst., 9(1):38–71, 1984.
10. Roussopoulos N., Kelley S., and Vincent F. Nearest neighbor queries. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1995, pp. 71–79.
11. Seidl T. and Kriegel H.-P. Optimal multi-step k-nearest neighbor search. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1998, pp. 154–165.
12. Yianilos P. Data structures and algorithms for nearest neighbor search in general metric spaces. In Proc. 4th Annual ACM - SIAM Symp. on Discrete Algorithms, 1993, pp. 311–321

Indexing Compressed Text

Paolo Ferragina, Rossano Venturini
University of Pisa, Pisa, Italy

Synonyms

Compressed full-text indexing; Compressed suffix array; Compressed suffix tree; Compressed and searchable data format

Definition

Given a text $T[1,n]$, the *Compressed Text Indexing* problem requires to building an indexing data structure over T that takes space close to the empirical entropy of the input text and answers queries on the occurrences of an arbitrary pattern $P[1,p]$ in T without any significant slowdown with respect to uncompressed indexes. There are three main queries: `count`(P), that returns the number of pattern occurrences in T, `locate`(P), that returns the starting positions of all pattern occurrences in T, and `extract`(i, j), that retrieves the substring $T[i, j]$.

Historical Background

String processing and searching tasks are at the core of modern web search, information retrieval (IR), data base and data mining applications. Most of text manipulations required by these applications involve, sooner or later, *searching* those (long) texts for (short) patterns or *accessing* portions of those texts for subsequent processing/mining tasks. Despite the increase in processing speed of current CPUs and memories/disks, sequential text searching long ago ceased to be a viable approach, and indexed text searching has become mandatory.

A *(full-)text index* is a data structure built over a text $T[1,n]$, drawn from an alphabet Σ of size σ, which significantly speeds up sequential searches for *arbitrary* pattern strings, at the cost of some additional space. *Suffix trees* and *suffix arrays* are the most well-known full-text indexes [5]. The *suffix tree* of a text T is a trie (or digital tree) built on all the n text suffixes $T[i, n]$, where unary paths are compacted to ensure $O(n)$ overall size. The suffix tree has n leaves, one per text suffix, and each internal node corresponds to a unique substring of T that occurs more than once. The suffix tree can count the *occ* occurrences of any pattern $P[1,p]$ in

time $O(p)$ by descending in the suffix tree according to the symbols of P, and it can locate these occurrences in optimal $O(occ)$ time by traversing the subtree of the node reached by counting. The suffix tree, however, uses much more space than the text itself because it requires $\Theta(n \log n)$ bits, whereas the text needs $n\lceil \log \sigma \rceil$ bits (logarithms are in base 2). In practice, a suffix tree requires from 10 to 20 times the text size, if carefully engineered [5].

The *suffix array* is a compact version of the suffix tree, obtained by storing in $SA[1,n]$ the starting positions of the suffixes of T listed in lexicographical order. This data structure still requires $\Theta(n \log n)$ bits in the worst case, but the constant hidden in the big-Oh notation is small in practice, namely it is no more than 4. SA can be obtained by traversing the leaves of the suffix tree, or it can be built directly in optimal linear time via ad-hoc sorting methods [5]. Since any substring of T is the prefix of a text suffix, finding all pattern occurrences boils down to finding all text suffixes that start with P. These suffixes form a lexicographic interval in SA that can be binary searched in $O(p \log n)$ time, as each comparison in the binary search requires examining up to p symbols of the pattern and of a text suffix. The time complexity can be improved to $O(p + \log n)$ by using an auxiliary data structure that doubles the space requirement of the suffix array, or it can be further reduced to $O(p + \log \sigma)$ by a proper sampling of the indexed suffixes (cfr. Suffix Trays). Once the interval $SA[sp,ep]$ containing all text suffixes starting with P has been identified, $\texttt{count}(P)$ is answered by returning the value $occ = ep - sp + 1$, and $\texttt{locate}(P)$ is answered by retrieving the entries $SA[sp]$, $SA[sp+1]$,...,$SA[ep]$.

The use of full-text indexes is not limited to (full-) text searching over one single text. It can be easily extended to multiple texts, and can also be used to support (prefix, suffix, or substring) queries over a dictionary $\mathcal{D}$ of strings having variable length. This problem is called *Dictionary Indexing* and occurs frequently in the implementation of IR and data mining applications. It can be solved via a (compressed) index built on a string S_D which is obtained by concatenating all dictionary strings, separated with a special symbol #. A prefix search for P in $\mathcal{D}$ can be implemented by counting/locating the query pattern $\#P$ in S_D; a suffix search can be implemented by searching for $P\#$ in S_D; substring searches are directly executed on S_D.

Foundations

The large space occupancy of full-text indexes has driven programmers to resort to inverted indexes to solve their searching operations on large textual datasets, and some researchers have actually concluded that the increased query power of full-text indexes has to be paid by additional storage space. Fortunately, a recent body of research showed that *compressed* full-text indexes can be designed by deploying algorithmic techniques and mathematical tools which lie at the crossing point of three distinct fields – data compression, algorithmics and databases (see e.g., [9,13,14]). Most of these indexes can be classified into two families – FM-indexes (FMI) and Compressed Suffix Arrays (CSA) – and achieve efficient query times and space close to the one achievable by the best known compressors, like gzip or bzip2. In theory, these indexes require $O(nH_k(T)) + o(n \log \sigma)$ bits of space, where $H_k(T)$ is the kth order empirical entropy of T (see Table 1). This bound is appealing because it can be sublinear in n, for highly compressible texts, and $nH_k(T)$ is the classic Information-Theoretic lower bound to the storage complexity of T by means of any kth order compressor, like gzip and bzip2 (recall that $\texttt{extract}(1,n) = T$).

The FM-Index Family

These compressed indexes were introduced by Ferragina and Manzini in [9], who devised a way to orchestrate in efficient time and space the relation that exists between the suffix array data structure and the *Burrows-Wheeler Transform* (shortly, BWT [4]). The BWT is a reversible transformation that permutes the symbols of the input string T into a new string $\texttt{bwt}(T)$ which is easier to compress, and can be computed in three steps (see Fig. 1):

1. Append at the end of T a special symbol \$ smaller than any other symbol of Σ;
2. Form a *conceptual* matrix $\mathcal{M}(T)$ whose rows are the cyclic rotations of string T\$ in lexicographic order;
3. Set string $\texttt{bwt}(T)$ to the last column L of the sorted matrix $\mathcal{M}(T)$.

Every column of $\mathcal{M}(T)$, hence also the transformed string L, is a permutation of T\$. In particular the first column of $\mathcal{M}(T)$, call it F, is obtained by lexicographically sorting the symbols of T\$ (or, equivalently, the symbols of L). Note that the sorting of the rows of

Indexing Compressed Text. Table 1. Best known complexities for the time (in big-Oh) and space (in bits) required by the main families of compressed full-text indexes

Index	Count	Locate	Extract	Space	References
FMI	p	$occ \cdot \texttt{polylog}(n)$	$\ell + \texttt{polylog}(n)$	$nH_k(T) + o(n)$	[8]
CSA	$p/\log_\sigma n + \texttt{polylog}(n)$	$occ \cdot \texttt{polylog}(n)$	$\ell/\log_\sigma n + \texttt{polylog}(n)$	$\gamma^{-1} nH_k(T) + o(n)$	[11]
LZ-INDEX	$p^2 \log p + p \log n + occ$	$occ \cdot \log n$	$\ell(1 + \epsilon^{-1}/\log_\sigma \ell)$	$(2 + \epsilon)nH_k(T) + o(n \log \sigma)$	[1]

Here $\epsilon > 0$ and $0 < \gamma < \frac{1}{3}$ are constants fixed in advance before the data structures are built; ℓ is the number of text symbols to be retrieved by `extract`; $H_k(T)$ is the k-th order empirical entropy of text T [14]. The reported complexities are worst-case and hold for $\sigma = O(\texttt{polylog}(n))$ assuming that $k \leq \alpha \log_\sigma n$ with $0 < \alpha < 1$ except for LZ-INDEX in which $k = o(\log_\sigma n)$. For more precise bounds (e.g., coefficients in $\texttt{polylog}(n)$ terms and the case $\sigma = \Omega(\texttt{polylog}(n))$ and for a thoughtful comparison of these indexes and their numerous variants, the reader is referred to [14].

$\mathcal{M}(T)$ is essentially equal to the sorting of the suffixes of T, because of the presence of the special symbol \$. This shows that: (i) symbols preceding the same substring (*context*) in T are grouped together in L, and thus give raise to clusters of nearly identical symbols; (ii) there is an obvious relation between $\mathcal{M}(T)$ and SA. Property (1) is the key for devising modern data compressors, Property (2) is crucial for designing compressed indexes and, additionally, suggests a way to compute the BWT through the construction of the suffix array of T: $L[0] = T[n]$ and, for any $1 \leq i \leq n$, set $L[i] = T[SA[i] - 1]$.

Burrows and Wheeler [4] devised two properties for the invertibility of the BWT:

1. Since the rows in $\mathcal{M}(T)$ are cyclically rotated, $L[i]$ *precedes* $F[i]$ in the original string T.
2. For any $c \in \Sigma$, the ℓth occurrence of c in F and the ℓth occurrence of c in L correspond to the *same* symbol of the string T.

As a result, the original text T can be obtained backwards from L by resorting to a function LF that maps row indexes to row indexes, and is defined as follows: if the BWT maps $T[j-1]$ to $L[i']$ and $T[j]$ to $L[i]$, then $LF(i) = i'$ (so LF implements a sort of *backward* step over T) [9]. Now, since the first row of $\mathcal{M}(T)$ is \$T, it can be stated that $T[n] = L[0]$ and, in general, $T[n-i] = L[LF^i(0)]$, for $i = 1,...,n-1$.

Starting from these basic properties, Ferragina and Manzini [9] proposed a way to combine the compressibility of the BWT with the indexing power of the suffix array. In particular, they have shown that searching operations on T can be reduced to counting queries of *single* symbols in L, now called `rank` operations. For any symbol $c \in \Sigma$ and position i in L, the query

		F		L
mississippi\$		\$	mississipp	i
ississippi\$m		i	\$mississip	p
ssissippi\$mi		i	ppi\$missis	s
sissippi\$mis		i	ssippi\$mis	s
issippi\$miss		i	ssissippi\$	m
ssippi\$missi	$\Longrightarrow$	m	ississippi	\$
sippi\$missis		p	i\$mississi	p
ippi\$mississ		p	pi\$mississ	i
ppi\$mississi		s	ippi\$missi	s
pi\$mississip		s	issippi\$mi	s
i\$mississipp		s	sippi\$miss	i
\$mississippi		s	sissippi\$m	i

Indexing Compressed Text. Figure 1. Example of Burrows-Wheeler transform for $T =$ `mississippi`. The matrix on the right has the rows sorted in lexicographic order. The output of the BWT is the column $L =$ `ipssm$pissii`.

$\texttt{rank}_c(L, i)$ returns how many times the symbol c appears in $L[1,i]$. An FM-index then consists of three key tools: a compressed representation of $\texttt{bwt}(T)$ that supports efficient `rank` queries, a small array $C[c]$ which tells how many symbols smaller than c appear in T (this takes $O(\sigma \log n)$ bits), and the so called *backward search* algorithm which carefully orchestrates the former two data structures in order to implement efficiently the `count` query. More precisely, FMI searches the pattern $P[1,p]$ backwards in p steps, which eventually identify the interval of text suffixes that are prefixed by P or, equivalently, the interval of rows of $\mathcal{M}(T)$ that are prefixed by P. This is done by maintaining, inductively for $i = p, p-1,...,1$, the interval $SA[sp_i, ep_i]$ that stores all text suffixes that are prefixed by the pattern suffix $P[i, p]$. At the beginning it is $i = p$, and so $SA[sp_p, ep_p]$ corresponds to all

suffixes which are prefixed by the last symbol $P[p]$: hence, it is enough to set $sp_p = C[P[p]] + 1$ and $ep_p = C[P[p] + 1]$. At any other step, the algorithm has inductively computed $SA[sp_{i+1}, ep_{i+1}]$, and thus it can derive the next interval of suffixes prefixed by $P[i, m]$ by setting $sp_i = C[P[i]] + \texttt{rank}_{P[i]}(L, sp_{i+1} - 1) + 1$ and $ep_i = C[P[i]] + \texttt{rank}_{P[i]}(L, ep_{i+1})$. These two computations are actually mapping (via *LF*) the first and last occurrences (if any) of symbol $P[i]$ in the substring $L[sp_{i+1}, ep_{i+1}]$ to their corresponding occurrences in *F*. (Indeed, [9] showed that any *LF* computation boils down to a `rank` query on *L*.) As a result, the backward-search algorithm requires to solve $2p$ rank queries on $L = \texttt{bwt}(T)$ in order to find out the (possibly empty) range $SA[sp, ep]$ of text suffixes prefixed by *P*. `count`(P) can be then solved by returning the value $occ = ep_1 - sp_1 + 1$.

Conversely, `locate` and `extract` need some extra information about the underlying suffix array, this impacts onto the space occupancy of the FMI. Recall that `locate`(P) requires to return, for any $i \in [sp, ep]$, the position $pos(i) = SA[i]$. For space reasons *SA* cannot be stored explicitly so that, for a fixed parameter $\mu = \lceil \log^{1+\epsilon} n \rceil$, FMI samples the rows of $\mathcal{M}(T)$ which correspond to text suffixes that start at positions of the form $1 + j \cdot \mu$. Each such pair $\langle$row, position$\rangle$ is stored explicitly in a data structure $\mathcal{S}$ that supports membership queries in constant time (on the *row*-component). Now, given a row index *i*, the value $pos(i)$ can be derived immediately from $\mathcal{S}$, if *i* is a sampled row; otherwise, the algorithm computes $j = LF^t(i)$, for $t = 1,2,...$, until *j* is a sampled row. In this case, $pos(i) = pos(j) + t$. The sampling strategy ensures that a row in $\mathcal{S}$ is found in at most μ iterations, and thus the *occ* occurrences of the pattern *P* can be located via $O(\mu \cdot occ)$ `rank` queries. The algorithm for `extract`(i,i') requires a similar approach and takes no more than $(i' - i + \mu + 1)$ rank queries.

The net result is that the space and time complexities of FMI depend on the value μ and on the performance guaranteed by the data structure used to compute rank queries on the BWT-string. The extra space required by the data structures added to support `locate` and `extract` is bounded by $O((n \log n)/\mu)$ bits, which is $o(n)$ whenever $\epsilon > 0$. The real challenge thus consists of representing `bwt`(T) in a compressed form and answering efficiently rank queries over it. Actually, all implementations of FMI differentiate themselves by the strategy used to solve this problem, as the alphabet size grows. Today, the literature offers many solutions, the most efficient ones are summarized below.

Lemma 1 *Let* $T[1,n]$ *be a string over an alphabet of size* σ*, and let* $L = \texttt{bwt}(T)$.

1. For $\sigma = O(\texttt{polylog}(n))$, *there exists a data structure which supports* `rank` *queries on L in* $O(1)$ *time using* $nH_k(T) + o(n)$ *bits of space, for any* $k \leq \alpha \log_\sigma n$ *and* $0 < \alpha < 1$, *and retrieves any symbol of L in the same time bound* [10, *Theorem 5*].
2. *For general* Σ, *there exists a data structure which supports* `rank` *queries on L in* $O(\log \log \sigma)$ *time, using* $nH_k(T) + n\ o(\log \sigma)$ *bits of space, for any* $k \leq \alpha \log_\sigma n$ *and* $0 < \alpha < 1$, *and retrieves any symbol of L in the same time bound* [2, *Theorem 4.2*].

By plugging this Lemma into the FMI data structure, one derives a compressed full-text index that supports efficiently the three full-text queries – namely, `count`, `locate`, `extract`– and occupies space approaching the *k*th order empirical entropy of *T* (see Table 1).

In practice, there are various implementations of FMI, whose engineering choices mainly refer to the way the `rank`-data structure built on `bwt`(T) is compressed and scales with the alphabet size of the indexed text. The site Pizza&Chili (see below) reports several implementations for FMI that mainly boil down to the following trick: `bwt`(T) is split into blocks (of equal or variable length) and values of $\texttt{rank}_c$ are precomputed for all block beginnings and all symbols $c \in \Sigma$. A query $\texttt{rank}_c(L, i)$ is answered by summing up the answer available for the beginning of the block that contains $L[i]$ plus the rest of the occurrences of *c* in that block – they are obtained either by sequentially decompressing the block or by using a proper compressed data structure built on it (e.g., the Wavelet Tree of [12]). The former approach favors compression, the latter favors query speed.

The CSA family. These compressed indexes were introduced by Grossi and Vitter [13], who showed how to compactly represent the suffix array *SA* in O(n log σ) bits and still be able to access any of its entries in efficient time. Their solution is based on a function Ψ, which is the inverse of the function LF introduced for BWT:

$$\Psi(i) = \begin{cases} i' \text{ such that } SA[i'] = SA[i] + 1 & (\text{if } SA[i] > n) \\ i' \text{ such that } SA[i'] = 1 & (\text{if } SA[i] = n) \end{cases}$$

In other words, $\Psi(i)$ refers to the position in the suffix array of the text suffix that follows $SA[i]$ in T, namely, the text suffix which is one-symbol shorter. The compact storage of SA proposed by Grossi and Vitter is based on a hierarchical decomposition that deploys Ψ. To represent $SA_0 = SA$ they use three vectors: B_0, Ψ_0 and SA_1. The binary vector $B_0[1,n]$ marks the entries of SA_0 which are even (suffixes). The vector $\Psi_0[1,\lceil n/2 \rceil]$ stores the values Ψ(i) for which $SA[i]$ is odd (hence $B_0[i] = 0$). The vector $SA_1[1,\lceil n/2 \rfloor]$ is a "halved" version of SA_0, in that it contains the even elements of SA_0 divided by 2. Surprisingly enough, these three vectors suffice to retrieve any entry SA[i]. Of course, it is easy to determine whether $SA[i]$ is even or odd by simply looking at $B_0[i]$. If $SA[i]$ is odd, the following suffix $SA[i] + 1 = SA[\Psi(i)]$ is even, and its suffix-array position can be determined as $\Psi(i) = \Psi_0(\texttt{rank}_0(B_0,i))$. If $SA[i]$ is even, it is enough to look at its *halved* value stored at $SA_1[\texttt{rank}_1(B_0,\text{i})]$. The three vectors Ψ_0, B_0 and SA_1 form the first level of the hierarchical decomposition of SA. This idea is applied recursively on SA_1 which is replaced by three other vectors: Ψ_1, B_1 and SA_2. This goes on until SA_h can be represented within $O(n)$ bits, namely when $h = \lceil \log \log n \rceil$. Accessing $SA[i]$ takes h time. By storing the text T, in additional $n\lceil \log \sigma \rceil$ bits, one can search for a pattern P via the classic binary-search, now on the compacted SA. Grossi and Vitter proposed to store vectors B in compressed form via proper `rank`-data structures (see [14] and references therein), and deployed the *piecewise increasing property* for Ψ – namely, if $T[SA[i]] = T[SA[i+1]]$, then $\Psi(i) < \Psi(i+1)$ – to store each level of Ψ within $\frac{1}{2} n \log \sigma$ bits, still preserving constant time lookup to any level of Ψ. Other time/space tradeoffs are possible by using different numbers of levels. Essentially, not all the levels are represented and the function Ψ is used to jump from one represented level to the next represented one.

Recently, CSA has been the subject of two main improvements. The first one, due to Sadakane [16], showed that the original text T can be replaced with a binary vector F such that $F[i] = 1$ iff the first symbol of the suffixes $SA[\text{i} - 1]$ and $SA[i]$ differs. Since the suffixes in SA are lexicographically sorted, one can determine the first symbol of any suffix in constant time by just executing *a* $rank_1$ query on F. This fact, combined with the retrieval of Ψ's values in constant time, allows comparing any suffix with the searched pattern $P[1,p]$ in time $O(p)$. Sadakane also provided an improved representation for Ψ achieving $nH_0(T)$ bits. Theoretically, the best variant of CSA is due to Grossi, Gupta and Vitter [12] who devised some further structural properties of Ψ that allow to come close to $nH_k(T)$ bits, still preserving the previous time complexities for all full-text queries (see Table 1). Practically, the best implementation of the CSA is the one proposed by Sadakane that actually does not use the hierarchical decomposition above, but orchestrates a compact representation of the function Ψ together with the backward search and the sampling strategy of the FMI family. This *hybrid* index is among the fastest compressed indexes to count and locate pattern occurrences over highly-compressible data.

Other Compressed Indexes

Previous families of compressed indexes based their search on the implicit or explicit availability of the suffix array data structure. Recent years have seen the design of several other approaches, the two most notable ones are the *LZ-index*, proposed by Navarro, and the *Compressed Suffix Tree*, devised by Sadakane and then improved by many other authors. The former index bases its design on the parsing of the text T via the LZ78-compression scheme, and then enriches its output by additional data structures that support efficient searches over the parsed phrases. By properly orchestrating LZ78-parsing with compressed dictionary data structures, [1] achieved interesting search and entropy-based space bounds which are not competitive theoretically with the ones obtained by FMI and CSA indexes (see Table 1) but are, nonetheless, fast in practice. As far as the compressed suffix-tree is concerned, it is worth noticing that the compression of this data structure is obtained by properly orchestrating succinct tree and succinct array encodings [15]. The total space is the one required by the CSA built on T plus no more than $6n + o(n)$ bits; all known suffix-tree operations are supported with a maximum slowdown of $O(\log n)$ time with respect to the uncompressed suffix tree.

Key Applications

Compressed full-text indexes might be used at the core of modern web search, IR, data base and data mining applications because, as Knuth observed in the Art of Computer Programming (vol. 3): "*space optimization is closely related to time optimization in a disk memory*". Data compression can not only squeeze the space

overhead of an index, but also improve its speed, as remarked earlier. Several authors [3,5,11,17,18] have recently addressed these issues in various settings but, nonetheless, there is much more room for theoretical and practical improvements.

Future Directions

An open challenge concerning compressed indexes is to fasten their `locate` queries in order to achieve the optimal $O(occ)$ time bound. The best known result is due to Ferragina and Manzini [9]: each occurrence is located in constant time, and the index takes $O(nH_k(S) \log^{\in} n) + o(n \log \sigma \log^{\in} n)$ bits, where $\in$ is any positive constant. This bound has the extra log-factor in front of the entropy term! Therefore, it is natural to ask: Is there a full-text index achieving $O(p + occ)$ query time and $O(nH_k(S)) + o(n \log \sigma)$ bits of space occupancy in the worst case? This result would be *provably better* than any known uncompressed full-text index.

Another interesting open problem consists of designing a compressed full-text index which is disk-aware or, better, memory-oblivious in that it scales optimally over all memory levels available in a modern PC. The above data structures are compressed, but their overall size may span many memory levels so that issues pertaining to proper *arrangement of data* and properly *structured algorithmic computations* come into play. The most attractive disk-aware index is the String B-tree [7]; whereas the best cache-oblivious index is the COSB-tree [3,8]. Unfortunately the former is uncompressed, whereas the latter uses a compression heuristic which does not guarantee entropy-bounds in the worst case. It would be therefore valuable, also in practice, to devise a compressed index that combines the I/O-efficiency of the (cache oblivious) String B-tree with the space efficiency of the compressed full-text indexes discussed in this entry. Some preliminary results have been devised in [8], but the ultimate goal has yet to be achieved.

Experimental Results

Site Pizza&Chili [6] provides a full experimental comparison among the major implementations of compressed indexes. The experiments mainly show that these indexes can compress a text within 40–80% of its original size, and support searches for 20,000–50,000 patterns of 20 chars each within a second, locate about 100,000 pattern occurrences per second, and decompress text symbols at a rate of about 1 MB/s. The compressed indexes are from one (`count`) to three (`locate`) orders of magnitudes slower than what one can achieve with a plain suffix array, at the benefit of using up to 18 times less space. This slowdown is due to the fact that search operations in compressed indexes access the memory in a non-local way thus eliciting many cache/IO misses, with a consequent degradation of the overall time performance. Nonetheless compressed indexes achieve a (search/extract) throughput which is significant and may match the efficiency specifications of most software tools running on commodity PCs. Recently, Ferragina and Venturini [11] provided a comparison among classic and compressed indexes for the Dictionary Indexing Problem showing that, in this case, compressed indexes may be faster than classic IR approaches.

Data Sets

Calgary Corpus (http://links.uwaterloo.ca/calgary.corpus.html)
Canterbury Corpus (http://corpus.canterbury.ac.nz)
Pizza&Chili Corpus (http://pizzachili.di.unipi.it or http://pizzachili.dcc.uchile.cl) see also [18]

URL to Code

Site Pizza&Chili (http://pizzachili.di.unipi.it or http://pizzachili.dcc.uchile.cl) collects implementations of the major compressed text indexes, and various tools and datasets to test them.

Cross-references

- ▶ Managing Compressed Structured Text
- ▶ Suffix Trees
- ▶ Text Compression
- ▶ Text Index Compression
- ▶ Text Indexing & Retrieval
- ▶ Text Indexing Techniques
- ▶ Text Representation
- ▶ XML Compression

Recommended Reading

1. Arroyuelo D., Navarro G., and Sadakane K. Reducing the space requirement of LZ-index. In Proc. 17th Annual Symposium on Combinatorial Pattern Matching, 2006, pp. 319–330.
2. Barbay J., He M., Munro J.I., and Srinivasa Rao S. Succinct indexes for string, binary relations and multi-labeled trees. In

I

Proc. 18th Annual ACM -SIAM Symp. on Discrete Algorithms, 2007, pp. 680–689.
3. Bender M.A., Farach-Colton M., and Kuszmaul B.C. Cache-oblivious string B-trees. In Proc. 25th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems, 2006, pp. 233–242.
4. Burrows M. and Wheeler D. A block sorting lossless data compression algorithm. Technical Report 124, Digital Equipment Corporation, 1994.
5. Ferragina P. String Search in External Memory: Data Structures and Algorithms, In Handbook of Computational Molecular Biology, Chapman & Hall, London, 2005.
6. Ferragina P., González R., Navarro G., and Venturini R. Compressed Text Indexes: From Theory to Practice, J. Exp. Algorithmics, 13:1.12–1.31, 2009.
7. Ferragina P. and Grossi R. The String B-tree: A new data structure for string search in external memory and its applications. J. ACM, 46(2):236–280, 1999.
8. Ferragina P., Grossi R., Gupta A., Shah R., and Vitter J.S. On searching compressed string collections cache-obliviously. In Proc. 27th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems, 2008, pp. 181–190.
9. Ferragina P. and Manzini G. Indexing compressed text. J. ACM, 52(4):552–581, 2005.
10. Ferragina P., Manzini G., Mäkinen V., and Navarro G. Compressed representations of sequences and full-text indexes. ACM Trans. Algorithms, 3(2), 2007.
11. Ferragina P. and Venturini R. Compressed permuterm index. In Proc. 33rd Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 2007, pp. 535–542.
12. Grossi R., Gupta A., and Vitter J.S. High-order entropy-compressed text indexes. In Proc. 14th Annual ACM-SIAM Symp. on Discrete Algorithms, 2003, pp. 841–850.
13. Grossi R. and Vitter J.S. Compressed suffix arrays and suffix trees with applications to text indexing and string matching. SIAM J. Comput., 35(2):378–407, 2005.
14. Navarro G. and Mäkinen V. Compressed full-text indexes. ACM Comput. Surv., 39(1), 2007.
15. Sadakane K. Compressed suffix trees with full functionality. Theory Comput. Syst., 41(4):589–607, 2007.
16. Sadakane K. New text indexing functionalities of the compressed suffix arrays. J. Algorithms, 48(2):294–413, 2007.
17. Sadakane K. Succinct data structures for flexible text retrieval systems. J. Discrete Algorithms, 5(1):12–22, 2007.
18. Tam S.L., Wong C.K., Lam T.W., Sung W.K., and Yiu S.M. Compressed indexing and local alignment of DNA. Bioinformatics, 24(6):791–797, 2008.

Indexing for Online Function Approximation

► Database Techniques to Improve Scientific Simulations

Indexing for Similarity Search

► High Dimensional Indexing

Indexing Granularity

► Indexing Units

Indexing Historical Spatio-Temporal Data

Mohamed F. Mokbel[1], Walid G. Aref[2]
[1]University of Minnesota, Minneapolis, MN, USA
[2]Purdue University, West Lafayette, IN, USA

Synonyms

Indexing the past; Historical spatio-temporal access methods; Trajectory indexing

Definition

Consider an object O that reports to a database server two consecutive locations $P_0 = (x_0,y_0)$ and $P_1 = (x_1,y_1)$ at times t_0 and t_1, respectively. The database server has no idea about the exact locations of object O between t_0 and t_1. To be able to answer queries regarding the user location at any time, the database server interpolates the two accurate locations through a trajectory that connects P_0 and P_1 through a straight line. While object O keeps sending location samples, the database server keeps accumulating set of consecutive trajectory lines that represent the historical movement of object O. Indexing historical spatio-temporal data includes dealing with such large numbers of trajectories. The main idea is to organize past trajectories in a way that supports historical spatial, temporal, and spatio-temporal queries.

Historical Background

The rapid increase in spatio-temporal applications calls for new auxiliary indexing structures. A typical spatio-temporal application is one that tracks the behavior of moving objects through location-aware devices (e.g., GPS). Through the last two decades, many spatio-temporal access methods were developed.

Spatio-temporal access methods focus on two orthogonal directions: (i) Indexing the past i.e., historical location data, (ii) Indexing the current and predicted future positions. This entry focuses on the former direction. A first approach to support spatio-temporal data is to extend existing spatial data (e.g., R-tree [4]) to support the temporal dimension. However, as the temporal dimension has distinct properties from other spatial dimensions, these approaches are later modified to give special attention to the temporal dimension.

Foundations

The main challenge in indexing historical spatio-temporal data is that the size of the history is continuously increasing over time. Consider moving objects that continuously send their positions. Keeping track of all updates is almost infeasible. Two approaches are used to minimize the history size: (i) *Sampling*. The stream of data is sampled at certain time positions. Linear interpolation may be used between sample points to form trajectory lines. (ii) *Update on change only*. Moving objects send information only when their data is changed (e.g., change in speed or direction). In general, spatio-temporal indexing methods for historical data can be categorized into the following three categories:

1. Three-Dimensional Structures

This category augments existing spatial access methods (e.g., the R-tree [4]) to support the newly introduced temporal dimension. Examples of this category include:

- *RT-tree* [15]: The RT-tree combines the foundation of the R-tree as a spatial access method and the TSB-tree [7] as a temporal access method. In the RT-tree, a new entry is added to the regular R-tree that indicates the start and end times of the current object. An RT-tree entry has the form (*id*, *MBR*, t_s, t_e), where *id* is the object identifier, *MBR* is the objects minimum bounding rectangle, and t_s and t_e give the time interval in which this object is valid. The RT-tree supports spatial queries as efficient as the regular R-tree. However time slice queries and interval queries may span the whole tree.
- *3D R-tree* [13]: The 3D R-tree treats time as yet another dimension in addition to the spatial dimensions. The main idea is to avoid discrimination between spatial and temporal queries. The 3D R-Tree supports both the temporal and spatial queries, although with performance drawbacks. A main drawback is that timeslice queries are no longer dependent on the live entries at the query time, but on the total number of entries in the history.
- *STR-tree* [9]: The STR-Tree is an extension of the R-Tree, with a different insert/split algorithm. Leaf nodes has the form (*id*, t_{id}, *MBR*, *o*) where t_{id} is the trajectory identifier and *o* is the orientation of this trajectory in the MBR. The main idea is to keep spatial closeness and partial trajectory preservation by trying to keep line segments belonging to the same trajectory together while keeping spatial closeness as the R-Tree. A parameter *p* is introduced to balance between spatial properties and trajectory preservation. *p* indicates the number of levels reserved for trajectory preservation. When inserting a new line segment the goal is to insert it as close as possible to its predecessor in the trajectory within *p* levels. A smaller *p* decreases the trajectory preservation, while increasing the spatial closeness.

2. Overlapping Two-Dimensional Structures

This category separates between the spatial and temporal dimensions. The main idea is to use a separate spatial index for each time instance. Then, a temporal index is used to index the spatial indexes. To reduce the storage overhead, consecutive spatial indexes may overlap to avoid storing multiple instances of objects that are not frequently changed over time. Examples of this category include:

- *MR-tree* [15]: The MR-tree employs the idea of overlapping B-trees [2] in the context of the R-tree. The main idea is to avoid the storage overhead of having separate R-trees for each timestamp. The saving in storage is achieved by not storing the common objects among consecutive R-trees. Instead, links from different roots point to the same nodes where all the node entries keep their values over the different timestamps. This idea is perfect in the case of a time slice query. The search is directed to the appropriate root, and then a spatial search is performed using the R-tree. However, the performance of time window queries is not efficient. Also, one major drawback is that many entries can be replicated. Consider the case that only one node entry is changed over two

consecutive timestamps, then all other node entries need to be replicated in two consecutive R-trees.

- *HR-tree* [8]: The Historical R-tree (HR-tree) is very similar to the MR-tree. The HR-tree has a concrete algorithm and implementation details of using the overlapping B-tree [2] in the context of the R-tree. The same idea of overlapping trees is applied in the context of quadtrees, where it results in *overlapping quadtrees* [14].
- *HR+-tree* [11]: The HR+-tree is designed mainly to avoid the replication of some entries in the HR-tree. The main reason for having duplicate entries in the HR-tree is that the HR-tree has a condition that any node can contain only entries that belong to the same root, i.e., ones that have the same timestamp. The HR+-tree relaxes this condition by allowing entries from different timestamps to reside in the same node. However, the parent of this node in each R-tree has only access to the entries that belong to the parent's timestamp. In other words, a node may have multiple parents, where each parent has access only to a different part of the node.
- *MV3R-tree* [12]: The MV3R-tree is based mainly on the multi-version B-tree (MVB-tree) [11]. The main idea is to build two trees, an MVR-tree to process timestamp queries, and a 3D R-tree to process long interval queries. Short interval queries are optimized to check which tree is to be used based on a threshold value.

3. Trajectory Indexing

This category is radically different from other categories where the main concern is to support trajectory-oriented queries. On the other side, spatial queries are not well supported. Examples of this category include:

- *TB-tree* [9]: The Trajectory-bundle tree (TB-tree) is an R-tree-like structure that strictly preserves trajectories. A leaf node can only contain segments belonging to the same trajectory. As a drawback, line segments of different trajectories that lie spatially close will be stored in different nodes. The TB-tree grows from left to right. The left-most leaf node is the first inserted node and the right-most leaf node is the last inserted one. The TB-tree is an extension of the STR-tree to handle only trajectories.
- *SETI* [3]: The Scalable and Efficient Trajectory Index (SETI) partitions the spatial dimension into static, non-overlapping partitions. The main observation is that the change of the spatial dimension is limited while the temporal dimension is continuously evolving. Thus, the spatial dimensions are partitioned statically. Within each partition the trajectory segments are indexed using an R-tree. Using a good partitioning function results in having line segments of the same trajectory stored in the same partition. Thus, trajectory preservation is achieved by minimizing the effect of the spatial dimensions in the R-tree. A segment that crosses the boundary of two spatial partitions is clipped and is stored twice in both partitions. This may lead to duplicates in the query result.
- *The SEB-tree* [10]: The Start/End timestamp B-tree (SEB-tree) has an idea similar to SETI, where the space is partitioned into zones that may be overlapped. Each zone is indexed using the SEB-tree that considers only the start and end timestamps of the moving objects. Each moving object is hashed to its zone. A key difference over SETI is that there are no trajectories. Instead only two-dimensional points are indexed. By having the spatial zoning partitioning, two-dimensional points that belong to similar trajectories are kept together.

Key Applications

Moving Object Databases

The wide spread of location-detection devices (e.g., GPS-like devices and cellular phones) along with the recent advances in mobile computing enable the so-called location-based environments. In such environments, a large number of moving objects continuously send their location information to a location-based database server. Storing and indexing past location information enable new types of queries that include: "*What are the vehicles near my shop yesterday between 7:00 and 8:00* A.M. *yesterday*" and "*At what time yesterday, my car was within one mile of a fast food restaurant.*" Indexing historical spatio-temporal data is a major module for efficient query retrieval for moving object databases.

Recommended Reading

1. Becker B., Gschwind S., Ohler T., Seeger B., and Widmayer P. An asymptotically optimal multiversion B-tree. VLDB J. 5(4):264–275, 1996.
2. Burton F.W., Kollias J.G., Matsakis D.G., and Kollias V.G. Implementation of overlapping B-trees for time and space

efficient representation of collections of similar files. The Computer Journal, 33(3):279–280, 1990.
3. Chakka V.P., Everspaugh A., and Patel J.M. Indexing large trajectory data sets with SETI. In Proc. 1st Biennial Conf. on Innovative Data Systems Research, 2003.
4. Guttman A. R-Trees: A dynamic index structure for spatial searching. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1984, pp. 47–57.
5. Hadjieleftheriou M., Kollios G., Tsotras V.J., and Gunopulos D. Efficient indexing of spatiotemporal objects. In Advances in Database Technology, Proc. 8th Int. Conf. on Extending Database Technology, 2002, pp. 251–268.
6. Kollios G., Tsotras V.J., Gunopulos D., Delis A., and Hadjieleftheriou M. Indexing animated objects using spatiotemporal access methods. IEEE Trans. Knowledge and Data Eng., 13(5):758–777, 2001.
7. Lomet D.B. and Salzberg B. Access methods for multiversion data. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1989, pp. 315–324.
8. Nascimento M.A. and Silva J.R.O. Towards historical R-trees. In Proc. 1998 ACM Symp. on Applied Computing, 1998, pp. 235–240.
9. Pfoser D., Jensen C.S., and Theodoridis Y. Novel approaches in query processing for moving object trajectories. In Proc. 26th Int. Conf. on Very Large Data Bases, 2000, pp. 395–406.
10. Song Z. and Roussopoulos N. SEB-tree: an approach to index continuously moving objects. In Proc. 4th Int. Conf. on Mobile Data Management, 2003, pp. 340–344.
11. Tao Y. and Papadias D. Efficient historical R-trees. In Proc. 13th Int. Conf. on Scientific and Statistical Database Management, 2001, pp. 223–232.
12. Tao Y. and Papadias D. MV3R-Tree: a spatio-temporal access method for timestamp and interval queries. In Proc. 27th Int. Conf. on Very Large Data Bases, 2001, pp. 431–440.
13. Theodoridis Y., Vazirgiannis M., and Sellis T. Spatio-temporal indexing for large multimedia applications. In Proc. Int. Conf. on Multimedia Computing and Systems, 1996, pp. 441–448.
14. Tzouramanis T., Vassilakopoulos M., and Manolopoulos Y. Overlapping linear quadtrees: a spatio-temporal access method. In Proc. 6th Int. Symp. on Advances in Geographic Inf. Syst., 1998, pp. 1–7.
15. Xu X., Han J., and Lu W. RT-Tree: An improved R-tree indexing structure for temporal spatial databases. In Proc. Int. Symp. on Spatial Data Handling, 1990, pp. 1040–1049.

Indexing Metric Spaces

Pavel Zezula, Michal Batko, Vlastislav Dohnal
Masaryk University, Brno, Czech Republic

Synonyms

Distance indexing

Definition

Metric space indexing is closely related to the recent digitization revolution where almost everything that one can see, hear, read, write or measure is available in digital form. Unlike traditional attribute-like data types such as numbers and strings of sortable domains, instances of these new data types are complex, and the only measure of comparison to apply is a sort of *similarity*. Such a situation implies an application of the *query-by-example* search paradigm where the database is searched for objects that are *near* the example object, also called the *query object*. A useful abstraction of this similarity is to see it as mathematical *metric space* [7]. The problem of organizing and searching large datasets of complex objects can then be considered from the perspective of generic or arbitrary metric spaces, sometimes labeled *distance spaces*. In general, the search problem can be described as follows:

► *Let D be a domain, d a distance measure on D, and (D,d) a metric space. Given a set $X \subseteq D$ of n elements, preprocess or structure the data so that similarity queries are answered efficiently.*

From a practical point of view, X can be seen as a file (a dataset or a collection) of objects that takes values from domain D, with d as the proximity measure, i.e., the distance function defined for an arbitrary pair of objects from D. Though several types of similarity queries exist and others are expected to appear in the future, the basic types are known as the *similarity range* query whose results are constrained by a maximum distance expressed as the query radius, and the *nearest neighbor(s)* query bounding the query response size by the maximal number of closest objects $k \leq n$.

The metric space indexing approach significantly extends the scope of traditional search approaches. It supports the execution of similarity queries without obviating the traditional attribute-like searching. The distance-searching approach to indexing is thereby highly *extensible*.

Historical Background

The need for similarity searching in collections of metric objects was recognized quite early and the history documented by numerous citations is nicely summarized in recent surveys such as [2, 6, 8, 10]. The first known proposal was the *Burhard-Keller tree* from 1973, which was later extended into the *Fixed queries tree* and the *Fixed queries array* techniques, all of which

specialize in discrete distance functions only. By recursive applications of the basic *ball* and *generalized-hyperplane* partitioning principles, the *Vantage point tree* and the *Generalized-hyperplane tree* were defined. The advantage of exploiting pre-computed distances to speed up retrieval was first recognized in the 1980's and formalized as the *Approximating and Eliminating Search Algorithm*, AESA. However, all of these early attempts were only considering small data collections stored in the main memory.

The importance of processing large datasets stored on disk memory was reflected for the first time in the M-tree [3]. It is a balanced metric tree supporting a disk-oriented storage, trying to optimize the I/O as well as the CPU costs of query processing by means of a synergistic combination of several elementary strategies. It can be considered as a generalization of the R-tree and the B-tree. The M-tree's success has been demonstrated by numerous modifications and extensions concerning mainly insertion algorithms, mechanisms of splitting, and strategies for query execution. The most significant extension is the Slim-Tree [9]. As an orthogonal approach to tree partitioning, the D-index [4] is constructed as a multi-tier hashing structure consisting of search-separable sets of objects on each tier organized in directly-accessible buckets. The structure supports easy insertion and its search costs are bounded because no more than one bucket needs to be accessed at each level for range queries up to a pre-defined value of the search radius. Other important hybrid approaches are known as the *Multi vantage point tree*, the *Geometric near-neighbor access tree*, and the *Spatial approximation tree*.

The similarity search in metric spaces is generally expensive and state-of-the-art access methods designed for a single computer do not provide sufficient performance for highly interactive applications that access large collections of data. The *approximate similarity search* techniques offer greatly improved efficiency vis à vis precise similarity searching at the expense of some imprecision in the results. The use of the approximate similarity search is mainly justified by the following two observations: (i) the similarity between objects is often subjective, thus very difficult to express as a unique rigorous function; (ii) similarity search processes are intrinsically iterative – users typically issue several similarity queries to the search system, possibly reusing the previous query results to express new ones. Practical experiments with approximate similarity search techniques register performance improvements up to two orders of magnitude compared to the precise evaluation of the same queries. Other techniques overcome the *scalability* problem by distributing partitions of data and by executing similarity queries in parallel using structured peer-to-peer networks. Four such peer-to-peer techniques are compared in [1].

Foundations

In the last decade, metric space indexing for similarity search has prompted major research efforts resulting in a number of specific theories, techniques, implementation paradigms and analytic tools aimed at making the distance-based approach viable. There are basically two ways to solve the indexing problem: (i) to transform the metric space so that a working solution from another domain can be applied; or (ii) exploit the general properties of metric functions directly. This entry focuses on native indexing.

Due to the lack of a global ordering or a fixed position as in coordinate spaces, elementary *partitioning* in generic metric spaces is defined with respect to either one or two fixed reference objects taken from D. The first principle, known as the *ball partitioning*, uses one given object as a center and a specific radius as a boundary to form a sphere (ball). The set of data is then divided into two disjoint subsets: objects inside the ball and those outside the ball. The other principle partitions the space using two given reference objects where the distance establishes the borderline of the *generalized hyperplane partitioning* – objects that are closer to the first reference object than to the second form one partition, while the rest of the objects belong to the second partition.

In addition to the partitioning principles, strategies for query execution play another important role in search structures because they can significantly influence the efficiency of answering queries. Efficient search algorithms for similarity range and nearest neighbor queries have been defined for metric indexes, based on the *branch and bound* strategy in principle. There are even algorithms for evaluating *incremental nearest neighbor* queries, as well as *similarity joins*. Approximate similarity searches have also been studied thoroughly in order to improve performance. The general idea of approximation algorithms is to relax some constraints of the "precise" similarity search to reduce search costs, as measured in disk accesses and/or the number of distance computations. This inevitably means that *false hits* or *false dismissals* may occur.

Since the computational complexity of some metric distance functions can be quite high, it is very important for metric indexes to limit the number of distance computations as much as possible. The rationale behind such strategies is to exploit already-evaluated distances between some objects while properly applying the metric space postulates – namely the *triangle inequality*, *symmetry*, and *non-negativity* – to determine bounds on distances between other objects. The most elementary case can be explained on three objects a,b, and $c \in D$. Provided the distances $d(a, b)$ and $d(b, c)$ are known, it is clear that the distance between a and c cannot be longer than their sum, i.e., $d(a, c) \leq d(a, b) + d(b, c)$. Alternatively, if the known distances are $d(a, c)$ and $d(b, c)$, the distance $d(a,b)$ must be at least as long as their difference, that is $d(a, b) \geq |d(a, c) - d(b, c)|$. Several *bounding strategies* originally proposed in [5] are summarized in [6]. These techniques represent the foundations of the general pruning rules that are employed, in a specific form, in practically all index structures for metric spaces. They can be considered as the basic formal background of metric indexes.

Key Applications

Treating data collections as metric objects is advantageous in that many data classes and information-seeking strategies conform to the metric view. Accordingly, a single metric indexing technique can be applied to many specific search problems quite different in nature. In this way, the important *extensibility* property of indexing structures is satisfied by default. An indexing scheme that allows various forms of queries, or which can be modified to provide additional functionality, is of more value than an indexing scheme otherwise equivalent in power or even better in certain respects, but which cannot be extended.

Distance functions of metric spaces represent a way of quantifying the closeness of objects in a given domain. The distance functions are often tailored to specific applications or a class of possible applications. In practice, the distance functions are specified by domain experts, however, no distance function restricts the range of query types that can be asked with this metric. The distance functions can be *discrete* or *continuous*. Another classification is possible according to the type of domain of compared objects. For example, the *Minkowski distance* functions form a whole family of metric functions, designated as the L_p metrics, because the individual cases depend on the numeric parameter p. These functions are defined on n-dimensional vectors, where the L_1 metric is known as the *Manhattan distance* (also the *City-Block distance*), the L_2 distance denotes the well-known *Euclidean distance*, and the L_∞ is called the *maximum distance*, the *infinite distance* or the *chessboard distance*. The closeness of sequences of symbols (strings) can be effectively measured by the *edit distance*, also called the *Levenshtein distance*. In order to identify common molecular subsequences, the Smith-Waterman algorithm has been proposed. In case the processed objects are sets, the *Jaccard's coefficient* or the *Hausdorff distance* can be applied. More details about metric functions can be found in [5], but the set of metric distance functions is still growing, as evidenced by the recent *Earth Mover's Distance*, for example.

Experiments show that metric indexing techniques are very competitive even in specific data domains traditionally supported by specialized index structures, the R-tree for *multi-dimensional data* and the B-tree for one-dimensional *attribute-like data* being two such examples. The number and variety of metric distance functions determine the range of applications which can apply the metric indexing approach. It has proved useful for *multimedia data features* since most of the standard MPEG7 image descriptors are metrics. But the application of metric indexing involves a lot of diverse fields such as *spatial databases*, *computer graphics and vision*, *game programming*, *geographic information systems*, *computational geometry*, *computer-aided design*, *robotics*, *computational biology* and many others.

Future Directions

The biggest challenge of a perspective search paradigm is to find self-organized solutions that evolve in time and still scale into the expected data volumes. In general, the self-organizing systems build and maintain an internal knowledgebase in response to the flow of data and queries. Such initiative must be based on solid theoretical backgrounds to avoid possible quick but ad-hoc solutions which will sooner or later fail due to the absence of rigorous definitions and unpredictable behavior. The research needs to go beyond the capabilities of traditional computer science and should try to find an inspiration in other scientific areas. The social sciences offer a promising alternative, especially advances in online social networking.

Experimental Results

Experimental results demonstrate the extensibility of the metric space approach since most of the proposed indexing tools work for any metric distance function. The performance is primarily influenced by the distribution of distances, the smaller the variation of distances, the more expensive the search becomes. At the same time, the scalability of expensive queries is practically linear [8]. In order to deal with the huge volumes of data required by current applications, distributed architectures with P2P navigation [9] seem to offer a solution. Given enough resources, P2P networks maintain almost constant response-time while scaling to data volumes larger by several orders of magnitude.

Data Sets

Several influential research groups are currently collecting large repositories to test the scalability of their indexing tools. One of the most significant and publicly available web pages is the UCI Machine Learning Repository http://mlearn.ics.uci.edu/MLRepository.html with nearly 200 various datasets.

URL To Code

http://www-db.deis.unibo.it/research/Mtree/
: the code of the original M-tree and many references to the related literature

http://lsd.fi.muni.cz/trac/mtree/
: an improved version of the M-tree

http://gbdi.icmc.usp.br/arboretum/
: several implementations of metric indexing structures

http://lsd.fi.muni.cz/trac/messif
: the Metric Similarity Implementation Framework, MESSIF, can be used to implement centralized and distributed similarity search indexing structure prototypes.

Cross-references

- ▶ Approximate Query Processing
- ▶ Closest-Pair Query
- ▶ Data Cleaning
- ▶ Digital Libraries
- ▶ Indexing and Similarity Search
- ▶ Metric Space
- ▶ Multimedia Data Indexing
- ▶ Multimedia Data Querying
- ▶ Nearest Neighbor Query
- ▶ Peer-to-Peer System
- ▶ Spatial Indexing Techniques
- ▶ Text Indexing Techniques

Recommended Reading

1. Batko M., Novak D., Falchi F., and Zezula P. On Scalability of the Similarity Search in the World of Peers. In Proc. 1st Int. Conf. Scalable Information Systems, 2006, pp. 1–12.
2. Chávez E., Navarro G., Baeza-Yates R., and Marroquín J.L. Searching in metric spaces. ACM Comput. Surv., 33 (3):273–321, 2001.
3. Ciaccia P., Patella M., and Zezula P. M-tree: An efficient access method for similarity search in metric spaces. In Proc. 23th Int. Conf. on Very Large Data Bases, 1997, pp. 426–435.
4. Dohnal V., Gennaro C., Savino P., and Zezula P. D-Index: Distance searching index for metric data sets. Multimedia Tools Appl., 21(1):9–33, 2003.
5. Hjaltason G.R. and Samet H. Incremental similarity search in multimedia databases. Technical Report CS-TR-4199, Computer Science Department, University of Maryland, College Park., November 2000.
6. Hjaltason G.R. and Samet H. Index-driven similarity search in metric spaces. ACM Trans. Database Syst., 28(4):517–580, 2003.
7. Kelly J.L. General Topology. D. Van Nostrand, New York, 1955.
8. Samet H. Foundations of Multidimensional and Metric Data Structures. Morgan Kaufmann, San Francisco, CA, USA, 2005.
9. Traina Jr. Traina A.J.M., Seeger B., and Faloutsos C. Slim-Trees: High Performance Metric Trees Minimizing Overlap between Nodes. In Advances in Database Technology, Proc. 7th Int. Conf. on Extending Database Technology, 2000, pp. 51–65.
10. Zezula P., Amato G., Dohnal V., and Batko M. Similarity Search: The Metric Space Approach, Springer, Berlin Heidelberg, New York, 2006.

Indexing of Data Warehouses

Theodore Johnson
AT&T Labs Research, Florham Park, NJ, USA

Synonyms

Data warehouse indexing

Definition

Indices are data structures especially designed to allow rapid access to data in large databases. Data warehouses are typically used to perform intensive analyses of very large data sets. Several indices, such as projection indices, bitmap indices, bitslice indices, and summary indices, have been developed to address the special needs of data warehousing, and are presented in this entry.

Historical Background

Data warehouses were developed to capture operational information, store it over a long period, and provide support for intensive analysis of historical data. The special needs of data warehouses – very large data sets, high dimensional data, and the extensive use of categorical data – has led to the development of specialized indices intended for use in data warehouses. The use of these indices was pioneered in such systems as Model 204 and Sybase IQ.

Foundations

Indices, such as *B-trees*, *R-trees*, *quad-trees*, *hash tables*, and *inverted indices*, are integral parts of any database system. Query processing in data warehouses place special demands on the indices: very large data sets, high dimensional data, large materialized aggregate views, and categorical data. Several special techniques have been developed in response.

Projection Indices: Data warehouses are often used to store one or more high dimensional *fact tables*, which contain historical records of past events. For example, a SALES fact table might have fields which describe the customer (CustomerID, FirstName, LastName), the product sold (ProductLine, Brand, Size, Color), the method of sale (Store, Referrer, SalesPerson), and values (i.e., *measures*) which quantify the sale (Price, Discount, Quantity).

The large number of dimensions, especially those involving categorical values with a small to moderate range (e.g., Store), or with redundant information (e.g., CustomerID determines FirstName and LastName) can cause an unnormalized fact table to require an excessive amount of storage. *Star schemas* and *snowflake schemas* are common techniques to reduce fact table storage costs. Another useful technique is *vertical partitioning*. Groups of columns, or even individual columns, are stored independently but related via their record ID, or *rid*. Columns with a low cardinality and/or many null values can use special encoding schemes. Sybase IQ made prominent use of this technique.

An advantage of vertical partitioning is that many queries reference only a few of the fields of the fact table. For example, a query might be, "What is the volume of sales which involve a Discount larger than 50%, broken down by Store." If SALES is vertically partitioned, an efficient query plan is to scan the Discount and Store fields. Whenever the Discount field is 50% or larger, retrieve the linked Store field and increment a corresponding counter.

In the above example, the Discount field acts as a *projection index* [9]. Access to the field is fast because of the compact vertically partitioned storage. The set of records which satisfy the "Discount $\geq$ 50%" might be dense (more than 1 in 100), making random access indices inefficient. Furthermore, complex predicates involving several fields can be processed in the same way, providing highly efficient multi-dimensional indices.

Bitmap Indices: Bitmap indices are a logical extension of projection indices [9]. Suppose that the predicate of the query is, "Store = 'Foo' and Brand = 'Bar'". Suppose that both individual predicates is true for about 1 in 100 records – dense enough that linear scanning is effective. However their conjunct is likely to be true for about 1 in 10,000 records – sparse enough that random access is preferable. Therefore an efficient query plan is to:

- Scan the Store vertical partition for values of Foo, and record the corresponding rids
- Scan the Brand vertical partition for values of Bar, and record the corresponding rids
- Compute the intersection of the two rid lists, and fetch the corresponding records

An efficient way to store the rids and then compute their intersection is to use a *bitmap*. Each bit corresponds to a record – the bit at position 0 refers to record 0, the bit at position 1215 refers to record 1215, and so on. A bitmap can be stored as a packed array of bits, e.g., 64 bits per word. Given two bitmaps, Boolean functions can be computed by taking the corresponding Boolean function of the corresponding packed bit array, which are typically very fast CPU register operations.

$$\begin{array}{ccccc} \text{Store} = \text{'Foo} & & \text{Brand} = \text{'Bar'} & & \\ \begin{bmatrix}1\\0\\0\\1\\1\\0\\0\\0\\0\\0\\1\\1\\0\\1\end{bmatrix} & \text{AND} & \begin{bmatrix}1\\0\\1\\0\\1\\1\\1\\0\\1\\0\\0\\1\\0\\0\end{bmatrix} & = & \begin{bmatrix}1\\0\\0\\0\\1\\0\\0\\0\\0\\0\\0\\1\\0\\0\end{bmatrix} \end{array}$$

Given a set of bitmaps, complex Boolean predicates can be evaluated very efficiently. Therefore, a natural extension of projection indices are *bitmap indices*: for each value of a field, create and store a bitmap. When a predicate is to be evaluated, the bitmaps can be fetched directly, bypassing the cost of computing them. Bitmap indices are very effective for fields with low cardinality, or for representing pre-computed predicates. However, bitmap indices require a large amount of storage if the field has more than a few distinct values.

Compressed Bitmap Indices: If a field has a large number of distinct values, the bitmaps for most of the values must be sparse, and therefore compressible. A variety of techniques based on *run-length encoding* were developed in the context of image compression, but are also usable for bitmap index compression. While the RLE-based techniques can achieve near-optimal compression, the compression and decompression can be slow because compression is achieved by compressing run length codes and packing them into a bit array.

A bitmap compression technique better suited for compressing bitmap indices are the *Byte-aligned Bitmap Codes*, or BBC codes [1]. Each code word consists of one or mote bytes, eliminating the need for bit extraction. Furthermore, Boolean operations can be performed on the BBC codes directly, creating BBC code output. However, modern processors operate on multi-byte words, and data elements such as integers must be word-aligned before they can be operated upon. *Word-aligned* codes [11] extend the BBC code idea, and achieve even greater performance by avoiding word alignment costs.

The penalty for using byte-aligned or word-aligned compression codes is that they are less space efficient than the best RLE codes. A bitmap index might therefore be stored using a variety of codecs, depending on the individual bitmap properties. Furthermore, evaluating a complex predicate requires an evaluation plan, and there are a very large number of evaluation plans for a large and complex predicate. Fortunately, there is a linear-time dynamic programming optimization algorithm for choosing an optimal evaluation plan. By using BBC codes, word-aligned codes, and optimized evaluation, a bitmap index can be stored in space similar to that of a conventional index, and yet evaluate complex predicates, including large range queries, very efficiently [1,11].

Hybrid B-tree/Bitmap Indices: If a bitmap index is used for a high cardinality field, the field will have a large number of bitmap indices, and finding the correct bitmaps to use in evaluating a predicate becomes a search problem. Conversely, a B-tree index for a non-key field will in general need to store a set of records for each indexed value. A natural solution to both issues is to record the rid set of a value using a compressed bitmap. Oracle uses this technique, with BBC codes used to compress the bitmap [9].

Bitslice Indices: If the indexed field has a very high cardinality (e.g., ranges over the integers), bitmap indices lose their value. However, one can still make use of bitmap indices by using bitmaps to index value ranges. For example, one can partition the range of an integer or floating point field into, say, 1,000 ranges, then create a bitmap for each range. Range queries are likely to be efficient, but point queries might not be as selective as desired. By a judicious choice of overlapping value ranges, one can obtain more value from each bitmap. For example, a *bitslice index* [8] applied on an integer value creates a bitmap for each bit position of the value. So, for example, the value 13 has set bits in the bit-0, bit-2, and bit-3 bitmaps, reset bits in the bit-1, bit-4, bit-5, etc. bitmaps. Point queries are efficiently supported with a bitslice index; but perhaps surprisingly, one can write range queries into compact predicates on bitslice indices (see below an example of bitslice indices on an integer field).

(193)	1	1	0	0	0	0	0	0
(87)	0	1	0	1	0	0	0	0
(225)	1	1	1	0	0	0	0	1
(7)	0	0	0	0	0	1	1	1

Aggregate-Storing Indices: Data warehouses very often support aggregation queries to summarize the mass of data in the fact tables. A common type of query is to ask for the value of an aggregate within a particular range of field value(s). A hierarchical index structure, such as a B-tree, summarizes the data at every level of its hierarchy. For example, a B-tree node has a field value range, and points (directly or indirectly) to the collection of records which have a value in that range for the indexed field.

One can change the definition of "summarize" to mean storing an aggregate value. In a leaf node entry for a value v, instead of pointing to the records which have value v for the indexed field, the entry contains an aggregate value, e.g., the sum of a measure field, over all records with value v in their indexed field. An entry in an interior node (above the leaf level) contains the

sum of the aggregates in the child, as well as a pointer to the child and its indexed value range. Finding the aggregate value of the range can be done by walking the tree, once down to the minimum value of the range, and once to its maximum value.

The principle of the aggregate-summarizing B-tree can be extended to handle more useful situations, including aggregate-summarization of multiple dimensions [4,6], and summarization of intervals (useful for temporal database systems) [5].

Summary Indices: A common technique for managing a very large data warehouse is *horizontal partitioning* – partitioning records by one or more predicates. A data warehouse often receives regular updates of new data into its fact table(s), and stores a window (e.g., 1 year) of the most recent data. Fact tables may also be partitioned by other attributes, to narrow search spaces, help ensure that partitions are dense, etc.

If a fact table is sliced into a large number of vertical partitions, the partition predicates act as another type of index. This idea can be extended to recording properties of the data in a partition. For example, a summary index might record aggregates of the data, such as minimum and maximum timestamps [7]. Alternatively, a summary index can record whether or not particular field values exist in the partition [3]. For example, the index can record a list of all customer IDs that appear in a partition (or conversely in which partitions a customer identifier appears). If the average customer ID appears in only a few partition, this type of index minimized the number of partition indices to be searched.

Join and Star Indices

A *join index* is a collection of pairs $\{(r,s)\}$ such that the record in table R with record ID (RID) r joins with the record in table S with RID s, according to the join predicate which defines the index. A join index accelerates the processing of joins, a common activity in a data warehouse with a star or snowflake schema. A *star index* is a join index between a fact table and each of its dimension tables.

Key Applications

The most commonly implemented of these indices is the bitmap index, which has been implemented in most of the major commercial databases used for data warehousing. Bitmap indices are also commonly used in scientific database applications, such as High Energy Physics. In HEP applications, bitmaps indices are often used to record "interest sets" – collections of events found to satisfy some complex property – to accelerate their retrieval for later processing [10].

Cross-references

- ▶ Bitmap-based Index Structures
- ▶ Bitmap Index
- ▶ Bitslice Signature Files
- ▶ B+-Tree
- ▶ Inverted Index
- ▶ Join Index
- ▶ Measure
- ▶ Quadtrees (and Family)
- ▶ R-Tree (and family)
- ▶ Snowflake Schema
- ▶ Star Schema

Recommended Reading

1. Amer-Yahia S. and Johnson T. Optimizing queries on compressed bitmaps. In Proc. 26th Int. Conf. on Very Large Data Bases, 2000, pp. 329–338.
2. Apaydin T., Canahuate G., Ferhatosmanoglu H., and Tosun A. Approximate encoding for direct access and query processing over compressed bitmaps. In Proc. 32nd Int. Conf. on Very Large Data Bases, 2006, pp. 846–857.
3. Johnson T. Coarse indices for a tape-based data warehouse. In Proc. 14th Int. Conf. on Data Engineering, 1998, pp. 231–240.
4. Johnson T. and Shasha D. Some approaches to index design for cube forest. IEEE Data Eng. Bull., 20(1):27–35, 1997.
5. Kline N. and Snodgrass R. Computing temporal aggregates. In Proc. 11th Int. Conf. on Data Engineering, 1995, pp. 222–231.
6. Kotidis Y. and Roussopoulos N. An alternative storage organization for ROLAP aggregate views based on cubetrees. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1998, pp. 249–258.
7. Moerkotte G. Small materialized aggregates: a light weight index structure for data warehousing. In Proc. 24th Int. Conf. on Very Large Data Bases, 1998, pp. 476–487.
8. O'Neil P. and Quass D. Improved query performance with variant indices. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1997, pp. 38–49.
9. Rdb7: Performance enhancements for 32 and 64 bit systems. Available online at: http://www.oracle.com/products/servers/rdb/html/fsvlm.html.
10. Wu K., Koegler W.S., Chen J., and Shoshani A. Using bitmap index for interactive exploration of large datasets. In Proc. 15th Int. Conf. on Scientific and Statistical Database Management, 2003, pp. 65–74.
11. Wu K., Otoo E.J., and Shoshani A. Compressing bitmap indexes for faster search operations. In Proc. 14th Int. Conf. on Scientific and Statistical Database Management, 2002, pp. 99–108.

I

Indexing of the Current and Near-Future Positions of Moving Objects

Simonas Šaltenis, Christian S. Jensen
Aalborg University, Aalborg, Denmark

Definition

A scenario is assumed where a large population of objects capable of reporting positional data to a central server exists. Specifically, an object may report its current position, but may also report its current velocity vector.

A key challenge is to be able to accommodate the very frequent updates inherent to this scenario. Another important challenge is to contend with, and indeed exploit, all the available positional data so that better query results to near-future queries are enabled.

To address these challenges, the position of an object is typically modeled as a linear function from time to space (typically the two- or three-dimensional Euclidean spaces are assumed). Such functions are readily available, and the positions they return get outdated less frequently than do constant functions, thus reducing the update rate. Further, they enable the computation of more accurate results for near-future queries.

The fundamental types of queries to be supported by an index include ones that retrieve the objects with a position within a specified, rectangular spatial region: *timeslice* and *window* queries consider a stationary region for a single time point or a time interval; and so-called *moving queries* attach a rectangle to a moving object and consider a time interval. The times supported range from the current time to some near-future time.

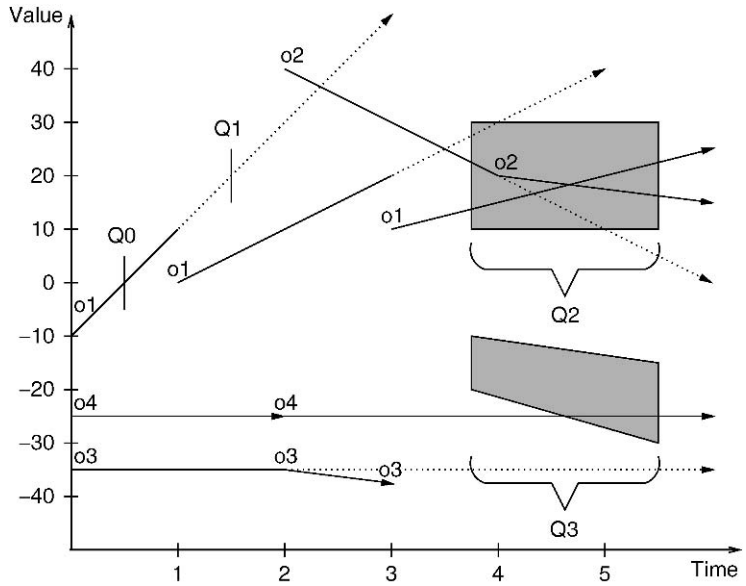
Figure 1 shows a set of trajectories in (x, t)-space of objects and illustrates the three types of queries: $Q0$ and $Q1$ are timeslice queries, $Q2$ is a window query, and $Q3$ is a moving query.

Other types of queries such as nearest neighbor queries or reverse nearest neighbor queries may also be supported. In addition, an index should also support updates, i.e., insertions and deletions.

For its illustration of key concepts, this entry uses primarily R-tree-based techniques; specifically, the TPR-tree [12] is covered in some detail, as it is the ancestor of a sizable family of indices.

Historical Background

The first proposal for the indexing of moving object positions as linear functions is due to Tayeb et al. [14] who propose to use PMR-quadtrees for indexing the future linear trajectories of one-dimensional moving point objects as line segments in (x, t)-space. Kollios et al. [6] also focus mostly on one-dimensional data



Indexing of the Current and Near-Future Positions of Moving Objects. Figure 1. Query examples for one-dimensional data [12].

and employ the so-called *duality* data transformation where a line $x = x(t_{ref}) + v(t - t_{ref})$ is transformed to the point $(x(t_{ref}),v)$, enabling the use of regular spatial indices.

Later research focuses on two-dimensional and, in some cases, three-dimensional data. In general, the proposed indexing methods can be classified according to the space that they index, i.e., what view is taken on the indexed data. Assume the objects move in d-dimensional space (d = 1,2,3). The first approach is to index future trajectories as lines in $(d + 1)$-dimensional space. This is the approach taken by Tayeb et al. [14], but the approach is difficult to extend to higher dimensions, and the index has to be rebuilt periodically.

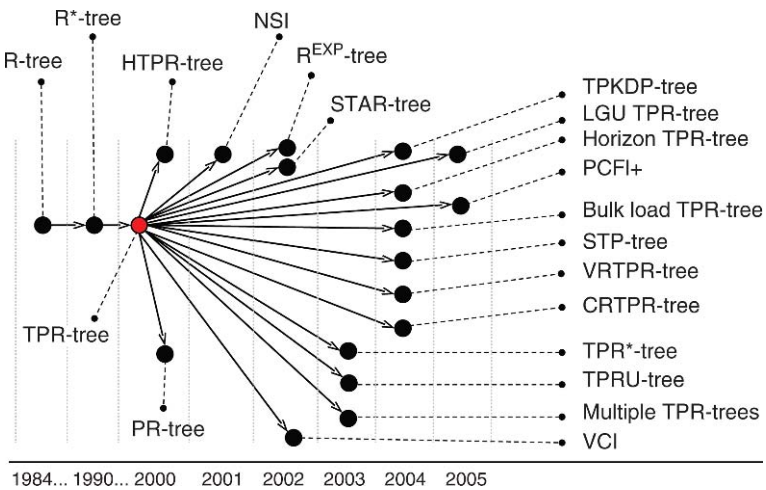
The second approach is to transform the trajectories to points in a higher-dimensional space which are then indexed. Queries are then also transformed to counter the data transformation. The transformation of Kollios et al. [6] maps linear functions in d-dimensional space into static points in $2d$-dimensional space.

In STRIPES [8], PR quadtrees are used to index these $2d$-dimensional points. Yiu et al. [15] instead propose to use space filling curves to further transform $2d$-dimensional points into one-dimensional points that are then indexed by the B^+-tree. Agarwal et al. [1] combine the duality transformation with kinetic data structures [2]. The main idea of kinetic data structures is to schedule future events that update a data structure so that necessary invariants hold.

The third approach, sometimes referred to as indexing in the *primal* space, is to index the data in their native, d-dimensional space, which is possible by parameterizing the index structure using velocity vectors and thus enabling the index to be "viewed" as of any future time. This absence of transformations yields quite intuitive indexing techniques.

The Time-Parameterized R-tree (TPR-tree) [12] exemplifies this approach. Proposed by Šaltenis et al. in 2000, the TPR-tree gave rise to a number of other access methods, as illustrated in Fig. 2. For example, the R^{EXP}-tree [11] extends the TPR-tree to index data with expiration times, so that the positions of the objects that do not update their positions are automatically removed from the index. The TPR^*-tree [13] introduces new heuristics with the objective of improving the query performance of the TPR-tree and to optimize it for workloads of queries that differ slightly from those targeted by the TPR-tree. The STAR-tree [10] modifies the TPR-tree by introducing more complex time-parameterized bounding rectangles and making the index self-adjustable. The Velocity Constrained Indexing (VCI) [9] technique uses the regular R-tree with an additional field of v_{max} in each node. The v_{max} is used to expand the bounding rectangles of the R-tree when future queries are processed.

The fourth approach is to index the objects' positions as of some specific time points, so-called label timestamps, and to extend the spatial extents of queries according to the maximum speeds of objects



Indexing of the Current and Near-Future Positions of Moving Objects. Figure 2. TPR-tree origins and follow-up research.

and the difference between the time specified in the query and the label timestamps. The B^x-tree [4,5] uses a combination of space-filling curves and B^+-trees to index the (static) positions of objects as of label timestamps.

Foundations

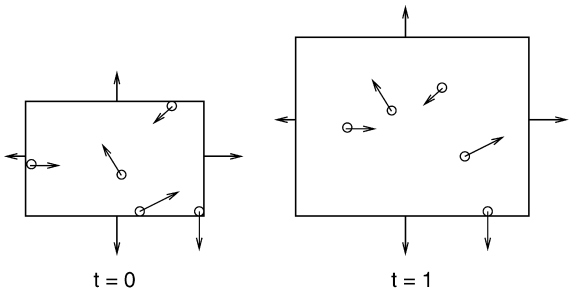
Data and Queries

For an object moving in d-dimensional space, the object's position at some time t is given by $\bar{x}(t) = (x_1(t), x_2(t),..., x_d(t))$, where it is assumed that the times t are not before the current time. This position is modeled as a linear function of time, which is specified by two parameters. The first is a position for the object at some specified time t_{ref}, $\bar{x}(t_{ref})$, which is called the reference position. The second parameter is a velocity vector for the object, $\bar{v} = (v_1, v_2,..., v_d)$. Thus, $\bar{x}(t) = \bar{x}(t_{ref}) + \bar{v}(t - t_{ref})$.

Then, as shown in Fig. 1, a *timeslice* query, $Q = (R, t)$, retrieves points that will be inside the d-dimensional hyper-rectangle R at time t. A *window* query, $Q = (R, t^{\vdash}, t^{\dashv})$, retrieves points that will be inside the hyper-rectangle R sometime during time-interval $[t^{\vdash}, t^{\dashv}]$. A *moving* query, $Q = (R_1, R_2, t^{\vdash}, t^{\dashv})$, retrieves points with trajectories in $(\bar{x}, t)$-space crossing the $(d + 1)$-dimensional trapezoid obtained by connecting R_1 at time $t^{\vdash}$ to R_2 at time $t^{\dashv}$.

The Structure of the TPR-Tree

A large number of indices utilize the structure of the TPR-tree, which indexes moving points in one, two, or three dimensions. It employs the basic structure of the R-tree, which stores data in the leaves of a balanced index tree, and each non-leaf index entry contains a minimum bounding rectangle (MBR) of all the data in the subtree pointed to by the entry. In contrast to the R-tree, the indexed points as well as the bounding rectangles are augmented with velocity vectors. This way, bounding rectangles are time parameterized – they can be computed for different time points. Velocities are associated with the edges of bounding rectangles so that the enclosed moving objects (points or other rectangles) remain inside the bounding rectangles at all times in the future. More specifically, if a number of points p_i are bounded at time t, the spatial and velocity extents of a bounding rectangle along the x axis are computed as follows:



Indexing of the Current and Near-Future Positions of Moving Objects. Figure 3. Example time-parameterized bounding rectangle [3].

$$x^{\vdash}(t) = \min_i\{p_i.x(t)\}; \quad x^{\dashv}(t) = \max_i\{p_i.x(t)\};$$
$$v_x^{\vdash} = \min_i\{p_i.v_x\}; \quad v_x^{\dashv} = \max_i\{p_i.v_x\}.$$

Figure 3 shows an example of the evolution of a bounding rectangle in the TPR-tree computed at $t = 0$. Note that, in contrast to R-trees, bounding rectangles in the TPR-tree are not minimum at all times. In most cases, they are minimum only at the time when they are computed. A process called *tightening*, performed whenever an index node is modified during an insertion or a deletion, recomputes a node's time-parameterized bounding rectangle, rendering it minimum at that time.

Querying the TPR-Tree

The TPR-tree can be interpreted as an R-tree for any specific time, t_q. This suggests that algorithms that are based on the R-tree are easily "portable" to the TPR-tree. For example, answering a timeslice query proceeds as for the R-tree, the only difference being that all bounding rectangles are computed for the time t_q specified in the query before intersection is checked.

To answer window queries and moving queries, the algorithm has to check if, in $(\bar{x}, t)$-space, the trapezoid of a query intersects with the trapezoid formed by the part of the trajectory of a bounding rectangle that is between the start and end times of the query. This can be checked using a simple algorithm [12]. Figure 4 illustrates the intersection between a one-dimensional time-parameterized bounding rectangle (interval) and a moving query.

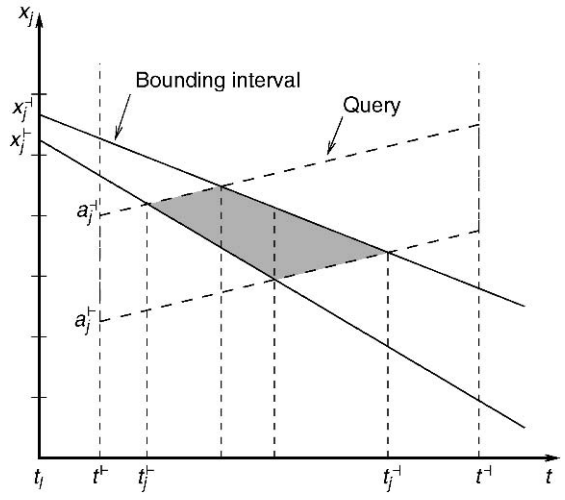
Updating the TPR-Tree

An update of the moving object's position is modeled as a deletion of the old position followed by an

insertion of the new position. The TPR-tree's update algorithms are based on the update algorithms of the R*-tree, which is an R-tree with improved update algorithms. The update algorithms of the TPR-tree differ from the corresponding algorithms of the R*-tree only in the heuristics that are used. The R*-tree uses heuristics that minimize certain functions, including the area of a bounding rectangle, the intersection of two bounding rectangles, the margin of a bounding rectangle, and the distance between the centers of two bounding rectangles. As the TPR-tree employs time-parameterized bounding rectangles, the above-mentioned functions are time dependent, and their evolution in time should be considered. Specifically, given an objective function $A(t)$, the following integral should be minimized:

$$\int_{t_c}^{t_c+H} A(t)dt,$$

where t_c is the time when the heuristics is being applied and H is a so-called *time horizon* parameter, the value of which reflects how far into the future queries are expected to "see" the effects of the application of this heuristics. If $A(t)$ is area (in d-dimensional space), the integral computes the area of the trapezoid that represents part of the trajectory of a bounding rectangle in $(\bar{x}, t)$-space (a d+1-dimensional volume, see also Fig. 4).



Indexing of the Current and Near-Future Positions of Moving Objects. Figure 4. Intersection of a bounding interval and a query [12].

Duality-Transformation Approach

The general approach of indexing moving points in their native space using a time-parameterized index structure such as the TPR-tree is closely related to the duality transformation approach [6, 8, 15].

Considering one-dimensional data, the duality transformation transforms the linear trajectory of a moving point $x = x(t_{ref}) + v(t - t_{ref})$ in (x, t)-space into a point $(x(t_{ref}), v)$, where t_{ref} is a chosen reference time. Queries, then, are also transformed.

Bounding points $(x(t_{ref}), v)$ in the dual space with a minimum bounding rectangle is equivalent to bounding them (as moving points) with a time-parameterized bounding interval computed at t_{ref}. Figure 5 shows the same bounding rectangle and query in $(x(t_{ref}), v)$-space and in (x, t)-space.

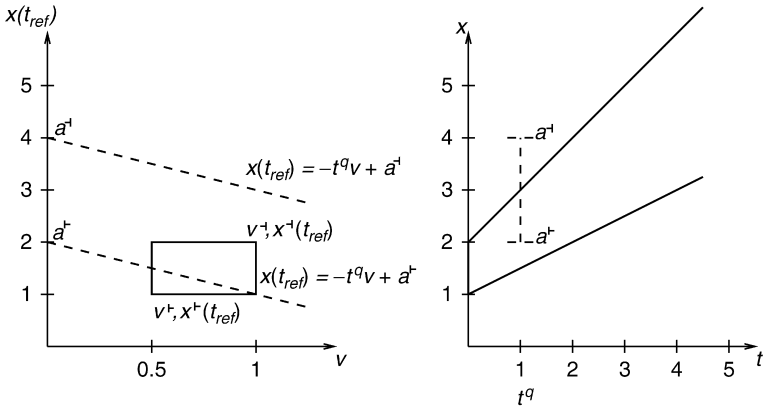
In spite of the equivalence among the bounding rectangles used in the two approaches, the algorithms used in different indexes may vary substantially. A duality-transformation index may not even explicitly use minimum bounding rectangles [15]. Furthermore, while the heuristics of time-parameterized indexes consider the objects' positions at the time the heuristics are applied, the algorithms of duality-transformation approaches always use a pre-chosen constant t_{ref}. For this reason, duality-transformation approaches usually use two (or more) indexes, such that updates are placed into the latest index; and when the earliest index becomes empty, a new index is created [6].

A sufficiently high update rate is crucial in order for both the time-parameterized indexes, such as the TPR-tree, and the indexes using the duality transformation to offer good query performance. The reasons for this are most obvious in the TPR-tree, where query performance degrades due to the uninterrupted expansion of time-parameterized bounding rectangles, which results in more queries intersecting with a given bounding rectangle.

Key Applications

Online, Position-Aware People, Vehicles, and Other Objects

The rapid and continued advances in positioning systems, e.g., GPS, wireless communication technologies, and electronics in general render it increasingly feasible to track and record the changing positions of objects capable of continuous movement. Indexing of such positions is necessary in some Location-Based Services



Indexing of the Current and Near-Future Positions of Moving Objects. Figure 5. Timeslice query (*dashed*) and bounding interval (*solid*) in dual ($x(t_{ref})$, v)-space and (x, t)-space.

(LBS), such as location-based games, tourist-related services, safety-related services, and transport-related services (e.g., fleet tracking).

Process Monitoring

Applications such as process monitoring do not depend on positioning technologies. In these, the position of a "moving point" could for example be a pair of temperature and pressure values at a specific sensor. A timeslice query then would retrieve all sensors with current measurements of temperature and pressure in given ranges.

Future Directions

Tracking continuous real-world phenomena inevitably involves high rates of updates that have to be processed by the index. Very recent research provides a number of interesting ideas for speeding up the processing of index updates. Further research is needed to fully explore trade-offs among update performance, query performance, and query accuracy. Finally, main-memory indexing of such data could be explored to dramatically boost the performance of index updates.

How to handle the always-present uncertainty about the positions of objects has not been sufficiently explored in connection with indexing and warrants further study.

URL to Code

The source code of the TPR-tree can be found at: http://www.cs.aau.dk/~simas/

The source code of the TPR*-tree can be found at: http://www.rtreeportal.org/

Cross-references

- ▶ Indexing Historical Spatio-Temporal Data
- ▶ R-Tree (and family)
- ▶ Spatial Indexing Techniques
- ▶ Spatio-Temporal Trajectories

Recommended Reading

1. Agarwal P.K., Arge L., and Erickson J. Indexing Moving Points. In Proc. 19th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems, 2000, pp. 175–186.
2. Basch J., Guibas L.J., and Hershberger J. Data Structures for Mobile Data. In Proc. 8th Annual ACM -SIAM Symp. on Discrete Algorithms, 1997, pp. 747–756.
3. Benetis R., Jensen C.S., Karčiauskas G., and Šaltenis S. Nearest and Reverse Nearest Neighbor Queries for Moving Objects. VLDB J., 15(3):229–249, 2006.
4. Jensen C.S., Lin D., and Ooi B.C. Query and Update Efficient B^+-Tree Based Indexing of Moving Objects. In Proc. 30th Int. Conf. on Very Large Data Bases, 2004, pp. 768–779.
5. Jensen C.S., Tiešytė D., and Tradišauskas N. Robust B^+-Tree-Based Indexing of Moving Objects. In Proc. 7th Int. Conf. on Mobile Data Management, 2006, p. 12.
6. Kollios G., Gunopulos D., and Tsotras V.J. On Indexing Mobile Objects. In Proc. 18th ACM SIGACT-SIGMOD-SIGART Symp, Principles of Database Systems, 1999, pp. 261–272.
7. Mokbel M. F., Ghanem T.M., and Aref W.G. Spatio-Temporal Access Methods. IEEE Data Eng. Bull., 26(2):40–49, 2003.
8. Patel J.M., Chen Y., and Chakka V.P. STRIPES: An Efficient Index for Predicted Trajectories. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 2004, pp. 637–646.
9. Prabhakar S., Xia Y., Kalashnikov D.V., Aref W.G., and Hambrusch S.E. Query Indexing and Velocity Constrained Indexing: Scalable Techniques for Continuous Queries on Moving Objects. IEEE Trans. Computers, 51(10):1124–1140, 2002.
10. Procopiuc C.M., Agarwal P.K., and Har-Peled S. STAR-Tree: An Efficient Self-Adjusting Index for Moving Objects. In Proc. of ALENEX Workshop, 2002, pp. 178–193.

11. Šaltenis S. and Jensen C.S. Indexing of Moving Objects for Location-Based Services. In Proc. 18th Int. Conf. on Data Engineering, 2002, pp. 463–472.
12. Šaltenis S., Jensen C.S., Leutenegger S.T., and Lopez M.A. Indexing the Positions of Continuously Moving Objects. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 2000, pp. 331–342.
13. Tao Y., Papadias D., and Sun J. The TPR*-Tree: An Optimized Spatio-Temporal Access Method for Predictive Queries. In Proc. 29th Int. Conf. on Very Large Data Bases, 2003, pp. 790–801.
14. Tayeb J., Ulusoy Ö., and Wolfson O. A quadtree based dynamic attribute indexing method. Computer J., 41(3):185–200, 1998.
15. Yiu M.L., Tao Y., and Mamoulis N. The B^{dual}-Tree: indexing moving objects by space filling curves in the dual space. VLDB J., 17(3):379–400, 200.

Indexing the Past

► Indexing Historical Spatio-Temporal Data

Indexing the Web

Edleno Silva de Moura[1], Marco Antonio Cristo[2]
[1]Federal University of Amazonas, Manaus, Brazil
[2]FUCAPI, Manaus, Brazil

Synonyms

Web indexing

Definition

The process of collecting, parsing, and storing data to provide fast and accurate retrieval of content available on the web. The result of this process is a structure called index that maps the collected data (for instance, words, phrases, concepts, or sound fragments) to the web location where it is possible to find content associated with the data (for instance, pages containing these words, phrases, concepts, or music with the sound fragments). Depending on the data collected, several indices may be created. The process can be manual or automatic. Manually generated indices include web directories, back-of-book-style indices, and metadata. Automatically generated indices are normally associated with the infra-structure of search engines.

Historical Background

One of the first efforts to index the web content was developed by a MIT student, Matthew Grey, who created a program to estimate the size of the Web. This program, called Word Wide Web Wanderer, was able to recursively traverse the web's hypertext structure, retrieving all the referenced documents [11,12,15]. It was the first crawler (also known as robot, bot, or spider). The URLs collected by this program formed the first database of web sites, the Wandex. It also caused quite a controversy due to the large amount of bandwidth it used. In response to this in October 1993, Martijn Koster created a program called ALIWEB (Archie-like Indexing of the Web). To avoid using crawlers, Koster asked webmasters to post their own index information for each page they wanted listed. ALIWEB, however, never reached a size large enough to compete with crawler-based indexers.

After its controversy start, crawlers became a standard component in most search engine indexers. By the end of 1993, search services such as JumpStation, World Wide Web Worm, the Repository-based Software Engineering (RBSE), and Excite used crawlers to gather information of Web pages. In general, these services indexed words extracted from URL, title, and text fragments of the pages.

The indices used by search engines were not designed towards human navigation. Thus, unlike book indices or yellow pages, they did not allow users to freely explore web contents by narrowing the field of interest. This observation motivated the University of Texas to create Elnet Galaxy in January 1994, the oldest web directory. In this index, contents were manually reviewed and organized into hierarchical categories. As in ALIWEB, only pages submitted to the service were listed. In April 1994, two students from Stanford University, David Filo and Jerry Yang, expanded a personal list of pages into a searchable directory, Yahoo!. Unlike Galaxy, Yahoo! automated aspects of the gathering and categorization process, blurring the distinction between search engine and directory.

Still in 1994, a student of the University of Washington, Brian Pinkerton, started another crawler-based search engine, the WebCrawler. It was the first one to index the full text of the pages. Soon after, in July, a project from Carnegie-Mellon University, headed by Michael Mauldin, Lycos, provided indices supporting prefix matching, word proximity, and a multimillion page catalog. In 1995, Altavista started indexing text

associated with images, music, and videos. In the following years, several other services were stated, such as Inktomi (1996), Google (1997), and MSN Search (1998), improving image and video indexing, addressing spam issues related to indexing, targeting other document and media formats as well as other web elements. For instance, the increasing adoption of link analysis in ranking algorithms motivated the maintenance of citation indices. Also in 1998, the Open Directory was started to create a human-reviewed directory constructed and maintained by a community of volunteers.

Foundations

Introduction

The web is a space for mass publication at an unprecedented scale. The earliest strategies to make all the information in this space discoverable by the users consisted of applying traditional information retrieval techniques to the new content [8,10]. As a consequence, the first search engines adopted approaches based on manually or automatically built indices.

When performing manual indexing, experts describe the web content by selecting terms which indicate what the content is about. These terms are normally selected from controlled vocabularies and are organized as flat or hierarchical lists. It is also common that the authors of the contents select keywords and metadata to describe them. The resulting indices may be oriented towards direct navigation by humans or to be retrieved by other programs.

An example of index oriented towards human navigation is the web directory. In such an index, links to sites are organized into hierarchical categories, according to the sites' contents. In web directories, normally the tasks of collecting and categorizing pages are carried out under supervision of human editors.

An example of index not oriented to humans is a hidden list of metadata. Metadata are data about data. As a mean of assisting a search engine to locate content or an information entity, they can be used to describe that content or entity. While not visible to humans, this information can provide contextual clues to automatic algorithms used by search engines.

Manual indexing presents two problems. First, it does not scale with the size of the web. This is an even more daunting problem considering the dynamic nature of web, where many documents have no persistent state and would require frequent revisions. Second, to be effective as a search tool, experts and users have to agree with respect to the taxonomy employed for classification, which is hard when using large taxonomies [8]. These problems contributed for the dominance of strategies based on automatic indexing.

In automatic indexing, a program (i.e., a web crawler) scans the web, collecting pages to extract data useful to describe them. The contents of the pages (text, media, links, etc.) are parsed, and decomposed into their components (tokens). Each token constitutes an atomic search criterion. For instance, in the case of textual content, the tokens are normally words and the most simple search query is composed of at least one word. They are normally converted to a standard form through some transformations and inserted into one or more data structures. These structures are the base for one or more indices.

The resulting indices are used by search engines to allow several search strategies, such as full-text search, and support different ranking algorithms, central components of systems with automatic indexing. Such algorithms increase the likelihood that the system will automatically order results according to their relevance to the users. Typically, the indexing is performed at a predetermined time interval due to the required time and processing costs.

While manually created indexes are commonly associated with natural language text, automatic indexers are likely to support additional kinds of media, such as images, sounds, and video. It is also common that indexers expand the textual content of a web page p with additional evidence such as the anchor text found in pages that link to p. The anchor text is the text highlighted by web browsers to indicate the presence of a link. Normally, it provides a useful description of the target page. Thus, pages with many incoming links can accumulate several descriptions emphasizing what the page is about. In fact, the use of different evidence is a common way to assist ranking algorithms to order results.

Data Structures to Support Automatic Indexing A typical data structure used to map tokens to their associated contents is the inverted file or inverted index [2,10]. The inverted file stores a list of occurrences of each token, normally in the form of a hash table or binary tree. In larger indices it normally assumes a

form of a distributed hash table. The list associated with each token is called an inverted list.

By using an inverted file, the search engine can find content (for instance, a document) through direct access. In its simplest design, this structure can only determine if a token exists in a document, since the inverted lists store only occurrence information. Further, they can only determine which documents match a query. They do not rank them.

In real indexers, however, the inverted list stores additional information to support searching and ranking. Further, the list can be sorted in such way that some documents are more likely to be retrieved than others, which enables several ranking speedup optimizations.

Additional information typically stored in inverted lists include the frequency and position of a token. In the case of textual content, the frequency information can be used to assist ranking algorithms to judge the relevance of a document to a search query whereas the position information enables search algorithms to support word proximity and phrase search.

To ensure that more promising documents will be retrieved earlier, the inverted list is normally sorted according to a query-independent document score. This score is calculated by combining query-independent factors associated with the document, such as the frequency that users click on it, its link popularity, URL size, and spam score.

Other data structures are used to support automatic web indexing. They are generally adopted to speedup retrieval or provide additional information for ranking algorithms. For instance, by using only the previously described inverted file, a search engine can correctly answer phrase queries. For this, it has to retrieve the inverted lists for all words in the phrase and intersect them. In practice, this is a very slow process and other strategies are employed. A simple alternative strategy is to extend the inverted file to support phrases along with words, as tokens. In such a case, only common phrases are included in the data structure. Another strategy consists in using a structure that facilitates the processing of pairs of words. For instance, the inverted list associated with a word *w* can be divided into sublists, according to the words that follow *w*. For instance, for retrieving the list associated with the phrase "web indexing", one can retrieve the inverted list of "web" and then the inverted sublist of "indexing."

Additional data structures used to support automatic web indexing are citation indices, n-gram indices, term document matrices, and tries, to cite a few. A citation index stores hyperlinks between documents to assist citation analysis, a set of bibliometric techniques used by ranking algorithms. N-gram indices store sequences of length of data to support some types of retrieval, such as phrase retrieval. Term document matrices are sparse matrices that store the occurrence of the words in the documents, needed for a technique called latent semantic analysis [4]. Tries are ordered tree data structures used to store an associative array where the keys are usually strings. A particular kind of trie is a suffix tree, used to store the suffixes of data strings in order to carry out fast full text searches.

I

Challenges and Design Factors in Automatic Indexing

By using automatic methods, search engines are able to index a significant part of the web while sustaining very low response times, when compared to manual indexing. Given the huge size and diversity of the web, many challenges were faced to reach such a performance.

From an engineering point of view, it was necessary to deal with several design factors, such as the amount of computer storage necessary to support the index, how the index should be filtered, compressed, and distributed [14]. Other key design factors are the update policy, reliability, and required electric power.

Regarding distribution, for instance, indices are normally partitioned by documents, that is, each node contains the index for a subset of all documents [8]. To answer a query, such query has to be submitted to each subset of documents and the results merged before being shown to the user.

A strategy to reduce the use of disk, network, and memory resources is to compress the key data structures. These compressed structures enable fewer disk and network accesses leading to faster processing, in spite of the processing costs associated with compression and decompression algorithms.

From a computer science point of view, it was necessary to find properly data structures to support the desired search and ranking operations in the required time and space. Examples of aspects considered included the processing of strings and n-grams, the possibility of approximated searching, document position, and the type of media to be indexed.

It was also necessary to deal with several algorithmic challenges. For instance, the adoption of distributed architectures required parallel algorithms for building the indices and synchronize the crawlers. Space, time, and accuracy requirements motivated researches on data selection with minimum impact on ranking, compression, subject categorization, and clustering. Robustness and completeness requirements motivated research on format analysis in order to design methods able to handle malformed documents, documents whose content is dynamically generated, proprietary formats, and multiple character sets.

Given the importance of ranking results for search engines, indices include much evidence which require the development of new ranking algorithms to take advantage of them. For instance, by storing hyperlinks between pages it is possible to infer the importance of the pages. A page pointed to by many other pages, specially by important ones, is probably an important page. This is the central idea of a very popular ranking strategy called PageRank [3]. This idea can be refined by characterizing the importance of a page according to its role, in particular, as a hub or an authority. The hubness of a page is the quality associated with its capability to point to other interesting pages in the web, whereas its authority is related to how good is considered its own informative content. HITS is a very well known ranking algorithm that infers page importance by taking into consideration these roles [6]. In particular, HITS explores a recursive definition of hub and authority pages, that is, a good hub is the page that points to many good authorities and a good authority is the page that is pointed to by many good hubs.

From a linguistic point of view, it was necessary to deal with the complexity inherent to the processing of natural language [13]. This is particularly difficult due to the necessity to handle multiple languages [8]. In a multilingual indexer, even apparently simple tasks as recognizing the word boundaries represent a challenge, since words are not clearly separated by white spaces in some languages such as Chinese, Japanese, and Arabic. Further, to deal with issues such as ambiguity, compression, and properly match of sentences, many search engines use additional information such as the lexical category of the words and their roots. All of this information requires language-dependent techniques and, by extent, automated language recognition, a subject of ongoing research in natural language processing.

Key Applications

Web indexing is essential for making the wealth of information in the web discoverable by the users.

Future Directions

The web continues growing uncoordinatedly, at large scale and with great diversity of interests [8]. These aspects of the web raise several problems that affect indexing, motivating many studies to address them. For instance, since the number of pages in the web grows beyond the capabilities of search engines collecting them, research has been and will likely continue to be focused on improving the ability of recognizing as early as possible not so useful pages in order to avoid indexing them [1].

On the other hand, most of the information in the web is stored in databases and is only available through systems that generate pages dynamically in response to user interactions. These dynamic pages comprise which is called *the hidden web* (also referred to as *deep web*). As a consequence, much digital content is inaccessible to typical crawlers. Research is under way to make it possible to index the hidden web, for instance, by exploring search query syntax, standard formats of online resources, and application programming interfaces [9].

There is an increasingly interest on indexing rich media formats. Current indexing approaches for such media take advantage of text clues such as image legends, video subtitles or music lyrics. New research has focused on describing them by means of their content [7] in order to make possible, for instance, the retrieval of an image by its description or of music by a sound fragment. The main challenge regarding media indexing is how to represent it such that it can be efficiently stored, retrieved, and compared. For instance, a music retrieval system should be able to map pitches to notes by analyzing waveforms, to store these notes into an index structure, and to match note strings allowing a certain amount of noise [9].

Another research focus is fighting spam [5]. The diversity of interests in the Web along with a growing audience led to the adoption of several strategies to ensure top positions on search engine results lists. Many of these strategies consisted of manipulating the content of the pages to deceive the indexer by using misleading terms or abusing of metatags. In response, movements to standardize metatag content has emerged. Also, research has lead to the design of

robust crawling algorithms able to deal with several spam traps and new methods to infer content quality and reliability.

Cross-references

- ▶ Information Retrieval
- ▶ Inverted Files
- ▶ Suffix Tree
- ▶ Text indexing Techniques
- ▶ Trie

Recommended Reading

1. Baeza-Yates R., Castillo C., Marin M., and Rodriguez A. Crawling a country: better strategies than breadth-first for web page ordering. In Proc. 14th Int. World Wide Web Conference, 2005. pp. 864–872.
2. Baeza-Yates R.A. and Ribeiro-Neto B. Modern information retrieval. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.
3. 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.
4. Deerwester S., Dumais S.T., Landauer T.K., Furnas G.W., and Harshman R.A. Indexing by latent semantic analysis. J. Soc. Inf. Sci., 41(6):391–407, 1990.
5. Heymann P., Koutrika G., and Garcia-Molina H. Fighting spam on social web sites: A survey of approaches and future challenges. IEEE Internet Comput., 11(6):36–45, 2007.
6. Kleinberg J.M. Authoritative sources in a hyperlinked environment. J. ACM, 46(5):604–632, 1999.
7. Liu Y., Zhang D., Lu G., and Ma W.Y. A survey of content-based image retrieval with high-level semantics. Pattern Recognit., 40 (1):262–282, 2007.
8. Manning C.D., Raghavan P., and Schütze H. Introduction to Information Retrieval, Ch. 18, 19, 20 (optional). Cambridge University Press, Cambridge, 2008.
9. Mostafa J. Seeking better web searches. Sci. Am. Mag., February 2005.
10. Salton G. Automatic Text Processing: the Transformation, Analysis, and Retrieval of Information by Computer. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.
11. Sonnenreich W. A History of Search Engines, 1997. Available at http://www.wiley.com/legacy/compbooks/sonnenreich/history.html.
12. Underwood L. A Brief History of Search Engines. Available at http://www.webreference.com/authoring/search_history.
13. Voorhees E.M. Natural language processing and information retrieval. In Information Extraction: Towards Scalable, Adaptable Systems, M.T. Pazienza (ed.), 1999, pp. 32–48.
14. Witten I.H., Moffat A., and Bell T.C. Managing Gigabytes: Compressing and Indexing Documents and Images, 2nd edn. Morgan Kaufmann, Los Altos, CA, 1999.
15. Zakon R.H. Hobbes' Internet Timeline. Available at http://zakon.org/robert/internet/timeline/.

Indexing Units

Jaap Kamps
University of Amsterdam, Amsterdam, The Netherlands

Synonyms

Indexing granularity

Definition

Indexing units refers to the granularity of information in the retrieval system's index, which can be in principle any document part of a structured text, and as a consequence determines the possible units of retrieval. There are three basic approaches. The first approach is to index every potentially retrievable unit as a whole – the so-called element-based approach [13]. The second approach is to index disjoint nodes – and relying on aggregation or score propagation methods for scoring higher-level nodes [e.g., 1,12]. The third approach is to index only selected elements, for example by indexing particular element types in separate indexes [10]. Various mixtures of these approaches have also been applied.

All approaches make implicit or explicit assumptions on the (most likely) *unit of retrieval*. Although there may be no designated retrieval unit (such as the document or root node of the structured document), this also does not mean that every document part (such as a sub-tree of the structured document) is an equally desirable retrieval unit. Such assumptions may be relatively generic (such as paragraphs and sections being more informative than very short excerpts in bold or italics) or may depend on the query at hand (such as a structured query requesting elements with a particular tag). In all cases these assumptions depend on the sort of structured documents (which may range from strict XML databases to loosely structured textual documents with mark-up), and on the sort of information need (which may range from a strict database query with well defined semantics to a vague information retrieval topic of request). Structured text retrieval typically deals with loosely structured textual documents and vague information retrieval queries.

Historical Background

Structured text retrieval has a long pre-history in text retrieval. The first test collections consisted of short document surrogates such as bibliographic

descriptions (in various forms) or abstracts. Since the 1990s, test collections consisted predominantly of full text newspaper and newswire data. Interestingly, the bibliographic descriptions were highly structured catalogue records, and the newspaper data were typically structured in SGML, yet no particular use was made of the internal document structure. In fact, the use of the internal structure was usually explicitly outlawed, the main motivation for this being the desire to develop retrieval techniques that would work on all (flat) texts.

The use of document structure derived from SGML mark-up was pioneered by Wilkinson [15], studying ad hoc SGML element retrieval, and by Myaeng et al. [11], exploring structured queries that contain references to the SGML document structure. Similar early work on HTML is in [4]. There are also many similarities with the early work on multimedia retrieval, such as the DOLORES system [5] and the FERMI model [3], which are addressing the problem of retrieving information from structured documents. Ad hoc XML element retrieval and best entry point retrieval was studied in the Focus project [9]. In recent years, the main thrust of research in structured text retrieval is the annual initiative for the evaluation of XML retrieval [7].

Foundations

Structured text retrieval typically deals with loosely structured textual documents and vague information retrieval queries, and the discussion is focused exclusively on this case. Indexing structured texts presents a number of challenges since such documents can be decomposed according to their internal structure. XML documents have a hierarchical structure of nested elements (or subtrees). That is, for example, an entire article consisting of front matter, body, and back matter. The body, in turn, consists of sections. The sections, again, consist of subsections. The subsections of paragraphs, and so on. Since there is no fixed unit of retrieval it is an open question what should be put in the index.

A prototypical structured text retrieval task is XML element retrieval. In text retrieval, evaluation is based on a "frozen" set of search requests (or "topics") with a set of known relevant results – XML elements regarded as relevant by the topic author. In XML retrieval, topics consist of a short keyword (or content-only) title; a structured (or content-and-structure) query in NEXI [14]; a single sentence description; and a long narrative statement of the search request. The retrieval system takes as *input* the standard keyword query, or the structured NEXI query. To give a concrete example, the ⟨title⟩ field of the INEX 2004 topic number 104 is

> ► *Toy Story*

The requested *output* is a ranked list of document components (in this case XML elements in the INEX IEEE collection containing full-text articles). There is no fixed *unit of retrieval*: if a whole article is relevant, return the ⟨article⟩ element, but if only a section is relevant, return the ⟨sec⟩ element. Retrieval systems will, of course, rank the XML elements based on the occurrences of query terms (or possibly phrases, stems, or synonyms based on these terms). However, whether a result is indeed relevant for the query is determined by a human judgment on whether the information in the element satisfies the topic author's information need. For the above mentioned INEX 2004 topic 104, the ⟨narrative⟩ field reads

> ► *To be relevant, a document/component must discuss some detail of the techniques or computer infrastructure used in the creation of the first entirely computer-animated feature-length movie, "Toy Story."*

A human judge (usually the topic author) will assess the relevance of the retrieved results, where relevant means that the element is both *exhaustive* (it provides useful information on the topic of request) and *specific* (there is a minimal amount of off-topic material) (There have been different measures developed over the years).

Although, in principle, any document part or XML element can be retrieved, some document parts tend to be more likely to be relevant. Table 1 shows the distribution of relevant elements over tag-names (Here relevant is according to the strict quantization function). In this case, most frequently paragraphs (⟨p⟩), sections (⟨sec⟩), and subsections (⟨ss1⟩, ⟨ss2⟩) are judged relevant, but also entire articles (⟨article⟩). The precise tags are a direct result of the particular mark-up structure of the IEEE collection, which is mostly based on the logical structure of the articles. Generalizing over the particular tag-names, there is also a suggestion on the granularity of information that is most likely to be a relevant result. The following two observations present themselves. First, the most frequent elements such as paragraphs and (sub)sections are of relatively short-length.

Indexing Units. Table 1. Distribution of relevant elements over tag names (reproduced from [8])

2002 assessments			2003 assessments		
Tag-name	Frequency	%	Tag-name	Frequency	%
⟨p⟩	383	27.47%	⟨sec⟩	303	20.89%
⟨article⟩	309	22.16%	⟨p⟩	303	20.89%
⟨sec⟩	291	20.87%	⟨article⟩	172	11.86%
⟨ss1⟩	115	8.24%	⟨bdy⟩	167	11.51%
⟨bdy⟩	90	6.45%	⟨ss1⟩	146	10.06%
⟨ip1⟩	61	4.37%	⟨ip1⟩	69	4.75%
⟨ss2⟩	25	1.79%	⟨ss2⟩	36	2.48%
⟨abs⟩	22	1.57%	⟨fig⟩	32	2.20%
⟨fm⟩	13	0.93%	⟨app⟩	20	1.37%
⟨st⟩	11	0.78%	⟨bb⟩	19	1.31%
⟨item⟩	8	0.57%	⟨art⟩	18	1.24%
⟨app⟩	7	0.50%	⟨bm⟩	17	1.17%

Indexing Units. Table 2. Example Document and Indexing Units

Example Document	Indexing subtrees	Indexing disjoint nodes
⟨article⟩ ⟨title⟩XXX⟨/title⟩ ⟨abstract⟩YYY⟨/abstract⟩ ⟨body⟩ ⟨sec⟩ZZZ⟨/sec⟩ ⟨sec⟩VVV⟨/sec⟩ ⟨/body⟩ ⟨/article⟩	1. ⟨article⟩ XXX YYY ZZZ VVV⟨/ article⟩	1. ⟨title⟩XXX⟨/title⟩
	2. ⟨title⟩XXX⟨/title⟩	2. ⟨abstract⟩YYY⟨/abstract⟩
	3. ⟨abstract⟩YYY⟨/abstract⟩	3. ⟨sec⟩ZZZ⟨/sec⟩
	4. ⟨body⟩ZZZ VVV⟨body⟩	4. ⟨sec⟩VVV⟨/sec⟩
	5. ⟨sec⟩ZZZ⟨/sec⟩	
	6. ⟨sec⟩VVV⟨/sec⟩	

Second, there is great variety of elements regarded as relevant. Even for a single topic there is a very similar variety of elements, making clear that relevancy is both depending on the topic of request, and on the precise structure of the document at hand.

Recall the question of what to put in the index. The most obvious approach is to index all information in the structured text. But already here different options present themselves, as is illustrated in Table 2. On the left-hand side of Table 2 is a very simple example document, an article with title, abstract and two sections. The first approach, shown in the middle of Table 2, is to index every retrievable unit, in this case every XML element. In a "bag of words" approach all six XML elements of the document are indexed separately, but each with all the content or text contained inside the element. That is, an element is indexed with both the text nodes directly contained in it, and all text nodes of its descendants. The indexing of subtrees of the XML hierarchy is known as the element-based approach [13]. Indexing subtrees is closest to traditional information retrieval since each XML node is a *bag of words* of itself and its descendants, and can be scored as ordinary plain text document. This directly relates structured text retrieval to the more general and well-understood problem of document retrieval. The indexing scheme is only using the structure to decompose the document into all retrievable units, and hence is applicable to any structured text. The main disadvantage is that it leads to a highly redundant index: text occurring at depth n of the XML tree is indexed n times.

On the right-hand side of Table 2, an alternative approach is shown, in which the text is only indexed once at the node where it occurs. The ⟨article⟩ and ⟨body⟩ elements are missing since they have no textual content. Since there are only four elements with content, the index is much smaller, and there is no redundancy of information. But the main advantage of indexing disjoint nodes is also creating a new problem of scoring higher level nodes. For example, as shown above, the whole article is a reasonably attractive XML element type, but it may not even occur in the index. This creates both a practical and a fundamental problem. The practical problem is that articles may contain thousands of XML elements, and hence may require considerable propagation of scores over the navigational axis. The fundamental problem is that it is not evident how to aggregate scores to ancestor elements, and various approaches exist in the literature. One of the earliest proposals is the augmentation approach of Abolhassini et al. [1], a straightforward propagation of scores to ancestor elements. The following simplified example illustrates the main idea behind augmentation. Assume the following document.

⟨body⟩
  ⟨sec⟩cat...⟨/sec⟩
  ⟨sec⟩dog...⟨/sec⟩
⟨/body⟩

and a query consisting of the two terms "cat dog." Furthermore assume that: /body/sec[1] scores 0.7 for cat; /body/sec[2] scores 0.4 for dog; and the rest 0 (that is, dog does not occur in the first section, and cat not in the second section). The problem is to determine the score of the body, which is not indexed itself. The *augmentation* approach propagates scores up with a certain weight (the augmentation factor) which is set to 0.3 based on experiments. The motivation for the augmentation factor is to avoid larger elements accumulating scores, and thus (almost) always get higher rankings than elements deep in the hierarchy. So in this case, the /body[1] will score $0.3 * 0.7 = 0.21$ for cat; and $0.3 * 0.4 = 0.12$ for dog. So the element /body[1], although not in the index, will be returned. In fact, it will be the highest ranked result for "andish" query evaluation where only results containing all query terms are returned. A very similar approach is taken by the GPX model and its decay factor [6]. The element specific language models of Ogilvie and Callan [12] provide an elegant alternative approach within the language modeling framework. Here, every XML element forms a particular language model, and the ancestor elements are modeled as mixture language models of their direct children. A final alternative is to just propagate term frequencies and effective reconstructing the element-based index discussed above, e.g., using the region models of Burkowski [2].

A third indexing approach is to index only selected elements in their entirety. This is essentially the approach taken in the FERMI model of Chiaramella [3]. The selection is tailored to the collection at hand, usually based on the human interpretation of the tags in the collection. An example of this approach is to index particular types of elements separately [10]. Mass and Mandelbrod [10] create separate indexes for articles (⟨article⟩, abstracts ⟨abs⟩, sections ⟨sec⟩, subsections ⟨ss1⟩, sub-subsections ⟨ss2⟩, and paragraphs ⟨p⟩ and ⟨ip1⟩). Each index provides statistics tailored to particular components, which may be an advantage if language statistics deviate significantly between element types. This is essentially a distributed approach where queries are issued to all indexes, and the results of each of the indexes are combined after score normalization. The selective indexing approach turned out to be effective for the IEEE collection used at INEX. The requirement to select particular element-types makes it strongly collection-dependent, and it is less straightforward to apply this indexing approach to arbitrary structured text.

Three prototypical indexing approaches for structured text have been discussed above. Some observations present themselves. First, there is a trade-off between exploiting the document structure, and being generically applicable to all structured text. The element-based approach uses the document structure for decomposing documents by focusing on the hierarchical structure only. This ignores (potentially) useful structure like the tag-names, their attributes, the schema or DTD, etc. However, to phrase it positively, since it is completely schema-ignorant the approach can handle data with any type of tag-structure, even mixed-schema XML. The selective indexing approach is on the opposite side of the spectrum. Here, the specific tags and their semantics and importance have to be taken into account when selecting the element types to index. In cases where it is known what the more important elements are, this gives powerful handles to exploit this information. The downside is, of

course, that the particular choice of element to index, and thereby the effectiveness of the approach, is completely dependent on the collection at hand. Second, there is a trade-off between indexing and query time complexity. The element based approach seems unattractive since its index is highly redundant: text appearing in a given element, will also appear in all the index entry for all the ancestors of this element. This may not be as undesirable as it may appear at first glance, since it can be viewed as a trade-off between query time and storage space complexity. The redundant index has essentially "precomputed" term frequencies per element, that otherwise need to be computed at query run time, and hence has relatively low query time complexity. The indexing complexity of the disjoint nodes approach requires much less storage space. However, an article may contain thousands of XML elements, and hence will require considerable propagation of scores over the navigational axes of the document at query time. Third, the indexing methods and retrieval models are standard information retrieval approaches or straightforward extensions of them.

Cross-references

- ► Aggregation-based Structured Text Retrieval
- ► Evaluation metrics for structured text retrieval
- ► INitiative for the Evaluation of XML Retrieval
- ► Propagation-based structured text retrieval
- ► XML retrieval

Recommended Reading

1. Abolhassani M., Fuhr N., and Malik S. HyREX at INEX 2003. In N. Fuhr, M. Lalmas, and S. Malik, editors, Proc. 2nd Int. Workshop of the Initiative for the Evaluation of XML Retrieval, 2004, pp. 27–32.
2. Burkowski F.J. Retrieval activities in a database consisting of heterogeneous collections of structured text. In Proc. 15th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 1992, pp. 112–125.
3. Chiaramella Y. Browsing and querying: Two complementary approaches for multimedia information retrieval. In Hypertext - Information Retrieval - Multimedia, 1997, pp. 9–26.
4. Cutler M., Shih Y., and Meng W. Using the structure of HTML documents to improve retrieval. In Proc. 1st USENIX Symp. on Internet Tech. and Syst., 1997.
5. Fuhr N., Gövert N., and Rölleke T. DOLORES: A system for logic-based retrieval of multimedia objects. In Proc. 19th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 1996, pp. 257–265.
6. Geva S. GPX – Gardens Point XML IR at INEX 2004. In Proc. 3rd Int. Workshop of the Initiative for the Evaluation of XML Retrieval, 2005, pp. 211–223.
7. INEX. INitiative for the Evaluation of XML Retrieval, 2007. http://inex.is.informatik.uni-duisburg.de/.
8. Kamps J., de Rijke M., and Sigurbjörnsson B. Length normalization in XML retrieval. In Proc. 30th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 2004, pp. 80–87.
9. Kazai G., Lalmas M., and Reid M. Construction of a test collection for the focussed retrieval of structured documents. In Proc. 25th European Conf. on IR Research, 2003, pp. 88–103.
10. Mass Y., and Mandelbrod M. Retrieving the most relevant XML components. In Proc. 2nd Int. Workshop of the Initiative for the Evaluation of XML Retrieval, 2004, pp. 53–58.
11. Myaeng S.H., Jang D.H., Kim M.S., and Zhoo Z.C. A flexible model for retrieval of SGML documents. In Proc. 21st Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 1998, pp. 138–145.
12. Ogilvie P., and Callan J. Using language models for flat text queries in XML retrieval. In Proc. 2nd Int. Workshop of the Initiative for the Evaluation of XML Retrieval, 2004, pp. 12–18.
13. Sigurbjörnsson B., Kamps J., and de Rijke M. An Element-Based Approch to XML Retrieval. In Proc. 2nd Int. Workshop of the Initiative for the Evaluation of XML Retrieval, 2004, pp. 19–26.
14. Trotman A. and Sigurbjörnsson B. Narrowed Extended XPath I (NEXI). In Proc. 3rd Int. Workshop of the Initiative for the Evaluation of XML Retrieval, 2005, pp. 16–40.
15. Wilkinson R. Effective retrieval of structured documents. In Proc. 17th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 1994, pp. 311–317.

Individual Data

► Micro Data

Individually Identifiable Data

Chris Clifton
Purdue University, West Lafayette, IN, USA

Synonyms

Personally identifiable data; Personal data

Definition

Individually Identifiable Data is data that identifies the person that the data is about, or that can be used to identify that individual. This generally refers to data that contains either an identification number, or

factors relating to physical, mental, economic, cultural, or social identity that could be used to link the data to an individual. Regulatory requirements for privacy generally apply (only) to individually identifiable data.

Key Points

Individually Identifiable Data is a legal term (e.g., personal data in the EU Privacy Directive [1], or Individually identifiable health information in U.S. Healthcare Laws [2]); in general privacy laws only protect individually identifiable data. Unfortunately, it is not clearly defined in technical terms. Anonymous data is presumably not individually identifiable, but what about *k-anonymous* data? (Presumably data is individually identifiable if $K = 1$, but at what level of K is data no longer individually identifiable?) This leads to considerable difficulty in developing data management technology that balances privacy with the utility of disclosing anonymous data.

Regulatory frameworks do give some guidance; for example the U.S. Healthcare Privacy rules allow data to be considered not individually identifiable if names, geographic units of less than 20,000 people, dates of finer granularity than a year (or that can indicate age for those over 89), biometric identifiers (fingerprint, full-face images), or any identifying numbers (telephone, account, vehicle license, IP address, etc.) other than numbers generated specifically for the particular dataset.

Cross-references

- ▶ Anonymity
- ▶ K-Anonymity
- ▶ Privacy Metrics

Recommended Reading

1. Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995. The protection of individuals with regard to the processing of personal data and on the free movement of such data. Off. J. Eur. Communities, 1(281):31–50, 1995.
2. U.S. Department of Health and Human Services Office for Civil Rights. Standard for privacy of individually identifiable health information. Technical report, August 2003.

INEX

▶ INitiative for the Evaluation of XML retrieval (INEX)

Inference Control in Statistical Databases

Josep Domingo-Ferrer
Universitat Rovira i Virgili, Tarragona, Catalonia

Synonyms

Statistical disclosure control (SDC); Statistical disclosure limitation (SDL)

Definition

Inference control in databases, also known as Statistical Disclosure Control (SDC), is a discipline that seeks to protect data so they can be published without revealing confidential information that can be linked to specific individuals among those to which the data correspond. SDC is applied to protect *respondent privacy* in areas such as official statistics, health statistics, e-commerce (sharing of consumer data), etc. Since data protection ultimately means data modification, the challenge for SDC is to achieve protection with minimum loss of the accuracy sought by database users.

Historical Background

The literature on inference control started in the 1970s, with the seminal contribution by Dalenius [4] in the statistical community and the works by Schlörer [12] and others in the database community. The 1980s saw moderate activity in this field. An excellent survey of the state of the art at the end of the 1980s is [1]. In the 1990s, there was renewed interest in the statistical community and the discipline was further developed under the names of statistical disclosure control in Europe and statistical disclosure limitation in America. Subsequent evolution has resulted in at least three clearly differentiated subdisciplines:

- *Tabular data protection.* This is the oldest and best established part of SDC, because tabular data have been the traditional output of national statistical offices. The goal here is to publish *static* aggregate information, i.e., tables, in such a way that no confidential information on specific individuals among those to which the table refers can be inferred. See [13] for a conceptual survey and [7] for a software survey.
- *Queryable databases.* The scenario here is a database to which the user can submit statistical queries

(sums, averages, etc.). The aggregate information obtained by a user as a result of successive queries should not allow him to infer information on specific individuals. Since the 1980s, this has been known to be a difficult problem, subject to the tracker attack [11]. SDC strategies here include perturbation, query restriction and camouflage (providing interval answers rather than exact answers).

- *Microdata protection.* This subdiscipline is about protecting static individual data, also called microdata. It is only recently that data collectors (statistical agencies and the like) have been persuaded to publish microdata. Therefore, microdata protection is the youngest subdiscipline and is experiencing continuous evolution in the last years.

Good general works on SDC are [13,9].

Foundations

Statistical disclosure control will be first reviewed for tabular data, then for queryable databases and finally for microdata.

In spite of tables displaying aggregate information, there is risk of disclosure in tabular data release. Several attacks are conceivable:

- *External attack.* For example, let a frequency table "Job" × "Town" be released where there is a single respondent for job J_i and town T_j. Then if a magnitude table is released with the average salary for each job type and each town, the exact salary of the only respondent with job J_i working in town T_j is publicly disclosed.
- *Internal attack.* Even if there are two respondents for job J_i and town T_j, the salary of each of them is disclosed to each other.
- *Dominance attack.* If one (or a few) respondents dominate in the contribution to a cell of a magnitude table, the dominant respondent(s) can upper-bound the contributions of the rest (e.g., if the table displays the total salary for each job type and town and one individual contributes 90% of that salary, he knows that his colleagues in the town are not doing very well).

SDC methods for tables fall into two classes: non-perturbative and perturbative. Non-perturbative methods do not modify the values in the tables; the best known method in this class is *cell suppression* (CS). Perturbative methods output a table with some modified values; well-known methods in this class include *controlled rounding* (CR) and the recent *controlled tabular adjustment* (CTA).

The idea of CS is to suppress those cells (primary suppressions) that are identified as sensitive by the so-called sensitivity rules. The two most common sensitivity rules are:

- *Dominance rule (n, k).* The cell is deemed sensitive if n or less respondents contribute at least k% of the cell value.
- *p%-rule.* The cell is deemed sensitive if some respondent can estimate the contribution by another respondent within p% accuracy.

After primary suppressions, additional suppressions (secondary suppressions) are performed to prevent primary suppressions from being computed or even inferred within a prescribed protection interval using the row and column constraints (marginal row and column totals). Usually, one attempts to minimize either the number of secondary suppressions or their pooled magnitude, which results in complex optimization problems. Most optimization methods used are heuristic, based on mixed integer linear programming or network flows (e.g., [6]), most of them implemented in the τ-Argus free software package [10]. CR rounds values in the table to multiples of a rounding base. This may entail rounding the marginal totals as well. On its side, CTA modifies the values in the table to prevent the inference of values of sensitive cells within a prescribed protection interval. The idea of CTA is to find the *closest* table to the original one that ensures such a protection for all sensitive cells. This requires optimization methods, which are typically based on mixed linear integer programming. Usually CTA causes less information loss than CS.

Regarding SDC of queryable databases, there are three main approaches to protect a confidential vector of numerical data from disclosure through answers to user queries:

- *Data perturbation.* Perturbing the data is a simple and effective approach whenever the users do not require deterministically correct answers to queries that are functions of the confidential vector. Perturbation can be applied to the records on which queries are computed (input perturbation) or to the query result after computing it on the original data (output perturbation). An example of perturbation methods can be found in [5].

- *Query restriction.* This is the right approach if the user does require deterministically correct answers and these answers have to be exact (i.e., a number). Since exact answers to queries provide the user with a very powerful information, it may become necessary to refuse to answer certain queries at some stage to avoid disclosure of a confidential datum. There are several criteria to decide whether a query can be answered; one of them is query set size control, that is, to refuse answers to queries which affect a set of records which is too small. An examples of the query restriction approach can be found in [3].
- *Camouflage.* If deterministically correct non-exact answers (i.e., small interval answers) suffice, confidentiality via camouflage (CVC [8]) is a good option. With this approach, unlimited answers to any conceivable query types are allowed. The idea of CVC is to "camouflage" the confidential vector a by making it part of the relative interior of a compact set Π of vectors. Then each query $q = f(a)$ is answered with an inverval $[q^-, q^+]$ containing $[f^-, f^+]$, where f^- and f^+ are, respectively, the minimum and the maximum of f over Π.

Regarding microdata SDC, given an original microdata set V, its goal is to release a protected microdata set V' in such a way that:

1. Disclosure risk (i.e., the risk that a user or an intruder can use $\mathbf{V}'$ to determine confidential attributes on a specific individual among those in $\mathbf{V}$) is low.
2. User analyses (regressions, means, etc.) on $\mathbf{V}'$ and on $\mathbf{V}$ yield the same or at least similar results.

Microdata protection methods can generate the protected microdata set $\mathbf{V}'$ either by

- *Masking original data*, i.e., generating $\mathbf{V}'$ a modified version of the original microdata set $\mathbf{V}$ or
- *Generating synthetic data* $\mathbf{V}'$ that preserve some statistical properties of the original data $\mathbf{V}$.

Masking methods can in turn be divided in two categories depending on their effect on the original data [13]:

- *Perturbative.* The microdata set is distorted before publication. In this way, unique combinations of scores in the original dataset may disappear and new unique combinations may appear in the perturbed dataset; such confusion is beneficial for preserving statistical confidentiality. The perturbation method used should be such that statistics computed on the perturbed dataset do not differ significantly from the statistics that would be obtained on the original dataset. *Noise addition, microaggregation, data/rank swapping, microdata rounding and PRAM* are examples of perturbative masking methods.
- *Non-perturbative. Non-perturbative masking methods* do not alter data; rather, they produce partial suppressions or reductions of detail in the original dataset. Sampling, global recoding, top and bottom coding and local suppression are examples of non-perturbative masking methods.

Microdata SDC faces an inherent trade-off between loss of information (i.e., analytical utility) and disclosure risk. Given a particular dataset, the microdata SDC method optimizing that trade-off is the best choice. Approaches to modeling such a trade-off include *SDC scores*, *k-anonymity* and R-U maps (plots of disclosure risk against data utility for a certain parameterization of an SDC method).

Key Applications

There are several areas of application of SDC techniques, which include but are not limited to the following:

- *Official statistics.* Most countries have legislation which compels national statistical agencies to guarantee statistical confidentiality when they release data collected from citizens or companies. This justifies the research on SDC undertaken by several countries, among them the European Union (e.g., the CASC project [2]) and the United States.
- *Health information.* This is one of the most sensitive areas regarding privacy. For example, in the U.S., the Privacy Rule of the Health Insurance Portability and Accountability Act (HIPAA) requires the strict regulation of protected health information for use in medical research. In most western countries, the situation is similar.
- *E-commerce.* Electronic commerce results in the automated collection of large amounts of consumer data. This wealth of information is very useful to companies, which are often interested in sharing it with their subsidiaries or partners. Such consumer information transfer should not result in public

profiling of individuals and is subject to strict regulation, especially in the European Union and the United States.

Future Directions

There are many open issues in SDC, some of which can be hopefully solved with further research and some which are likely to stay open due to the inherent nature of SDC. First some of the issues that probably could and should be settled in the near future are listed:

- Identifying a comprehensive listing of data uses (e.g., regression models, association rules, etc.) that would allow the definition of data use-specific information loss measures broadly accepted by the community; those new measures could complement and/or replace the generic measures currently used. Work in this line has been started in Europe in 2006 under the CENEX SDC project sponsored by Eurostat.
- Devising disclosure risk assessment procedures which are as universally applicable as record linkage while being less greedy in computational terms.
- Identifying, for each domain of application, which are the external data sources that intruders can typically access in order to attempt re-identification. This would help data protectors figuring out in more realistic terms which are the disclosure scenarios they should protect data against.
- Creating one or several benchmarks to assess the performance of SDC methods. Benchmark creation is currently hampered by the confidentiality of the original datasets to be protected. Data protectors should agree on a collection of non-confidential original-looking data sets (financial datasets, population datasets, etc.) which can be used by anybody to compare the performance of SDC methods. The benchmark should also incorporate state-of-the-art disclosure risk assessment methods, which requires continuous update and maintenance.

There are other issues whose solution seems less likely in the near future, due to the very nature of SDC methods. If an intruder knows the SDC algorithm used to create a protected data set, he can mount algorithm-specific re-identification attacks which can disclose more confidential information than conventional data mining attacks. Keeping the SDC algorithm secret used would seem a solution, but in many cases the protected dataset itself gives some clues on the SDC algorithm used to produce it. Such is the case for a rounded, microaggregated or partially suppressed microdata set. Thus, it is unclear to what extent the SDC algorithm used can be kept secret.

Cross-references

- ▶ Data Rank/Swapping
- ▶ Disclosure Risk
- ▶ Information Loss Measures
- ▶ *k*-Anonymity
- ▶ Microaggregation
- ▶ Microdata
- ▶ Microdata Rounding
- ▶ Noise Addition
- ▶ Non-Perturbative Masking Methods
- ▶ PRAM
- ▶ Record matching
- ▶ SDC Score
- ▶ Synthetic Microdata
- ▶ Tabular Data

Recommended Reading

1. Adam N.R. and Wortmann. J.C. Security-control for statistical databases: a comparative study. ACM Comput. Surv., 21(4):515–556, 1989.
2. CASC. Computational aspects of statistical confidentiality, 2004. European project IST-2000-25069 CASC, Fifth FP, 2001–2004, http://neon.vb.cbs.nl/casc.
3. Chin F.Y. and Ozsoyoglu G. Auditing and inference control in statistical databases. IEEE Trans. Software Eng., SE-8:574–582, 1982.
4. Dalenius T. The invasion of privacy problem and statistics production. An overview. Statistik Tidskrift, 12:213–225, 1974.
5. Duncan G.T. and Mukherjee S. Optimal disclosure limitation strategy in statistical databases: deterring tracker attacks through additive noise. J. Am. Stat. Assoc., 45:720–729, 2000.
6. Fischetti M. and Salazar J.-J. Solving the cell suppression problem on tabular data with linear constraints. Manag. Sci., 47(7):4, 2001.
7. Giessing S. Survey on methods for tabular data protection in argus. In J. Domingo-Ferrer and V. Torra (eds.)., Privacy in Statistical Databases, LNCS. vol. 3050. Springer, 2004, pp. 1–13.
8. Gopal R., Garfinkel R., and Goes P. Confidentiality via camouflage: the CVC approach to disclosure limitation when answering queries to databases. Operat. Res., 50:501–516, 2002.
9. Hundepool A., Domingo-Ferrer J., Franconi L., Giessing S., Lenz R., Longhurst J., Schulte-Nordholt E., Seri G., and DeWolf P.-P. Handbook on Statistical Disclosure Control (version 1.0). Eurostat (CENEX SDC Project Deliverable), 2006.
10. Hundepool A., van de Wetering A., Ramaswamy R., de Wolf P.-P., Giessing S., Fischetti M., Salazar J.-J., Castro J., and Lowthian P τ-*ARGUS v. 3.2 Software and User's Manual,* CENEX SDC Project Deliverable, February 2007. http://neon.vb.cbs.nl/casc/TAU.html

11. Schlörer J. Disclosure from statistical databases: quantitative aspects of trackers. ACM Trans. Database Syst., 5:467–492, 1980.
12. Schlörer J. Identification and retrieval of personal records from a statistical data bank. Methods Inform. Med., 14(1):7–13, 1975.
13. Willenborg L. and DeWaal T. Elements of Statistical Disclosure Control. Springer, Berlin Heidelberg New York, 2001.

Infinity

► Forever

Information

► Information Quality Assessment

Information Browsing

► Information Navigation

Information Displays

► Dense Pixel Displays

Information Extraction

Heng Ji
New York University, New York, NY, USA

Definition

Information Extraction (IE) is a task of extracting pre-specified types of facts from written texts or speech transcripts, and converting them into structured representations (e.g., databases).

IE terminologies are explained via an example as follows.

- Input Sentence:

Media tycoon Barry Diller on Wednesday quit as chief of Vivendi Universal Entertainment, the entertainment unit of French giant Vivendi Universal whose future appears up for grabs.

- IE output:
  - Entities:
    Person Entity: {Media tycoon, Barry Diller}
    Organization Entity: {Vivendi Universal Entertainment, the entertainment unit}
    Organization Entity: {French giant, Vivendi Universal}
  - "Part-Whole" relation:
    {Vivendi Universal Entertainment, the entertainment unit} is part of {French giant, Vivendi Universal}.
  - "End-Position" event.

The above sentence includes a "Personnel_End-Position" event mention, with the trigger word which most clearly expresses the event occurrence, the position, the person who quit the position, the organization, and the time during which the event happened (Table 1).

Historical Background

The earliest IE system was directed by Naomi Sager of the Linguistic String Project group [11] in the medical domain. However, the specific task of information extraction was formally evaluated through the U.S. Defense Advanced Research Projects Agency (DARPA) sponsored Message Understanding Conferences (MUC) program from 1987 to 1998 [4].

There were four specific evaluations: Named entity, coreference and template element reflected in the evaluation tasks introduced for MUC-6, and template relation introduced in MUC-7.

The MUC tasks have been inherited by the U.S. National Institute of Standards and Technology (NIST) Automatic Content Extraction (ACE) program (The ACE task description can be found at http://www.nist.gov/speech/tests/ace/and the ACE guidelines at

Information Extraction. Table 1. Event extraction example

Trigger	Quit	
Person	Barry Diller	Media tycoon
Organization	Vivendi universal entertainment	The entertainment unit of french giant vivendi universal
Position	Chief	
Time-within	Wednesday	

http://www.ldc.upenn.edu/Projects/ACE/), with more general types of entities/relations/events defined. ACE includes the following tasks.

Entity Detection and Recognition

ACE defines the following terminologies for the entity detection and recognition task:

entity: an object or a set of objects in one of the semantic categories of interest

mention: a reference to an entity (typically, a noun phrase)

name mention: a reference by name to an entity

nominal mention: a reference by a common noun or noun phrase to an entity

Seven types of entities were defined: PER (persons), ORG (organizations), GPE ("geo-political entities" – locations which are also political units, such as countries, counties, and cities), LOC (other locations without governments, such as bodies of water and mountains), FAC (facility), WEA (Weapon) and VEH (Vehicle) mentioned in an input document. This task was proposed in 2000 and evaluated in English, and then expanded to include Chinese and Arabic in 2003, Spanish in 2007.

Relation Detection and Recognition

The relation detection task was proposed in 2002, aiming to find specified types of semantic relations between pairs of entities. ACE 2007 had 7 types of relations, with 19 subtypes. The following table lists some examples (Table 2).

Event Detection and Recognition

ACE defined 8 types of events, with 33 subtypes. Some examples are presented in Table 3.

Entity Translation

Entity Translation is a cross-lingual IE track at ACE 2007 to take in a document in a foreign language (e.g., Chinese or Arabic) and extract the English catalog of the entities.

Foundations

There are two main approaches to develop IE systems, described separately as follows.

Pattern Matching Based IE

Many IE systems during MUC evaluation use high-accuracy rules, dictionaries and patterns for each specific domain. For example, for the end-position event in Table 1, an IE system generates patterns such as

- [Person] quit as [Position] of [Organization]

Manually writing and editing patterns requires some skill and considerable time. So some systems have moved on to learning these patterns automatically based on an annotated corpus pre-processed by syntactic and semantic analyzers. A more comprehensive survey of pattern matching based IE approaches can be found in [8].

The above pattern acquisition is still quite costly because for particular domain a separate annotated corpus is needed. Therefore some systems have used unsupervised learning approach [10,12,13]. The general idea is to obtain a pattern if a pair of arguments (mostly names) (Arg_1, Arg_2) and their context C_{12} appear frequently in other instances of the event.

The idea of using bootstrapping to obtain patterns was first proposed by Riloff [10]. Riloff [10] manually pre-classified the documents into relevant and irrelevant, then collect and score patterns around each noun

Information Extraction. Table 2. Examples of the ACE relation types

Relation Type	Example
Agent-Artifact (User-Owner-Inventor-Manufacturer)	*Rubin Military Design, the makers of the Kursk*
ORG-Affliation (Employment)	Mr. Smith, *the CEO of Microsoft*
Gen-Affiliation (Citizen-Resident-Religion-Ethnicity)	*Salzburg Red Cross officials*
Physical (Near)	*A town some 50 miles south of Salzburg*

Information Extraction. Table 3. Examples of the ACE event types

Event Type	Example
Movement (Transport)	*Homeless people* have been *moved* to *schools*
Business (Start-ORG)	*Schweitzer founded a hospital* in *1913*
Conflict (Attack)	The *attack* on *Gaza* killed *13 people*
Personnel (Start-Position)	*Cornell Medical Center recruited 12 nursing students*
Justice (Arrest)	*Zawahiri was arrested* in *Iran*

I

phrase. Yangarber et al. [13] used seed patterns to address the limitation of manual document classification. They started with a few initial seed patterns, and then applied an incremental discovery procedure to identify new set of patterns. Both of [10,13] are based on predicate-argument or subject-verb-object structures. Sudo et al. [12] presented a new Subtree model based on dependency parsing, and proved the Subtree model can obtain higher recall while preserve high precision.

Machine Learning Based IE

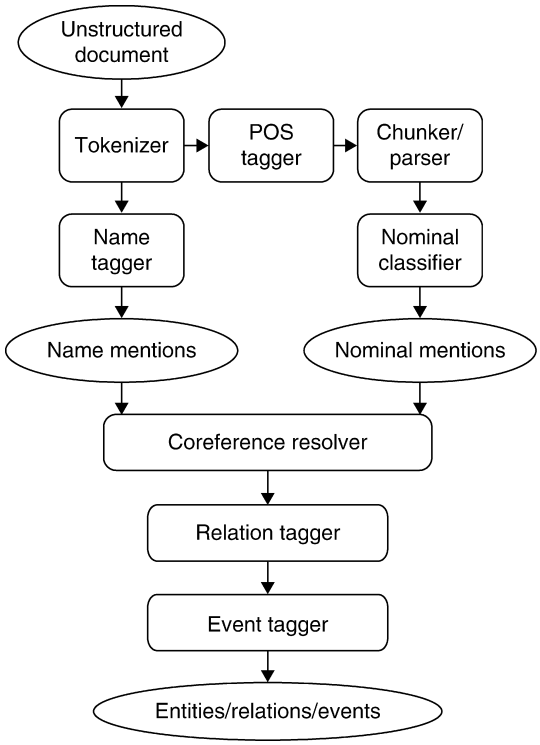
The IE systems relying entirely on pattern matching have attempted some success in MUC domains. However these patterns cannot be easily adapted into new domains. Therefore, IE research has grown by splitting the task into several components and then applying machine learning methods to address each component separately.

Machine learning based IE systems typically include name identification and classification, parsing (or partial parsing), semantic classification of nominal mentions, coreference resolution, relation extraction and event extraction. A typical IE system pipeline is presented in Fig. 1. For instance, state-of-the-art IE systems such as BBN system [2], IBM system [3] and NYU system [5] were developed in this pipeline style. This "pipeline" design provides great opportunity to applying a wide range of learning models and incorporating diverse levels of linguistic features to improve each component. Large progress has been achieved on some of these components. In the following some typical learning methods are described for the important components.

Trainable Name Tagging

The problem of name recognition and classification has been intensively studied since 1995, when it was introduced as part of the MUC-6 Evaluation. A wide variety of unified learning algorithms have been applied to the name tagging task, including Hidden Markov Models (HMMs), Maximum Entropy Models, Decision Trees, Conditional Random Fields and Support Vector Machines.

The most well-known BBN's Nymble name tagger [1] used several methods to improve performance over a simple HMM. Within each of the name class states, a statistical bigram model is employed, with the usual one-word-per-state emission. The various probabilities involve word co-occurrence, word features, and class probabilities. Since these probabilities are estimated based on observations seen in a corpus, several levels of "back-off models" are used to reflect the strength of support for a given statistic, including a back-off from words to word features.



Information Extraction. Figure 1. A minimal machine learning based IE system pipeline.

Trainable Coreference Resolution

Coreference Resolution is the task of determining whether two mentions refer to the same entity. For example in the sentence in Table 1, the name mention "Barry Diller" and the nominal mention "media tycoon" refer to the same person entity.

In a corpus-trained system, coreference resolution is usually converted into a supervised binary classification problem of determining whether a candidate mention is referring to an antecedent or not. Here an "antecedent" can be another single mention, or a cluster of mentions which the system has generated. Each pair is assigned probability value by a supervised learning based classifier. If the sampling is constructed on each mention pair, then a separate clustering algorithm is applied to group coreferring mentions.

Most coreference resolution systems use representations built out of the lexical and syntactic attributes

of the mentions for which reference is to be established [9]. A typical feature set includes:

1. Representing agreement of various kinds between mentions (number, gender)
2. Degree of string similarity
3. Synonymy between mention heads
4. Measures of distance between mentions (such as the Hobbs distance)
5. The presence or absence of determiners or quantifiers

Though gains have been made with such methods, there are clearly cases where this sort of local information will not be sufficient to resolve coreference correctly. Coreference is by definition a semantic relationship, therefore a successful coreference system should exploit world knowledge, inference, and other forms of semantic relations in order to resolve hard cases. Since 2005, researchers have returned to the once-popular semantic-knowledge-rich approach, investigating a variety of semantic knowledge sources. For example, [7] incorporated the feedback from semantic relation detection to infer and correct coreference analysis. If, for example, two library mentions which are located in two different cities, then these mentions are less likely to corefer.

Trainable Relation Detection

For ACE-type relations, various machine learning methods have been used such as K-Nearest-Neighbor [9] and Support Vector Machines [14]. The typical features used to classify relations include:

1. The heads of the mentions and their context words
2. Entity and mention type of the heads of the mentions
3. The sequence of the heads of the constituents, chunks between the two mentions
4. The syntactic relation path between the two mentions
5. Dependent words of the mentions

Trainable Event Detection

A typical event extraction pipeline includes three main steps:

1. Trigger Identification
   Identify the trigger word in a given sentence and assign event type using the probability computed from the training corpora.
2. Argument Identification
   For a given trigger and a mention, determine whether the mention is an argument of the trigger or not.
3. Argument Classification
   For an identified argument, classify the argument as a specific event role.

Event detection heavily relies high-quality deep parsing [2,5] have further shown that the predicate-argument structures can provide deeper linguistic analysis and therefore effectively enhance the performance of event detection.

Key Applications

An enormous amount of information is now available through the Web. Much of this information is encoded in natural language, which makes it accessible to some people (those who can read the particular language), but much less amenable to computer processing (beyond simple keyword search). If computers can be enabled to extract and utilize the knowledge embedded in these texts, a powerful knowledge resource for many fields will be unleashed. Some typical applications of IE are presented as follows.

1. IE for Daily News
   IE can be applied to identify the events in the daily news articles. If an informative database can be returned based on the facts extracted by IE from multiple sources of news, it can be a very valuable result and save the time a user has to spend in browsing. For example, for the news articles about Olympic sport games, an IE engine can automatically provide a table of the player's person names, the team names they come from and the game results.
2. IE for Financial Reports
   Every year the U.S. government releases the annual reports from millions of industrial agencies. The financial analysis companies then gather all these reports and analyze the most up-to-date information such as the company start-up and merge events, the competition and cooperation relations among banks or companies. It will be very helpful if an automatic IE system is applied to compress these articles into data bases. Recently such IE systems are widely applied in the financial domain to assist human analysts.
3. IE for Biology Literatures
   In the biology domain, thousands of new papers and data sets are published in natural language on a daily basis. It has become impractical for scientists to manually track all these new results and observations, and manually mine the data sets to construct

a knowledge base. IE can play a significant role by automatically generating an accurate summary of facts (e.g., gene named entities) and predicting new results (e.g., Bio-nano structures of different peptide sequences), and thus assist scientists in decision making.

4. IE for Medical Reports
   Since the early work by Sager et al. [11], IE has obtained successful applications in processing the narrative clinical documents including patient discharge summaries and radiology reports. Some of these systems have shown positive impact on providing information to assist clinical decision, result analysis, error detection, etc.

Future Directions

For each IE component there are different aspects to improve. This section proposes some high-level directions in which IE can be further explored.

1. Cross-document Information Extraction
   One of the initial goals for IE was to create a database of relations and events from the entire input corpus, and allow further logical reasoning on the database. The artificial constraint that extraction should be done independently for each document was introduced in part to simplify the task and its evaluation.

   However, almost all the current event extraction systems focus on processing single documents and, except for coreference resolution, operate a sentence at a time. Therefore, one interesting area worth exploring would be to gather together IE results from a set of related documents, and then apply inference and constraints to propagate correct results and fix the wrong information generated from the within-document IE system [6].
2. IE for Noisy Input
   Recently there has been rapid progress in applying text processing techniques on "noisy" texts such as the output of automatic speech recognition (ASR) and machine translation (MT). The potential ASR transcription and machine translation errors, in particular name recognition errors, make IE more difficult. However, it is possible to optimize the parameters in the ASR or MT systems for IE purpose. Another interesting direction would be using IE results to provide feedback to ASR and MT in a joint inference framework.
3. Cross-lingual IE
   A shrinking fraction of the world's web pages are written in a language different from the user's own, and so the ability to access information from foreign languages is becoming increasingly important. This need can be addressed in part by the research on cross-lingual IE (CLIE).
4. Active Learning for Domain Adaptation
   Since about one decade ago in MUC program, the "portability" problem has become a noticeable bottleneck for IE techniques. Until today this problem has not yet been solved. There is an urgent need to develop effective adaptation algorithms to apply IE systems to a new domain with low cost. Active learning and semi-supervised learning techniques, which have achieved success on name tagging, may be worth expanding to all stages in the IE pipeline.

Experimental Results

The state-of-the-art IE results can refer to the ACE evaluation results on NIST website (http://www.nist.gov/speech/tests/ace/). All IE results are given in terms of the entity/relation/event value scores, as produced by the official ACE scorer. These value scores include weighted penalties for missing extractions, spurious extractions, and for type errors in corresponding extractions (Scoring details can be found in the ACE07 evaluation plan: http://www.nist.gov/speech/tests/ace/ace07/doc/ace07-evalplan.v1.3a.pdf). The top systems obtained mention values in the range of 70–85, entity values in the range of 60–70, relation values in the range of 35–45, event values in the range of 15–30.

Data Sets

- ACE IE: http://projects.ldc.upenn.edu/ace/data/
  IE training data for English/Chinese/Arabic/Spanish
- CONLL 2002: http://www.cnts.ua.ac.be/conll2002/ner.tgz
  Name tagging training data for Dutch and Spanish
- CONLL 2003: http://www.cnts.ua.ac.be/conll2003/ner.tgz
  Name tagging training data for English and German

URL to Code

- UIMA: http://incubator.apache.org/uima/svn.html
  IBM NLP platform

- Jet: http://www.cs.nyu.edu/cs/faculty/grishman/jet/license.html
  NYU IE toolkit
- Gate: http://gate.ac.uk/download/index.html
  University of Sheffield IE toolkit
- Mallet: http://mallet.cs.umass.edu/index.php/Main_Page
  University of Massachusetts NLP toolkit
- MinorThird: http://minorthird.sourceforge.net/
  Carnegie Mellon University NLP toolkit

Cross-references

▶ Column Segmentation
▶ Cross-Language Mining and Retrieval
▶ Languages for Web Data Extraction
▶ Structured and Semi-Structured Document Databases
▶ Text Indexing and Retrieval
▶ Text Summarization
▶ Topic Detection and Tracking
▶ Web Information Extraction
▶ Wrapper Induction

Recommended Reading

1. Bikel D.M., Miller S., Schwartz R., and Weischedel R. Nymble: a high-performance learning name-finder. In Proc. 5th Conf. on Applied Natural Language Processing, 1997, pp. 194–201.
2. Boschee E., Weischedel R. and Zamanian A. Automatic evidence extraction. In Proc Int. Conf. on Intelligence Analysis, McLean, VA, 2005.
3. Florian R., Jing H., Kambhatla N. and Zitouni I. Factorizing complex models: a case study in mention detection. In Proc. 26th Int. Conf. Computational Linguistics, 2006, pp. 473–480.
4. Grishman R. and Sundheim B. Message understanding conference – 6: a brief history. In Proc. 16th Int. Conf. on Computational Linguistics, 1996, pp. 466–471.
5. Grishman R., Westbrook D. and Meyers A. NYU's English ACE 2005 system description. In Proc. ACE 2005 Evaluation/PI Workshop, 2005.
6. Ji H. and Grishman R. Refining Event Extraction Through unsupervised cross-document inference. In Proc. 46th Annual Meeting Assoc. for Computational Linguistics, 2008, pp. 254–262.
7. Ji H., Westbrook D., and Grishman R. Using semantic relations to refine coreference decisions. Proc. Conf. Human Language Tech. and Empirical Methods in Natural Language Proc. 2005, pp. 17–24.
8. Muslea I. Extraction patterns for information extraction tasks: a survey. In Proc. National Conf. on Artificial Intelligence (AAAI-99) Workshop on Machine Learning for Information Extraction, 1999.
9. Ng V. and Cardie C. Improving machine learning approaches to coreference resolution. In Proc. 40th Annual Meeting of the Assoc. for Computational Linguistics, 2002, pp. 104–111.
10. Riloff E. Automatically generating extraction patterns from untagged text. In Proc. 10th National Conf. on AI, 1996, pp. 1044–1049.
11. Sager N. Natural Language Information Processing: A Computer Grammar of English and its Applications. Addison Wesley, Reading, MA, 1981.
12. Sudo K., Sekine S. and Grishman R. An improved extraction pattern representation model for automatic IE pattern acquisition. In Proc. 41st Annual Meeting of the Assoc. for Computational Linguistics, 2003, pp. 224–231.
13. Yangarber R., Grishman R., Tapanainen P. and Huttunen S. Automatic acquisition of domain knowledge for information extraction. In Proc. 20th Int. Conf. Computational Linguistics, 2000, pp. 940–946.
14. Zhou G., Su J., Zhang J. and Zhang M. Exploring various knowledge in relation extraction. In Proc. 43rd Annual Meeting of the Assoc. for Computational Linguistics, 2005, pp. 427–434.

I

Information Filtering

Christian Fluhr
CEA LIST, Fontenay-aux, Roses, France

Synonyms

SDI, Selective dissemination of information; push transactions; IF

Definition

Information retrieval (IR) and information filtering (IF) are strongly related [3,5]. Information retrieval indexes a large set of documents and when a user asks a query, answers are extracted out of this set. Information filtering processes a stream of documents and for each document arriving in the system a comparison is made with one or more filtering profiles provided by users and in case of a match the document is sent to the user who created the profile. Applications of information filtering are found in competitive intelligence or technology watch. Another way of filtering is to send the document to the user only if the match is negative. This is useful for applications such as child protection or anti-spam.

Another difference is that IR queries are short, for immediate use, with answers expected as if in a conversational mode (in less than few seconds). For Information filtering, queries are permanent, can be elaborated using a long process, are often quite lengthy sometimes including relevant example documents and must be refined using feedback over time.

Historical Background

The first systems of information filtering, then called selective dissemination of information (SDI), appeared in the early 1960s [4]. For a long time they were used principally by librarians to route journal articles, using fixed profiles, to users interested in a particular domain. Because of the time lag then existing between publication, manual indexing, and bibliographic database updating, this kind of service gave delayed access to sometimes stale information.

Newswires about current events was then and remains a good source of fresh information. Companies, eager for information about their competitors, and financiers, interested in having early information about interesting companies, are large consumers of newswire.

The internet has enlarged the use of information filtering because both the number of users and the quantity of information has increased as more information becomes accessible. News filtering remains a major activity, but new activities (such as the filtering of spam and child protection in accessing internet sites) have appeared.

Another new filtering activity brought by the internet is the possibility to advertise new product availability according to user interest, even if this interest is only implicitly observed on by their browsing behavior or search engine queries. This is called *push advertising*.

Foundations

Organization of a filtering system

A filtering system (Fig. 1) processes a stream of documents. Each entering document is compared to a stored expression of user need constructed using key words, possibly limited to relevant structural parts of the input documents. If the document is considered relevant for this profile, the document is sent to the user. The following figure illustrates this process:

One might say that Information Retrieval compares a unique query with several documents whereas Information Filtering compares a unique document to several queries.

In Information Retrieval, a user generates a spontaneous, simple query and desires a rapid response. On the contrary, in Information Filtering, because the subject of the search is permanent, the user has time to elaborate and refine the query, called rather a *profile* here. The search process is continuous. It is not a conversational process though the user still wants to be notified as early as possible about relevant incoming information.

In practice, search engines providers propose a filtering service which is not based on a real filtering system but they use their search engine by periodically submitting the profile query to the database limiting the answers to documents that have arrived since the last run. Of course this is a simulation that does not guarantee the fastest access time to information but this is sufficient for many users.

Profile building and evolution

For Boolean systems, a profile can be a very long and complex equation of keywords that can have hundreds of keywords and logical operators. Generally the elaboration of such queries is performed by professional librarians.

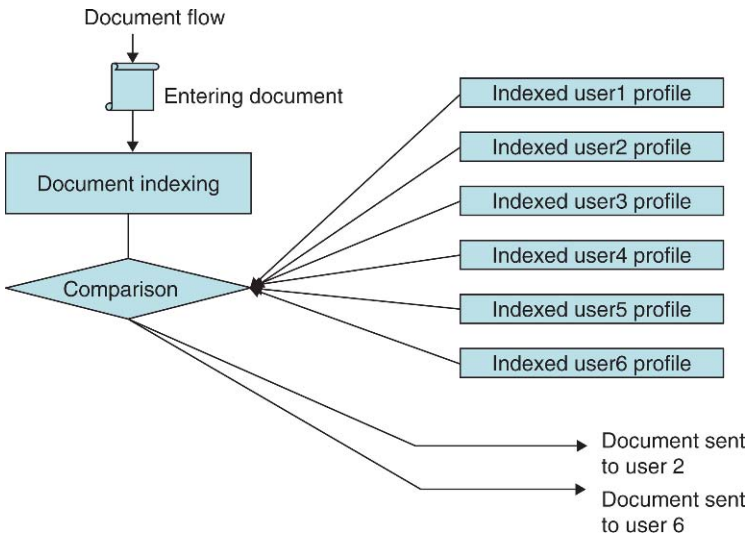
The best way to define a profile is to give a list of keywords along with some document passages relevant for the profile. Such profiles must be processed by systems that can compute a semantic proximity between the profile and the incoming document to decide relevance.

Even if the user's long term information need is stable, the profile expressing this need varies over time. At first, the initial description of user need is often incomplete. Verifying the filtering behavior of an initial set of filtered documents, the user can fine-tune, adding additional keywords, and try to refine ambiguities that he had not foreseen. This process can be simplified for the user by supervised relevance feedback. In this setup, the user need only provide positive or negative reactions to a limited number of proposed documents. According to this positive or negative judgment, the profile can be adjusted by addition or deletion of words, or by modification of their weights in the comparison.

A second reason for profile evolution over time is due to a clearer understanding of the problem on the part of the user, after viewing of retrieved documents. This learning on the part of the user can lead him to rebuild a modified profile.

Comparison procedure

Most filtering systems use the standard information retrieval approach of comparing the vector representing the document with the vector representing the profile. Because the filtering result must be a



Information Filtering. Figure 1. Organization of a filtering system

yes-or-no choice, it is necessary to create a threshold of relevance. A first difficulty is choosing a good threshold. In a filtering system, there is no database as documents arrive in a stream, so document frequency for a term is not known. This can be overcome by computing document frequencies on the fly without producing a database.

The process of attribution of a document to a profile can also be assimilated to a categorization process which assigns documents to one of a set of classes. Key words and document samples car be considered as representative of a category, addition of new relevant documents increase data that can be used to learn the difference between the two classes "good for this profile" and "bad for this profile". Popular categorization methods like Rocchio [6], KNN (K Nearest Neighbors) [9] and SVM (Support Vector Machine) [2] can be used to perform this learning.

As users are interested in a precise domain, the document as an information unit can be too large as a response. Technologies for passage retrieval can be usefully applied to solve this problem. Documents are then cut into parts that are semantically homogeneous. These separate parts are compared to profiles instead of the full document.

Relevance feedback

Because at the beginning for filtering, little information is provided for the profile, it is necessary to improve the profile through relevance feedback. The reaction of the user accepting or rejecting proposed documents gives implicit information about positive or negative influence of the words contained in these documents (Weights of terms in the original query are adjusted and new words are also added to provide a new vector representing the adjusted profile) [1].

Named entities in filtering

Named entities are proper nouns representing names of persons, of organizations, of places, of products, or numerical information like dates, measures, and percentages. They are particularly important for filtering, especially in the case of competitive intelligence.

Thematic filtering versus event filtering

According to the user need, some profiles are more domain oriented and comparison between words vectors are a good solution. Other needs are more event-oriented and necessitate more language processing to identify events that are characterized by a particular action with a particular class of actors. Examples of such event-oriented filtering can be nomination or resigning of company managers, terrorist acts, purchase of a company by another, etc.

Evaluation of filtering systems

Since the beginning of the TREC (Text REtrieval Conference) in 1992, Information Filtering was considered for evaluation. A training corpus was given to participants with queries to train their system. Afterward a new corpus was given to participants and the previously supplied queries were then applied on this new

corpus. Participants had to give an ordered list of relevant documents (1000) according to the computed relevance between documents and each query.

This was far from real usage of filtering systems. For this reason, from 1998 to 2002, a new evaluation paradigm was established to evaluate filtering systems [7]. The first difference was that only very few documents are used for training. The participants then had to make a binary decision (document relevant or not) rather than perform a ranking. In addition, documents had to be treated one after the other and for each document proposed as relevant a simulated feedback could be performed. This simulated feedback enabled the tested system to increase their effectiveness. Such process is called adaptive filtering.

Since then, new measures have been established. These measures attempt to take into consideration the different types of users. Some users are interested in having only relevant documents even if they miss many other relevant ones. Other users want to be sure to get all relevant documents but will also accept noise in the form of non relevant documents.

Because of the binary decision about relevance, it is not possible to apply measures developed for answers consisting of ranked list of ordered documents according a level of relevance such as used in the ad hoc track of TREC.

Linear utility: This measure assigns a positive or negative cost to elements in the following contingency table.

	Relevant	Not relevant
Retrieved	R+/A	N+/B
Not retrieved	R–/C	N–/D

R+, R−, N+, N− are the number of document in each category

A, B, C, D give the relative cost for each category.

For example, in TREC 11, the linear utility measure uses the following values A = 2, B = −1, C = 0, D = 0

F-beta: This is a modification of the classical F-measure [8]. It combines recall and precision with a parameter beta b which gives a relative weighting between recall and precision.

$0 <= \beta <= 1$

F-measure = $((1 + \beta) * \text{precision} * \text{recall})/((\beta * \text{precision}) + \text{recall})$

$\beta = 1$ is neutral, in TREC 2001 $\beta = 0{,}50$ which is an advantage given to precision

Key Applications

Information Filtering is a tool which is used for technology watch and competitive intelligence. It is used both in the security domain and in the civil economy. Companies need to know as early as possible what the activity of their competitors and their potential partners is, in order to decide their future on the technical level and also on the commercial and strategic level.

The other growing need for Information Filtering is the elimination of unsolicited information by filtering spam in the incoming mails. It is also applied to prevent access to some sites (pornographic, pedophile, racist; . . .) for child users of internet.

Cross-references

▶ Categorization
▶ Relevance Feedback

Recommended Reading

1. Allan J. Incremental relevance feedback for information filtering. In Proc. 19th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 1996, pp. 18–22.
2. Alsaffar A.H., Deogun J., and Sever H. 1Optimal queries in information filtering, In Proc. 12th Int. Symp. Foundations Intelligent Syst., 2000, pp. 435–443.
3. Belkin N.J. and Croft W.B. Information filtering and information retrieval: two sides of the same coin? Commun. ACM, 35 (12):29–38, December 1992.
4. Brandenberg W., Fallon H.C., Hensley C.B., Savage T.R., and Sowarby A.J. The SDI-2 system. IBM advance system Development Division report, 17–031, Yorktown-Heights, NY, April 1961.
5. Hanani U., Shapira B., and Shoval P. Information filtering: overview of issues, research and systems. User Model. User-adapt. Interact. 11:203–259, 2001.
6. Rocchio J.J. *I*The SMART retrieval system: experiments in automatic document processing, Chapter XIV. Relevance feedback in information retrieval, G. Salton (ed.). Prentice-Hall, NJ, 1971, pp. 313–323.
7. Soboroff I. Robertson S. Building a filtering test collection for TREC 2002, In Proc. 26th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 2003, pp. 243–250.
8. Van Rijsbergen K. Information retrieval, (2nd ed.). Butterworworths, London, 1979.
9. Yang Y., An evaluation of statistical approach to text categorization, report CMU-CS-97-127, Carnegie Mellon University, 1997.

Information Foraging

Peter Pirolli
Palo Alto Research Center, Palo Alto, CA, USA

Synonyms

Information seeking; Browsing

Definition

Information foraging theory [9] provides scientific predictions and explanations of information-seeking behavior in human-computer interaction (HCI). It also provides engineering models for designing and evaluating information systems and their user interfaces. Information foraging theory assumes that people adapt their information-seeking behavior to maximize their rate of gaining useful information to meet their ongoing goals. Specific models of human-information interaction are developed through (i) rational analysis of the structure and constraints of the task environment and information environment that constitute the ultimate forces driving information-seeking behavior and (ii) specification of cognitive models (user knowledge and cognitive processes) that constitute the psychological machinery behind the observed information-seeking behavior of users. Rational analysis draws upon theories and mathematical approaches developed to address rational choice under uncertainty, and specifically draws upon many models developed to address food-foraging strategies in optimal foraging theory (a branch of behavioral ecology in biology). Cognitive models are developed as programs in computational cognitive architectures that simulate human perception, action, thinking, memory, and other aspects of human information processing.

Historical Background

Information Foraging Theory arose during the 1990s, coinciding with an explosion in the amount of information that became available to the average computer user, and with the development of new technologies for accessing and interacting with information. The late 1980s witnessed several strands of HCI research that were devoted to ameliorating problems of exploring and finding electronically stored information. The confluence of increased computing power, storage, networking, and information access and hypermedia research in the late 1980s set the stage for the widespread deployment of hypermedia in the form of the World Wide Web in 1993. The emergence of the Web in the 1990s provided new challenges and opportunities for HCI. The increased wealth of accessible content, and the use of the Web as a place to do business, exacerbated the need to improve the user experience on the Web and other large-scale collections of content.

Models of information foraging were initially developed by analogy to optimal foraging theory. A typical optimal foraging model characterizes an agent's interaction with the environment as an optimal solution to the tradeoff of costs of finding, choosing, and consuming food against the energetic benefit gained from that food. Information foraging theory, however, assumes that costs of interaction are weighed against the benefits gained from gathering information and knowledge. These models attempt to understand the design of an agent's behavior by assuming that it is well engineered (adapted) for the problems posed by the environment. The two optimal foraging models that were initially applied to information seeking and browsing were the diet model (prediction of the food types that an animal selects to eat or not) and the patch model (prediction of when an animal chooses to give up food seeking in a food patch to go to another). The information diet model was applied to the prediction of how users would select some information collections and ignore others in order to maximize the benefits gained in information browsing. The information patch model was used to predict when users would give up on search in an information collection (e.g., a Web site). The first detailed application of the information diet and patch models was used to evaluate a novel document clustering browser called Scatter/Gather [10,11]. Detailed predictions of individual user behavior with Scatter/Gather were derived from computer simulations (cognitive models) of user perceptions, cognition, and action. A novel cognitive modeling technique introduced by information foraging theory predicted how users assessed cues from the user interface to make information seeking and browsing choices. *Information scent* refers to these user interface cues that guide users to the information they seek. Information foraging theory has been used to develop models of information-seeking on the Web, and interaction with information visualizations. It has also been used as a foundation for design guidelines, novel user interfaces, and automated usability evaluation tools.

Foundations

It is assumed that a user's fitness is improved to the extent that the user can gain valuable information to solve problems faced in everyday life. Increasing the rate at which people can find, make sense of, and use valuable information improves the human capacity to behave intelligently. It is assumed that adaptive information systems evolve towards states that maximize gains of valuable information per unit cost.

Users engaged in information foraging will exhibit such adaptive tendencies, and they will prefer technologies that tend to maximize the value (utility) of knowledge gained per unit cost of interaction.

Information Foraging Theory has adopted the *rational analysis* program [1]. Rational analysis addresses the questions: (i) *what* environmental problem is solved, (ii) *why* is a given behavioral strategy a good solution to the problem, and (iii) *how* is that solution realized by cognitive mechanism. The products of this approach include (i) characterizations of the relevant goals and environment, (ii) mathematical rational choice models (e.g., optimization models) of idealized behavioral strategies for achieving those goals in that environment, and (iii) computational cognitive models. Rational analysis in information foraging theory focuses on the task environment that is the aim of performance, the information environment that structures access to valuable knowledge, and the adaptive fit of the human-information interaction system to the demands of these environments. The following recipe has been proposed for rational analysis [1].

1. Precisely specify the goals of the agent.
2. Develop a formal model of the environment to which the agents is adapted.
3. Make minimal assumptions about the computational costs.
4. Derive the optimal behavior of the agent considering (1–3).
5. Test the optimality predictions against data.
6. Iterate.

The recipe focuses on optimal behavior under given goals and environmental constraints with minimal assumptions about the mechanisms that might produce such behavior.

To illustrate, rational analyses of information foraging on the Web have focused on the (i) the choice of the most cost-effective and useful browsing actions to take based on the relation of a user's information need to the perceived cues (information scent) associated with Web links and (ii) the decision of whether to continue at a Web site or leave based on ongoing assessments of the site's potential usefulness and costs. The rational analysis of information scent assumes that the goal of the information forager is to use perceived information scent cues (e.g., a Web link) to predict the utility of desired sources of content (i.e., a Web page that provides a needed answer), and to choose to navigate the links having the maximum expected utility. It is minimally assumed that the user's cognitive system represents information scent cues and information goals in cognitive structures called *chunks*. A Bayesian analysis leads to a specification of *strengths* of association, S_{ji}, among chunks that reflects the log likelihood odds of an information source associated with information cue j being relevant to a goal chunk i:

$$S_{ji} = \log\left(\frac{\Pr(i|j)}{\Pr(i)}\right), \tag{1}$$

where $\Pr(i|j)$ is the probability (based on past experience) that chunk i has occurred when chunk j has occurred in the environment, and $\Pr(i)$ is the base rate probability of chunk i occurring in the environment. Equation (1) is also known as *Pointwise Mutual Information*. Each chunk i in the user's goal is assumed to receive *activation*, A_i, from all associated information scent chunks j,

$$A_i = \sum_j S_{ji}, \tag{2}$$

and the total amount of activation received by all goal chunks is,

$$V = \sum_i A_i. \tag{3}$$

It is assumed that the utility of choosing a particular link is just the sum of activation it receives plus random noise drawn from independently and identically distributed (IID) Gumbel distributions. The probability that a user will choose link L, having a summed activation V_L, from a set of links C on a Web page, given an information goal, G is,

$$\Pr(L|G, C) = \frac{e^{\mu V_L}}{\sum_{k \in C} e^{\mu V_k}}. \tag{4}$$

where μ is a scaling parameter. Another choice facing a Web user is whether to continue navigating a particular Web site or leave. The rational analysis of this problem employs a modified patch model from optimal foraging theory. It is assumed that the user employs learning mechanisms to develop an assessment of the potential yield of a Web site, based on the user's current experiential state *x*. This *potential function* h(x) is

$$h(x) = U(x) - C(t). \quad (5)$$

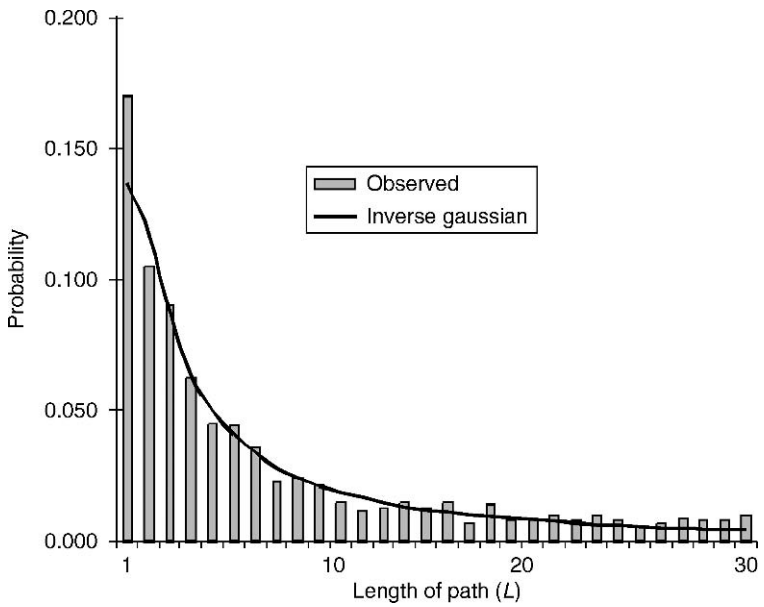
where U(x) is the utility of continued foraging in the current Web site and $C(t)$ is the *opportunity cost* of foraging for the t amount of time that is expected to be spent in the information patch. So long as the potential of the Web site is positive (the utility of continuing is greater than the opportunity cost) then the user will continue foraging.

Individual information foraging behavior can be related to the behavior of aggregates of users. For instance, the *Law of Surfing* [6] relates the rational analysis of individual Web surfing to aggregate Web user behavior. Such aggregate distributions are of practical interest because content providers on the Web often want to know how long people will remain at their Web sites. The Law of Surfing characterizes the distribution of the length, L, of sequences of page visits by Web users. Figure 1 presents a typical empirical distribution of the length of paths taken by visitors to a Web site. The distribution is skewed with the bulk of path lengths being short, and a long positive tail. The skewness of the distribution, and the long positive tail, imply that the mean of the distribution will typically be larger than the mode.

The Law of Surfing assumes that the expected utility from continuing on to the next state, X_t, is stochastically related to the expected utility of the current state X_{t-1},

$$U(X_t) = U(X_{t-1}) + \varepsilon_t, \quad (6)$$

where the ε_t are IID Gaussian distributions with mean m and variance s^2. This is known as a *Wiener process* (a Brownian motion process) with a random drift parameter m and noise s^2. It is assumed that the process continues until a threshold expected utility is reached. That is, from an initial starting page in a Web foraging episode, the expectation is that users continue browsing (surfing) according to the Wiener process specified in (6) until some threshold θ is reached. It is assumed that an individual will continue to surf until the expected cost of continuing is perceived to be larger than the discounted expected value of the information to be found in the future. In the limit, this analysis of Web surfing is the same as the analysis of *first passage times* in Brownian motion. First passage times are distributed as an Inverse Gaussian



Information Foraging. Figure 1. The Law of Surfing: The probability distribution of the length of paths surfed by users prior to leaving a web site are approximated by an inverse Gaussian distribution.

distribution: the probability density function of L, the length of sequences of Web page visits, is distributed is

$$f(L) = \sqrt{\frac{\lambda}{2\pi}}\, L^{-3/2} e^{\frac{\lambda}{2v^2 L}(L-v)^2}, \quad L > 0, \qquad (7)$$

where the parameter v is the expected value, and λ is related to the expected value and variance as

$$\frac{\lambda = v^3}{Var[L]}.$$

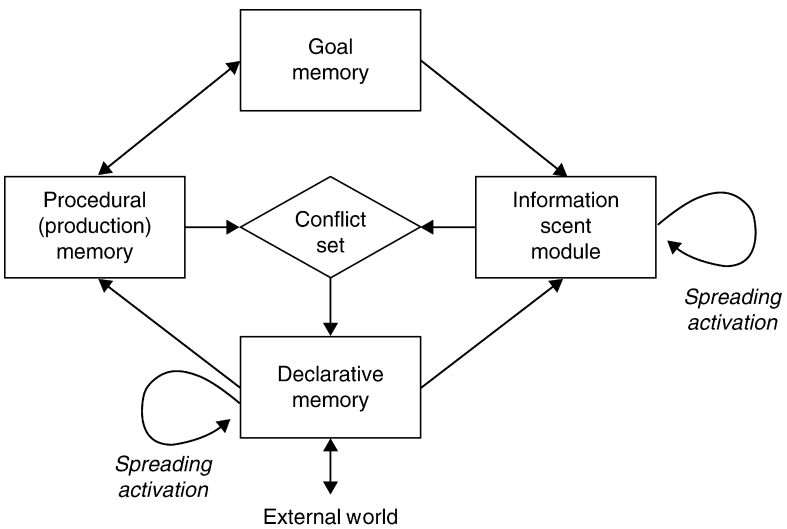
Computational cognitive models developed in information foraging theory simulate the psychological mechanisms that yield adaptive behavior. Typically, these are implemented in a simulation system that embodies a theory of *cognitive architecture* developed in psychology [2]. Theories of cognitive architecture attempt to provide a deeper account of the mechanisms underlying cognition, learning, and performance. The ACT family of *production system* theories has the longest history (since 1976) of these kinds of cognitive architectures.

Figure 2 presents the ACT-Scent cognitive architecture developed in the ACT family. It includes a module that computes *information scent*. The architecture includes a *declarative memory* containing declarative knowledge represented as chunks, which correspond to things that the user's mind is aware it knows and that can be easily described to others, such as the content of Web links, or the functionality of browser buttons, and the current user's goal (e.g., evaluating a link, choosing a link, etc.). The architecture also includes a *procedural memory*. Procedural knowledge is represented as *production rules*. For instance, the following is an English gloss of a production rule, Click-link, that is used to simulate the choice of a Web link,

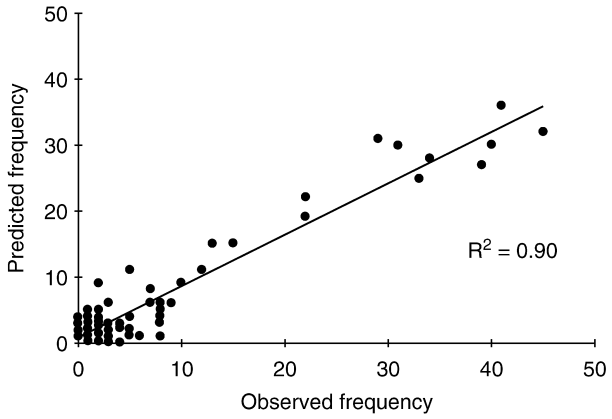
Click-link:
IF the goal is to process a link
& there is a information-seeking task
& there is a browser
& there is a link that has been read
& the link has a link description
THEN
Click on the link

If selected, the rule will execute the action of clicking on the link. A production rule has a *condition* side and an *action* side. When the all the conditions on the condition side (the "IF" portion) are *matched* to goal information and declarative memory, the production may be *fired* and when it does, the actions (the "THEN" portion) of the production will be *executed* to possibly update goals and memory, or initiate behavior. At any point in time, only a single production can fire. When there is more than one match, the matching productions form a *conflict set*. One production is then selected from the conflict set based on its utility based on information scent. The *goal memory* contains representations of intentions driving behavior. The information scent module computes the utility of actions.

Information perceived from the external world is encoded into chunks in declarative memory. Goals



Information Foraging. Figure 2. The ACT-Scent cognitive architecture used in simulating web foraging.



Information Foraging. Figure 3. The scatter plot for the observed and theoretically predicted frequency that users select links at the Yahoo! web site assessed over eight information-seeking tasks. Predictions were generated by Monte Carlo simulations using an ACT-Scent model.

and subgoals controlling the flow of cognitive behavior are stored in goal memory. The system matches production rules in production memory against goals and activated information in declarative memory and those that match form a conflict set. The matched rule instantiations in the conflict set are evaluated by utility computations in the information scent module. Based on the utility evaluation, a single production rule instantiation is executed, updates are made to goal memory and declarative memory, if necessary, and the cycle begins again. Simulations models generate predicted user behavior. Figure 3 presents the predicted frequencies for links being chosen at a popular Web site generated by Monte Carlo simulations of users working on pre-specified tasks.

Key Applications

Engineering Models of Browser Use

Graph-based algorithms based on information foraging models have been used to predict the flow of user at a Web site and identify Web site navigation problems, and this was implemented in a automated usability analysis tools available through a Web interface [5]. Dynamic programming models based on information foraging theory have been used to identify trade-offs in the design of the Scatter/Gather document cluster browser that varied over task conditions. The CogTool-Explorer [8] tool supports the rapid prototyping of user interfaces and then automatically simulates user interaction with specified designs to make performance time predictions.

Novel Search and Browsing Interfaces

Novel user interface techniques have been developed to automatically render improved information scent cues that yield more efficient user browser [7,14].

Usability Guidelines

Concepts and metaphors from information foraging theory have been influential in the development of Web usability guidelines. The concept of information scent has been used to develop Web page design guidelines aimed at producing efficient matches between user goals and Web link cues [13].

Future Directions

Information foraging theory has been extended to predictions of user interaction with highly interactive information visualizations [12,4], programming environments, and skimming and reading time allocation under varying deadline conditions. Recent work is also extending the research to social information foraging (information seeking and production by collections of users) in systems such as social bookmarking sites [9,4].

Cross-references

- ▶ Browsing
- ▶ Browsing in Digital Libraries
- ▶ Information Navigation
- ▶ Information Retrieval

- ▶ Navigation
- ▶ Searching Digital Libraries
- ▶ Usability

Recommended Reading

1. Anderson J.R. The Adaptive Character of Thought. Lawrence Erlbaum, Hillsdale, NJ, 1990.
2. Anderson J.R., Bothell D., Byrne M.D., Douglass S., Lebiere C., and Qin Y. An integrated theory of mind. Psychol. Rev., 11 (4):1036–1060, 2004.
3. Budiu R., Pirolli P., Fleetwood M., and Heiser J. Navigation in degree of interest trees. In Proc. Working Conf. on Advanced Visual Interfaces, 2006, pp. 457–462.
4. Chi E.H., Pirolli P., and Lam S.K. Aspects of augmented social cognition: social information foraging and social search. In Human Computer Interaction International, D. Schuler (ed.). Springer, 2007, pp. 60–69.
5. Chi E.H., Rosien A., Suppattanasiri G., Williams A., Royer C., Chow C., Robles E., Dalal B., Chen J., and Cousins S. The bloodhound project: automating discovery of web usability issues using the InfoScent simulator. In Proc. SIGCHI Conf. on Human Factors in Computing Systems, 2003, pp. 505–512.
6. Huberman B.A., Pirolli P., Pitkow J., and Lukose R.J. Strong regularities in World Wide Web surfing. Science, 280 (5360):95–97, 1998.
7. Olston C. and Chi E.H. ScentTrails: integrating browsing and searching on the Web. ACM Trans. Comput. Hum. Interact., 10(3):177–197, 2003.
8. Pirolli P. Exploring browser design trade-offs using a dynamical model of optimal information foraging. In Proc. SIGCHI Conf. on Human Factors in Computing Systems., 1998, pp. 33–40.
9. Pirolli P. Information Foraging: A Theory of Adaptive Interaction with Information. Oxford University Press, New York, NY, 2007.
10. Pirolli P. and Card S.K. Information foraging in information access environments. In Proc. SIGCHI Conf. on Human Factors in Computing Systems., 1995, pp. 51–58.
11. Pirolli P. and Card S.K. Information foraging. Psychol. Rev., 106:643–675, 1999.
12. Pirolli P., Card S.K., and Van Der Wege M.M. The effects of information scent on visual search in the Hyperbolic Tree Browser. ACM Trans. Comput. Hum. Interact., 10(1):20–53, 2003.
13. Spool J.M., Perfetti C., and Brittan D. Designing for the Scent of Information. User Interface Engineering, Middleton, MA, 2004.
14. Woodruff A., Rosenholtz R., Morrison J.B., Faulring A., and Pirolli P. A comparison of the use of text summaries, plain thumbnails, and enhanced thumbnails for web search tasks. J. Am. Soc. Inf. Sci. Technol., 53:172–185, 2002.

Information Graphic

- ▶ Chart

Information Hiding

- ▶ Steganography

Information Integration

Alon Halevy
Google Inc., Mountain View, CA, USA

Synonyms

Data integration; Enterprise information integration

Definition

Information integration systems offer uniform access to a set of autonomous and heterogeneous data sources. Sources can range from database systems and legacy systems to forms on the Web, web services and flat files. The data in the sources need not be completely structured as in relational databases. The number of sources in an information integration application can range from a handful to thousands.

Historical Background

Database applications are typically heavily designed and tuned for a specific context. But as data management needs in enterprises change and the information economy evolves, the need to combine information from multiple sources arises frequently. Examples of such scenarios include mergers and acquisitions, internal restructuring, the need to interoperate with third parties and to expose data on the web. Since the early 1980s the need to combine multiple heterogeneous data sources has become a research topic for the database research community. In the 1990s, as the web emerged and the many data sources came on line, the need to integrate data and for companies to work with third-parties grew, and these trends fueled the information integration research and industry.

Foundations

Information integration is a challenging problem for three different classes of reasons: systems-level, logical-level and social challenges.

The systems challenges arise fundamentally because enterprises need to enable multiple systems to talk seamlessly to each other. This is hard even when they

are all running ODBC/JDBC compliant databases, let alone radically different systems. For example, while SQL is a standard query language for relational databases, there are some differences in the way different vendors implement it. Executing queries efficiently over multiple systems is even more challenging. In addition to the difficulties that arise in distributed databases, information integration systems cannot assume that data is A priori distributed to the different nodes by one entity and in an organized and known fashion. Furthermore, the query processing capabilities of each source can be very different. For example, while one source may be a full SQL engine and therefore may be able to accept very complex queries, another source may be a web-form and therefore only accepts a very small number of query templates.

The second set of challenges has to do with the way data is logically organized in the data sources. At the core, the problem is that when schemas are designed by different people and for different applications, there will be significant differences between them, even when they model the same domain. Differences include (i) naming of tables and attributes, (ii) tabular organization of the data (or hierarchical structure in XML), (iii) domain coverage and level of detail modeled, and (iv) differences in data-level representation of objects.

The social challenges in building an information integration application, while not technical, are often very significant in practice. The first challenge may be to *find* a particular set of data. It may be hard to find a particular piece of data within a large enterprise, and in some cases, the data needed is not captured in the proper form. Even when the location of the data is known, owners of the data may not want to cooperate with an integration effort. In some cases this may happen for competitive reasons, and in others it may be because systems are carefully tuned for performance and the owners are afraid that participating in an information integration effort may adversely affect their system. Of course, privacy and security concerns are rampant.

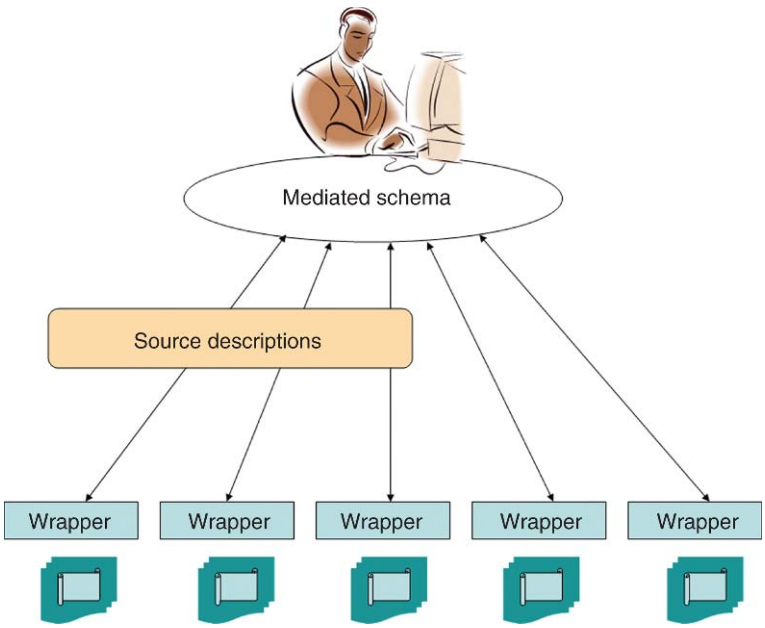
For all these reasons, the goal of information integration is to build tools that make it *easier* to build integration applications, rather than completely automating the process.

I

Information Integration Architecture

Figure 1 shows a prototypical architecture of an information integration system, often referred to as *virtual information integration*. The *data sources* are shown on the bottom of the figure. As explained earlier, data sources can vary on many dimensions, such as the data model underlying them, their schema, and their ability to process queries.

The *wrappers* are (hopefully small) programs whose role is to send queries to a data source, receive



Information Integration. Figure 1. Logical components of a virtual information integration system.

answers and possibly apply some basic transformations on the answer. For example, a wrapper to a web form source would accept a query and translate into the appropriate HTTP request with a URL that poses the query on the source. When the answer comes back in the form of an HTML file, the wrapper would extract the tuples from the HTML file. There are two main approaches to creating wrappers. The first approach is to explicitly write a set of rules for extracting the structure from the answer returned from the source. The second approach is to train the wrapper on a set of examples, and use Machine Learning techniques to learn the rules.

The top of the figure depicts the *mediated schema*. This is the schema in which users (or applications) pose queries. The mediated schema is built for the information integration application and contains *only* the aspects of the domain that are relevant to the application. Therefore, it does not necessarily contain all the attributes present in the sources, but only a subset of them. The mediated schema is not meant to store any data, but is purely a logical schema.

The key to building an information integration application are the *source descriptions*. These descriptions specify the properties of the sources that the system needs to know in order to use their data. The main component of source descriptions are the *semantic mappings*, that relate the schemata of the data sources to the mediated schema. The semantic mappings specify how attributes in the sources correspond to attributes in the mediated schema (when such correspondences exist), and how the different groupings of attributes into tables are resolved. In addition, the semantic mappings specify how to resolve differences in how data values are specified in different sources. It is important to emphasize that the virtual information integration architecture only requires specifying mappings between the data sources and the mediated schema and not between every pair of data sources. Hence, the number of mappings that need to be specified is the same as the number of sources and not the square of that number.

Schema Mediation Languages

There has been quite a bit of research on developing appropriate languages for specifying schema mappings. The common theme to these languages is that they use query expressions for the mappings, and they differ in how they use these expressions.

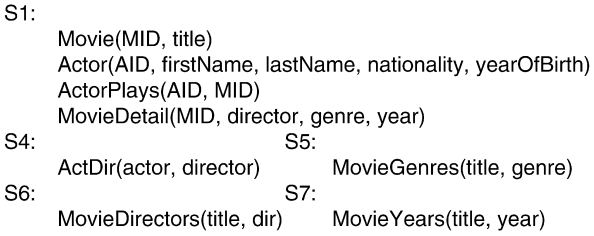
The Global-as-View (GAV) language describes the mediated schema as a set of view definitions over the source relations. In contrast, the Local-as-View language (LAV) describes the data sources as views over the mediated schema. The GLAV language combines the expressive power of GAV and LAV. The following example illustrates these differences.

Example: Consider a mediated schema with the following three relations: `Movie(title, director, year, genre)`, `Actors(title, name)` `Plays(movie, location, startTime)` *and the sources shown in* Figure 2.

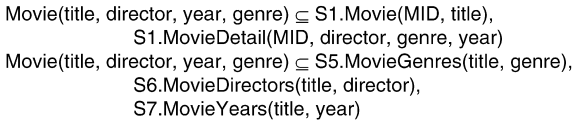
The mappings shown in Figure 3 are GAV schema mappings. Note how Movie is defined to be the union of two conjunctive queries.

The $\subseteq$ *symbol in the mapping denotes the* open-world assumption*: that is, the data sources may not contain all the data in the domain.*

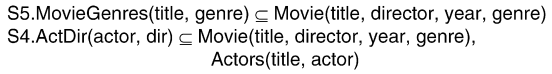
The mappings shown in Figure 4 are LAV schema mappings. The first description states that S5 *contains a subset of the projection on a relation in the mediated*



Information Integration. Figure 2.



Information Integration. Figure 3.



Information Integration. Figure 4.

schema, and the second description states that S4 *is a subset of the join of two relations in the mediated schema. Neither of these sources can be described in GAV.*

As shown, the main advantage of GAV is that query processing is conceptually easier. The main benefit of LAV over GAV is that describing a data source does not require knowing which other data sources exist in the system. Hence, it is easier to add more sources to the system. Furthermore, since the source descriptions in LAV can leverage the expressive power of the view definition language, it was easier to describe precise constraints on the contents of the sources and describe sources that have different relational structures than the mediated schema. Describing such constraints is crucial because it enables the system to select a minimal number of data sources relevant to a particular query.

User queries are posed in terms of the relations in the mediated schema. Hence, the first step the system must do is *reformulate* the query into queries that refer to the schemas of the data sources. To do so, the system uses the source descriptions. The result of the reformulation is a set of queries that refer to the schemata of the data sources and whose combination will yield the answer to the original query. The result of reformulation is referred to as a *logical query plan*.

Hence, an important aspect of schema mapping languages is developing algorithms for (and understanding the complexity of) query reformulation. For GAV, since the mediated schema is defined as a set of views over the sources, query reformulation amounts to query unfolding. In the case of LAV, since sources are described as views, query reformulation amounts to rewriting queries using views. While, in general, the complexity of answering queries using views is exponential, there are several algorithms that work efficiently in practice. Query reformulation in GLAV is a combination of answering queries using views followed by a query unfolding step.

In addition to schema mappings, source descriptions also need to consider (i) access-pattern limitations to sources: for example, a database served behind a web form or web service typically requires a set of inputs to serve tuples, and (ii) completeness (or partial completeness) of data sources: when a data source is known to be complete on a particular slice of the domain, the system can eliminate the need to access other data sources. The differences between complete and incomplete sources are formulated in terms of the open-world (closed-world) assumption.

Generating Schema Mappings

A major bottleneck in setting up an information integration application is the effort requires to create the source descriptions, and more specifically, writing the semantic mappings between the sources and the mediated schema. Writing such mappings (and maintaining them) required database expertise (to express them in a formal language) and business knowledge (to understand the meaning of the schemas being mapped).

To address this problem, several techniques for semi-automatic schema mapping have been developed with the goal of reducing the time it takes a human to create mappings. These techniques rely on several principles. First, the techniques explore methods to map between schemas based on clues that can be obtained from the schemas themselves, such as linguistic similarities between schema elements and overlaps in data values or data types of columns. Second, based on the observation that none of the above techniques is foolproof, the next development involved systems that combined a set of individual techniques to create mappings. Finally, based on the observation that schema mapping tasks are often repetitive, novel schema mapping methods incorporate Machine Learning techniques that enable the system to leverage past work. It should also be noted that the process of generating schema mappings is typically divided into two steps. In the first step, referred to as *schema matching*, the system generate *correspondences* between attributes (or other schema elements) in the two schemas. In the second step, the system builds on the correspondences and creates expressions in the mapping language.

In addition to the schema level, an information integration system also needs to reconcile references at the data level. There are often cases where the same object in the world is referenced in different ways in data sets (e.g., people, addresses, company names, genes). The problem of reference reconciliation is to automatically detect references to the same object and to collapse them. Unlike reconciling schema heterogeneity, the amounts of data are typically much bigger, and therefore these techniques need to put more emphasis on being mostly automatic. These techniques typically rely on some sophisticated forms of string matching, augmented by looking at neighboring values in the same row of a table to gather additional match clues.

Information integration is one of several contexts in which mappings between data sources need to be

created, manipulated, composed and inverted. The area of Model Management was developed to provide a formal framework for supporting generic operations on schemas and mappings between them. Knowledge Representation languages, and in particular, Description Logics, have also been shown to offer benefits in modeling data sources in a flexible fashion. In a sense, Description Logics offer a schema and query language that can also express very rich constraints, but still support effective reasoning. More generally, information integration has been an active topic also in the field of Artificial Intelligence, leveraging techniques from Knowledge Representation, Machine Learning and Automated Planning.

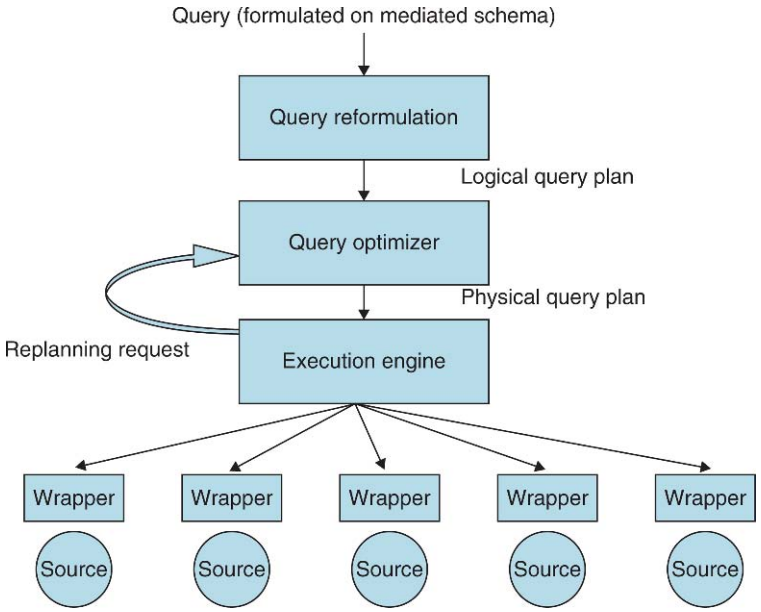
Query Processing

Given a logical query plan, it needs to be optimized and executed by the system (see Fig. 5). Herein lies the second main difference between database systems and information integration systems. Unlike the conventional database setting, an information integration system cannot neatly divide its processing into a query optimization step followed by a query execution step. The contexts in which an information integration system operate are very dynamic and the optimizer has much less information (e.g., statistics) than the traditional setting. As a result, two things happen: (i) the optimizer may not have enough information to decide on a good plan, and (ii) a plan that looks good at optimization time may be arbitrarily bad if the sources do not respond exactly as expected. To address these challenges, researchers developed techniques for *adaptive query processing*. (Adaptive query processing was already investigated in traditional database systems because even there, the estimates of the optimizer may be wrong, leading to bad plans).

The key idea of adaptive query processing is that the engine may decide to change the query plan *during* execution. The variations on the techniques developed have to do with how and when the plan can be changed. For example, one strategy may consider changing the plan only at materialization points, or the system may put conditionals into the plan that examine the intermediate results and decide what to do next. More aggressive techniques consider a different plan for every single tuple, while others continuously monitor the execution and change plans when it appears as if there is a better global plan. One of the challenges that each of these techniques needs to keep in mind is to minimize wasted computation (i.e., intermediate results that are computed and then discarded).

Related Data Management Architectures

Virtual information integration is one of several architectures for sharing and integrating data. Before the development of virtual information integration, data



Information Integration. Figure 5. Components of an information integration system.

warehousing was the common method for integrating data from multiple sources. However, data warehousing suffered from the fact that the data is loaded only periodically into the warehouse and therefore may be stale. Furthermore, data warehousing requires a single physical store thereby limiting the range of contexts in which data can be shared.

The emergence of peer-to-peer file sharing systems inspired the data management research community to consider P2P architectures for data sharing. The advantage of peer-data management systems is that it is no longer necessary to create a single mediated schema in cases where such a schema is hard to build or agree upon.

Finally, data exchange refers to an architecture that includes source and target databases, and the semantic mappings specify how to populate the target from data in the source. Many of the same issues encountered with GLAV mappings also occur here.

The Information Integration Industry

Beginning in the middle 1990s, information integration moved from the lab into the commercial arena. Today, this industry is known as Enterprise Information Integration (EII). The vision underlying this industry is to provide tools for integrating data from multiple sources *without* having to first load all the data into a central warehouse as required by previous solutions. Broadly speaking, the architectures underlying the products were based on the principles investigated in the research arena.

Some of the first applications in which these systems were fielded successfully were customer-relationship management, where the challenge was to provide the customer-facing worker a *global view* of a customer whose data is residing in multiple sources, and digital dashboards that required tracking information from multiple sources in real time. Of the many challenges faced by the industry, perhaps the greatest one was the business question of whether to build a horizontal platform that can be used in any application or to build special tools for a particular vertical. The argument for the vertical approach was that customers care about solving their *entire* problem, rather than paying for yet another piece of the solution and having to worry about how it integrates with other pieces. In addition, there are challenges in integrating with other middleware tools, such as Enterprise Application Integration tools.

In addition to the enterprise market, information integration has also played an important role in internet search. For example, the *vertical search* market focuses on creating specialized search engines that integrate data from multiple deep web sources in specific domains (e.g., travel, jobs). These engines also embed complex source descriptions.

Finally, information integration has also been a significant focus in the life sciences, where diverse data is being produced at increasing rates, and progress depends on researchers' ability to synthesize data from multiple sources. Personal Information Management is also an application where information integration is taking a significant role.

Key Applications

Some of the key applications of information integration are:

1. Enterprise data management, querying across several enterprise data repositories,
2. Accessing multiple data sources on the web (and in particular, the deep web),
3. Large scientific projects where multiple scientists are independently producing data sets,
4. Coordination accross mulitple government agencies.

Future Directions

With all the progress to date, the set-up time for information integration systems is still too long. To set up an information integration application, one still needs to create a mediated schema, and create semantic mappings to obtain any visibility into the data sources. An important research challenge is to provide as many services as possible with as little set up time as possible. The management of *dataspaces* emphasizes the idea of pay-as-you-go data management: offer some services immediately without any setup time, and improve the services as more investment is made into creating semantic relationships.

To support pay-as-you-go data management, the information integration system needs to model and reason about uncertainty. Uncertainty can appear in several forms: in the underlying data, the imprecise nature of semantic mappings and vaguely specified queries (i.e., keyword queries). Hence, incorporating uncertainty into data management and in particular,

to information integration systems, is an important challenge going forward.

Cross-references

- ▶ Adaptive Query Processing
- ▶ Model Management
- ▶ Query Rewriting Using Views
- ▶ Query Translation
- ▶ View-Based Data Integration
- ▶ XML Information Integration

Recommended Reading

1. Deshpande A., Ives Z., and Raman V. Adaptive query processing. Foundations and Trends in Databases. Now Publishers, 2007. http://www.nowpublishers.com/dbs.
2. Franklin M., Halevy A., and Maier D. Dataspaces: a new abstraction for data management. ACM SIGMOD Rec. 34(4):27–33, December, 2005.
3. Haas L. Beauty and the beast: The theory and practice of information integration. In Proc. 11th Int. Conf. on Database Theory, 2007, pp. 28–43.
4. Halevy A.Y. Answering queries using views: a survey. VLDB J., 10(4), 2001.
5. Halevy A.Y., Ashish N., Bitton D., Carey M.J., Draper D., Pollock J., Rosenthal A., and Sikka V. Enterprise information integration: successes, challenges and controversies. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 2005, pp. 778–787.
6. Lenzerini M. Data integration: a theoretical perspective. In Proc. 21st ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems, 2002, pp. 233–246.
7. Rahm E. and Bernstein P.A. A survey of approaches to automatic schema matching. VLDB J., 10(4):334–350, 2001.

Information Integration Techniques for Scientific Data

Amarnath Gupta
University of California San Diego, La Jolla, CA, USA

Synonyms

Graph theory; Graph data structure; Graph database

Definition

Information integration refers to the field of study of techniques attempting to combine information from disparate sources despite differing conceptual, contextual and lexical representations. One goal of information integration, commonly held by the data management community, is to combine the information in such a way that the user gets a unified view of the data. In other words, the user should see and query the data as though it is present in a common, unified schema.

In the domain of scientific applications, the problems and expectations are different. In this domain the semantics of data play a very strong role in data integration, and semantic compatibility need to be ensured as part of the data integration process. Further, straightforward view-based data integration, which works well for commercial applications, does not always suit the needs of the scientific users. Finally, scientists need to ensure that the result of any query on integrated data is scientifically valid.

Historical Background

Database applications are typically heavily designed and tuned for a specific context. But as data management needs in enterprises change and the information economy evolves, the need to combine information from multiple sources arises frequently. Examples of such scenarios include mergers and acquisitions, internal restructuring, the need to interoperate with third parties and to expose data on the web. Hence, from the early 1980s the need to combine multiple heterogeneous data sources has become a research topic for the database research community. In the 1990s, as the web emerged and the many data sources came on line, the need to integrate data and for companies to work with third-parties grew, and these trends fueled the information integration research and industry.

Foundations

There are several technical challenges in information integration in science.

Semantic Heterogeneity

Semantic heterogeneity [6] refers to a problem that occurs in both scientific and commercial applications and arises when the data to be integrated have been developed by different groups for different purposes. An aspect of semantic heterogeneity is schema heterogeneity (one schema uses two attributes "firstName" and "lastName," and another uses a single attribute "fullName"); another aspect is data heterogeneity ("IBM" in one database versus "International Business Machines" in another). A more complex kind of semantic heterogeneity involves generalization, where one database has a value like "thoracic surgeon" and another has a value "surgeon." This difference can be

reconciled if the system uses the additional knowledge that a "thoracic surgeon is a surgeon," and have an integration rule that a narrower term can be mapped to the more general term. In scientific databases, the reconciliation between schema and data heterogeneities becomes more complex. The term "Purkinje cell" can be mapped to the more general term "neuron" if the anatomical location of the Purkinje cell is known to be the brain, it should map to "muscle cell" if the location is known to be the heart, otherwise it will map to the even more general term "cell." Similar mapping rules can be defined at the schema level. Thus, one needs a more extensive, yet easily computable, mapping logic, and an integration framework that uses this extra mapping layer. Hakimpour and Geppert [5] and Ludäscher et al. [7] make use of ontologies or other semantic structures to specify these semantic mappings.

Role of Metadata in Managing Semantic Heterogeneity

Metadata refers to additional information about data that cover the scope, the creation, the coverage and so forth that is not directly represented in the data. In science, the norm is to call the experimental output as the "data" and other related information like the experimental protocol, the people conducting the experiment and recording the data etc. as metadata. Metadata can play a partial role in resolving some forms of semantic heterogeneity in both commercial and scientific applications. Suppose there are two tables in two sources where each contains a variable called "oxygen-saturation." Is it possible to create integrated views that would perform unions of the tables, or a join operation with "oxygen-saturation" as the joining column? Since they both have the same data types and notionally represent the same physical quantity, it might seem that combining the relations using this variable is a valid operation. However, if it is known that one database has a daily value of this variable, while the other database has monthly averages, then the answer does not remain so obvious any more. One role of metadata is to capture this kind of information such that the information integration rules can use additional guard conditions based on metadata.

Functional Relationships

Functions play a significant role in scientific applications. The data exposed to a user is often the result of a computation. For example, in many geospatial applications, the value of a variable at a specific location is not directly stored and needs to be provided through interpolation. This is one of the many typical cases where the user can get access to the data only through functions [2]. In an integration scenario, the mapping between two data elements coming from two different data sources may also be established through functions. Suppose there are two sources S1 and S2, and both provide a variable Y when a parameter X is supplied to them. When an integration system accesses S1, it has to use the function f1, and for S2 it has to use the function f2. Suppose, the integration system takes a value x1 and sends it to both sources by executing f1(x1) and f2(x1), and gets back values y1 and y2 respectively. The relationship between y1 and y2 may need to be determined by mapping functions g1 and g2 to be applied to the two sources respectively. Thus if the values returned are equivalent then g1(f1(x1)) will be equivalent to g2(f2(x1)). If the sources need to be integrated through a view that requires a semijoin, transformation functions are needed to convert the output of one source to input of another source. In actual scientific applications that involve wider variety of data types, these functions can be complicated and specialized and need a separate service. While information integration products like DB2 DiscoveryLink from IBM allow sources to export functions, optimized techniques for data integration with functions is still an area of research.

Non-schematic Integration

Not all scientific data integration can be done by schema integration, or even schema integration extended by semantic mapping structures. This is especially true if the task of the "integration" is to compute a cascade of functions whose parameters come from different sources. This form of information integration is covered in detail in the entry *Scientific Workflow Management*. A different class of non-schematic information integration occurs through statistical tools. Critchlow et al. [3] reviews a number of "data integration" tools in the domain of drug discovery. All these two use multiple data sources from domain specific data types like microarrays and protein interaction graphs, and use statistical structures like decision trees and Bayesian networks to achieve data combination. How data management techniques can be effectively used for this class of data integration remains an open issue.

New Issues

Interactive Information Integration The ability to visualize information is common to many scientific disciplines. Scientists often find the fully automated schema mapping and integration techniques developed by computer scientists to be less useful because they would like the data integration process to be more interactive and exploratory. In this mode of integration, the user needs a system's guidance to discover sources that may potentially be integrated, would use exploratory tools to investigate which subset of data from different sources might scientifically qualify for integration, and provide additional semantic input at query time to ensure compatibility of data that need to match each other. Research projects are currently underway to explore how interactivity can be introduced in information integration and querying.

Quality of Integration

A common goal in any science task is to develop a better understanding of a phenomenon by observing, modeling and computation. In the case of information integration, the desired goal is to ensure that when two previously unconnected pieces of information are combined, the resulting product of integration will provide a better or more accurate or novel understanding that the unconnected data pieces could not have provided by themselves. A real impediment in achieving this is that scientific data often has uncertainties and suffers from obsolescence due to the progressive nature of scientific information. Further, data from different sources can be contradictory or incompatible and combining them may need additional techniques that a data integration system needs to support. A formal framework is therefore needed to assess the "goodness" of the integration. Qi et al. [8] has taken an initial step for integrating conflicting data in an archeological application.

Key Applications

A number of scientific information integration systems have developed over the last decade. A few of them are listed below.

BIRN

The Biomedical Informatics Research Network (http://www.nbirn.net) has developed a semantic information integration system for Neuroscience applications. It uses a relational Global-As-View Mediator [4] extended with a semantic network of inter-term relationships to achieve scientific information integration.

Data Foundry

The Data Foundry system (https://computation.llnl.gov/casc/datafoundry) [3] uses a mediator based system for integration of Bioinformatics data. The system contains a modeling language to represent metadata which is used to generate a specific mediator for an information integration task.

GEON

In the domain of geological sciences, the Geosciences Network (http://www.geongrid.org/) [1] uses all information resources including maps and spatial information to be explicitly registered to an OWL (Web Ontology Language – see http://www.w3.org/TR/owl-features/) ontology. The ontology-registered data sources are then queried using an SQL called SOQL.

Cross-references

► Ontologies and Life Science Data Management
► Scientific Workflows

Recommended Reading

1. Bowers S., Lin K., and Ludaescher B. On integrating scientific resources through semantic registration. In Proc. 16th Int. Conf. on Scientific and Statistical Database Management, 2004, p. 349.
2. Cluet S., Delobel C., Siméon J., and Smaga K. Your mediators need data conversion! In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1998, pp. 177–188.
3. Critchlow T., Fidelis K., Ganesh M., Musick R., and Slezak T. Datafoundry: information management for scientific data. IEEE Trans. Inf. Technol. Biomed., 4(1):52–57, 2000.
4. Gupta A., Ludäscher B., Martone M.E., Rajasekar A., Ross E., Qian X., Santini S., He H., Zaslavsky I. BIRN-M: a semantic mediator for solving real-world neuroscience problems. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 2003, p. 678.
5. Hakimpour F. and Geppert A. Resolving semantic heterogeneity in schema integration. In Proc. Int. Conf. on Formal Ontology in Information System, 2001, pp. 297–308.
6. Halevy A. Why your data won't mix. ACM Queue, 3(8):50–58, 2003.
7. Ludäscher B., Gupta A., and Martone M.E. Model-based mediation with domain maps. In Proc. 17th Int. Conf. on Data Engineering, 2001, pp. 81–90.
8. Qi Y., Candan K.S., Sapino M.L., and Kintigh K.W. Integrating and querying taxonomies with quest in the presence of conflicts. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 2007, pp. 1153–1155.
9. Searls D.B. Data integration: challenges for drug discovery. Nature Rev. Drug Discov., 4:45–58, 2005.

Information Lifecycle Management

Hiroshi Yoshida
Fujitsu Limited, Yokohama, Japan

Synonyms
ILM

Definition
Information lifecycle management is the activity of storing data in the most appropriate storage infrastructure, according to the business value of that data as it changes over time. The Storage Networking Industry Association (SNIA) defines Information Lifecycle management as follows:

The policies, processes, practices, services and tools used to align the business value of information with the most appropriate and cost-effective infrastructure from the time when information is created through its final disposition. Information is aligned with business requirements through management policies and service levels associated with applications, metadata and data.

Key Points
The business value of information varies during its lifecycle, e.g., from the time when information is created through to its final disposition. Considering the varying business value of information, it is necessary to provide cost-effective storage infrastructure which meets the requirements corresponding to the business value. For example, information related to current transactions is most valuable from the viewpoint of current business operations and is usually stored in online disk arrays with high performance and high availability. Information on transactions during the past year has value as statistics and is stored in less expensive hard disk drives. Finally, information on transactions over several years has value only in meeting regulatory compliance and is stored in offline tape libraries.

Information lifecycle management is the generalization of such practices. In a typical example of ILM process, information is classified based on business values first, referring to its metadata or content. Then appropriate service level requirements are defined for information based on its classification. On the other hand, storage infrastructure resources are also classified based on service level attributes such as performance and availability. Finally the most cost-effective storage infrastructure resources, which meet the classified information requirements, are chosen. When information value changes, the information is migrated between multiple storage infrastructure classes according to predefined policies.

A simple example of information lifecycle management is hierarchical storage management (HSM). In HSM, files are migrated between multiple storage tiers such as high performance disk arrays, low performance hard disks, and tape libraries based on access frequency or latest access date. Instead of such simplistic policies, supporting policies which reflect the change of business value of information is the key to implementing ILM.

Cross-references
- ▶ DAS
- ▶ SAN
- ▶ SRM
- ▶ Storage Consolidation
- ▶ Storage Network Architectures
- ▶ Storage Networking Industry Association
- ▶ Storage Protocols
- ▶ Storage Virtualization

Recommended Reading
1. Storage Network Industry Association. Information Lifecycle Management Initiative, 2007. Available at: http://www.snia.org/forums/dmf/programs/ilmi
2. Storage Network Industry Association. Storage Network Industry Association tutorials, 2007. Available at: http://www.snia.org/education/tutorials/

Information Loss Measures

Josep Domingo-Ferrer
Universitat Rovira i Virgili, Tarragona, Catalonia

Synonyms
Data utility measures

Definition
Defining what a generic information loss measure is can be a tricky issue. Roughly speaking, it should capture the amount of information loss for a reasonable range of data uses. It will be said that there is little

information loss if the protected dataset is analytically valid and interesting according to the following definitions by Winkler [4]:

A protected microdata set is *analytically valid* if it approximately preserves the following with respect to the original data (some conditions apply only to continuous attributes):

1. Means and covariances on a small set of subdomains (subsets of records and/or attributes)
2. Marginal values for a few tabulations of the data
3. At least one distributional characteristic

A microdata set is *analytically interesting* if six attributes on important subdomains are provided that can be validly analyzed.

More precise conditions of analytical validity and analytical interest cannot be stated without taking specific data uses into account. As imprecise as they may be, the above definitions suggest some possible measures:

- Compare raw records in the original and the protected dataset. The more similar the SDC method to the identity function, the less the impact (but the higher the disclosure risk!). This requires pairing records in the original dataset and records in the protected dataset. For masking methods, each record in the protected dataset is naturally paired to the record in the original dataset it originates from. For synthetic protected datasets, pairing is less obvious.
- Compare some statistics computed on the original and the protected datasets. The above definitions list some statistics which should be preserved as much as possible by an SDC method.

Key Points

A strict evaluation of information loss must be based on the data uses to be supported by the protected data. The greater the differences between the results obtained on original and protected data for those uses, the higher the loss of information. However, very often microdata protection cannot be performed in a data use specific manner, for the following reasons:

- Potential data uses are very diverse and it may be even hard to identify them all at the moment of data release by the data protector.
- Even if all data uses could be identified, releasing several versions of the same original dataset so that the *i*-th version has an information loss optimized for the *i*-th data use may result in unexpected disclosure.

Since datasets must often be protected with no specific data use in mind, generic information loss measures are desirable to guide the data protector in assessing how much harm is being inflicted to the data by a particular SDC technique.

Information loss measures for numerical data. Assume a microdata set with n individuals (records) I_1, I_2,...,I_n and p continuous attributes Z_1, Z_2,...,Z_p. Let X be the matrix representing the original microdata set (rows are records and columns are attributes). Let X' be the matrix representing the protected microdata set. The following tools are useful to characterize the information contained in the dataset:

- Covariance matrices V (on X) and V' (on X').
- Correlation matrices R and R'.
- Correlation matrices RF and RF' between the p attributes and the p factors PC_1,...,PC_p obtained through principal components analysis.
- Communality between each of the p attributes and the first principal component PC_1 (or other principal components PCi's). Communality is the percentage of each attribute that is explained by PC_1 (or PC_i). Let C be the vector of communalities for X and C' the corresponding vector for X'.
- Factor score coefficient matrices F and F'. Matrix F contains the factors that should multiply each attribute in X to obtain its projection on each principal component. F' is the corresponding matrix for X'.

There does not seem to be a single quantitative measure which completely reflects those structural differences. Therefore, it was proposed in Domingo-Ferrer, Mateo-Sanz and Torra [1] and Domingo-Ferrer and Torra [2] to measure information loss through the discrepancies between matrices X, V, R, RF, C and F obtained on the original data and the corresponding X', V', R', RF', C' and F' obtained on the protected dataset. In particular, discrepancy between correlations is related to the information loss for data uses such as regressions and cross tabulations.

Matrix discrepancy can be measured in at least three ways:

- *Mean square error.* Sum of squared component wise differences between pairs of matrices, divided by the number of cells in either matrix.
- *Mean absolute error.* Sum of absolute component wise differences between pairs of matrices, divided by the number of cells in either matrix.

- *Mean variation.* Sum of absolute percent variation of components in the matrix computed on protected data with respect to components in the matrix computed on original data, divided by the number of cells in either matrix. This approach has the advantage of not being affected by scale changes of attributes.

For alternative [0,1]-bounded information loss measures for numerical data, see Mateo-Sanz et al. [3].

Information loss measures for categorical data. These can be based on direct comparison of categorical values, comparison of contingency tables, on Shannon's entropy. See Domingo-Ferrer and Torra [2] for more details.

Cross-references

- ► Disclosure Risk
- ► Inference Control in Statistical Databases
- ► Microdata
- ► SDC Score

Recommended Reading

1. Domingo-Ferrer J., Mateo-Sanz J.M., and Torra V. Comparing SDC methods for microdata on the basis of information loss and disclosure risk. In Pre-proceedings of ETK-NTTS'2001, vol. 2, 2001, pp. 807–826.
2. Domingo-Ferrer J. and Torra V. Disclosure protection methods and information loss for microdata. In Confidentiality, Disclosure and Data Access: Theory and Practical Applications for Statistical Agencies, P. Doyle J.I. Lane J.J.M. Theeuwes L. Zayatz (eds.). Elsevier, Amsterdam, 2001, pp. 91–110.
3. Mateo-Sanz J.M., Domingo-Ferrer J. and Sebé F. Probabilistic information loss measures in confidentiality protection of continuous microdata. Data Mining Knowl. Discov., 11(2):181–193, 2005.
4. Winkler W.E. Re-identification methods for evaluating the confidentiality of analytically valid microdata. Res. Off. Stat., 1(2):50–69, 1998.

Information Navigation

Thomas Rist
University of Applied Sciences, Augsburg, Germany

Synonyms

Information foraging; Information browsing; Interactive information exploration

Definition

The word "navigate" has its roots in the two Latin words "navis" (ship) and "agere" (to "move"/"direct") and has been used for centuries in the domain of nautics for activities necessary to find one's way, and to control the movement of a vessel while traveling from one location to another. Different scientific discoveries, technical inventions, and cultural backgrounds, have brought about a wide variety of navigation techniques and navigation devices, all of them have in common that they assume a certain structuring of the underlying geographical space. With the advent of electronically stored volumes of information, and electronically networked information sources in particular, the concept of geographical navigation has been generalized and adopted as a metaphor for accessing chunks of digitalized information in a goal-directed way. In this metaphorical view perceivable presentations of meaningful information chunks of an overall information space with form the counterparts of physical locations in a geographical space, and information navigation capitalizes on the fact that structural relations between physical locations, such as neighborhood, proximity, distance, connectedness, reachability, or crossway can be mapped onto meaningful relations between information chunks as well. Depending on nature of an application, a user is confronted with different navigation tasks. A typical task in a database application is to find the shortest access path from an entry point to a certain data record, in an e-Learning system, the navigation task may manifest itself in accessing the learning units in a didactically meaningful sequential order, while in an online bookstore, a user may navigate through the assortment of books following the traces of other customers with a similar interest profile. In any case, goal-directedness and exploration of structure distinguishes navigation activities from arbitrary, disoriented movements, be it in a geographical space, an information space or a combination of both in which an information space is superimposed on a geographical space.

Key Points

With the increasing amount of accessible electronic data, information navigation has become a central issue in many application domains and a challenge for architects and developers of user interfaces as aids to assist users in accomplishing various navigation tasks. Navigation tasks are often described relative to an navigation space that is characterized by the set of

potentially accessible information units, and the structural relationships that hold between these units. The structure of an navigation space determines in which order information units can be accessed and thus the effort it takes to navigate from an entry point to a particular unit. The spectrum of commonly used principles includes linear, hierarchical/tree-like, radial, and graph-like structuring of navigation spaces. The structuring of a navigation space should, however, comply with a reasonable, semantically-motivated structuring of the information units of the application domain.

An effective navigation aide supports a user in at least two basic navigation tasks: *orientation* – "where am I," and *path continuation* – "where to go from here to get (closer) to the target?" Techniques to support keeping orientation include: sitemaps and distorted views – such as "*fisheye views*" – both designed to show a user's location in a broader context, or navigation histories, such as "*bread-crump navigation*" as today used by many web-sites to show a user her trail from a chosen entry-point to her current location. Supporting users in the decision making of how to continue a navigation path from a current location/information unit comprises the disclosure of available and relevant options, as well as de-emphasizing or even hiding of options that should not be taken with regard of achieving the underlying navigation objective.

Among the supporting techniques are techniques that aim to provide environmental cues about where to find further interesting information – the "*information scent*" approach is an exemplar of such a technique. Also, the concept of "adaptive menus" falls into this category of techniques. It has been introduced with the aim to facilitate a user's navigation through a menu-based interface by hiding options which are regarded less relevant in a certain context. In practice, however, it is often too hard a task for a computer system to make appropriate assumptions about the user's current tasks and aims and therefore inadequate suggestions for the continuation of a navigation may be given. Also, accessibility, comprehensibility and effectiveness of navigation structures of web-sites have become major criteria in most web-usability evaluations.

Earlier work on information navigation has mainly focused on graphical user interfaces (GUIs) and hypermedia-style information access with view-based navigation aids, assuming a user navigates through an information space by traversing interlinked visual presentations. Research includes the development of visual navigation aids as well as the understanding of the cognitive processes involved in conducting navigation tasks (e.g., [1]).

More recently, navigation issues are being researched in other areas too, including voice-dialogue systems, virtual reality applications, and computer games. In the emerging fields of ubiquitous and mobile computing the mutual interplay of geo-spatial navigation with information navigation provides interesting potential for new services that benefit mobile users. The advent of large-scale internet shops and forums has shed light on yet another aspect of navigation – social navigation, a term that has been coined for a model of navigation support that capitalizes on observed navigation patterns of others [2].

Cross-references

▶ Hypermedia
▶ Visual Data Exploration

Recommended Reading

1. Furnas G.W. Effective view navigation. In Proc. SIGCHI Conf. on Human Factors in Computing Systems., 1997, pp. 367–374.
2. Höök K., Munro A., and Benyon D. Designing Information Spaces: The Social Navigation Approach. Springer, Berlin, 2002.

Information Quality: Managing Information as a Product

Diane M. Strong
Worcester Polytechnic Institute, Worcester, MA, USA

Definition

Traditionally, information has been viewed as a by-product of a computer system or an event. From this viewpoint, the focus is on designing and delivering computer systems, rather than designing and delivering information. To increase the quality of information available to information consumers, organizations need to treat information as a product being intentionally produced for those who will use that information. This means actively managing information and its quality. Such an information product (IP) approach focuses attention on information quality, i.e., delivering high-quality information that is fit for use by information consumers, rather than solely on data

quality, i.e., maintaining the quality of the data stored in databases or data warehouses. While the terms "data" and "information" are often used interchangeably in the information and data quality literature, the term information quality is more often used in studies that take an IP approach and explicitly acknowledge the needs of information consumers, whereas data quality is more often used when the quality of data is assessed separately from the needs of users.

Historical Background

While there are some early seminal papers, information quality (IQ) only emerged as a recognized field of information technology (IT) research in the mid-1990s. For example, the first International Conference on Information Quality was held in 1996 at MIT [8]. In 1998, Wang et al. [14] published "Manage your Information as a Product", which argued for the concept of treating information as a product that is intentionally managed and produced for consumers who have expectations about its quality. This concept provided researchers with a useful lens for framing their information quality research ideas and led to many studies developing methods, measures, techniques, and recommendations for managing information as a product. Together, these methods, measures, techniques, and recommendations for managing information as a product are referred to as the information product (IP) approach.

Foundations

Managing information as a product, i.e., taking an information product (IP) approach, means actively managing organizational information in order to deliver high-quality information that is fit for use by information consumers [14]. The IP approach involves four principles for managing information as a product, with four corresponding areas of information quality research. These four areas of research, which are described below, are (i) the needs of information consumers, (ii) processes for producing high-quality information products, (iii) the life cycle of information products, and (iv) organizational governance structures and recommendations for actively managing information as a product.

The Needs of Information Consumers

Like other products, the quality of information is judged by those who purchase and use information products, referred to as information consumers. Thus, information is of high-quality to the extent that it is fit for use by information consumers. This means that information quality includes both objective dimensions (those dimensions that can be defined independently of a user or a task), such as accuracy, and subjective or contextual dimensions (those dimensions that are defined relative to the context of a user performing some task that requires that information), such as timeliness. For example, whether information is sufficiently timely depends on the task being performed, e.g., stock prices for a historical analysis or for day trading. Even those dimensions that can be objectively defined may still have a subjective or contextual aspect, e.g., whether the information is sufficiently accurate for the task versus whether it is completely accurate.

Research on the needs of information consumers has used marketing research techniques to understand the dimensions or characteristics of information products that consumers assess when considering an IP for their use. The result is sixteen dimensions that information consumers consider when assessing the quality of information, which are grouped into four categories, see Table 1 [15]. The intrinsic dimensions are those that capture the quality of the information itself. The contextual dimensions are those that capture quality in the context of the task and the users. The accessibility dimensions capture the quality of the process of obtaining and manipulating the information. The representational dimensions capture quality related to how the information is represented

Information Quality: Managing Information as a Product. Table 1. Information quality categories and dimensions

IQ category	IQ dimensions
Intrinsic IQ	Free-of-error, Objectivity, Believability, Reputation
Accessibility IQ	Accessibility, Ease of Manipulation, Security
Contextual IQ	Relevancy, Value-Added, Timeliness, Completeness, Amount of information
Representational IQ	Interpretability, Ease of understanding, Concise representation, Consistent representation

to users. These dimensions have been verified in organizational contexts [11,12] and by numerous studies that have used these dimensions or some subset of them in IQ studies.

The Information Quality Assessment (IQA) questionnaire measures information quality along these sixteen IQ dimensions [6]. Each dimension is represented by several questions for which questionnaire respondents rate the quality of their information on a scale of zero (not at all) to ten (completely). For example, two questions for the free-of-error dimension are "This information is correct" and "This information is accurate." This questionnaire has demonstrated reliability and validity and has been used in research studies and in organizations seeking to improve their IQ. One interesting result from baseline IQ measures at several companies is that IT professionals responsible for storing and maintaining the information consistently assess information quality higher than do the information consumers who use that information.

While the IQA provides a comprehensive IQ assessment instrument, it is not the only method for assessing IQ [7]. For some dimensions, it is possible to develop objective measures that do not require data and time from information consumers for assessment. The typical approach is to define a metric on a 0–1 scale such as the following metric for timeliness [1]:

$$\text{Timeliness} = \left\{ \text{Max}\left[\left(\frac{1 - \text{currency}}{\text{shelf-life}} \right), 0 \right] \right\}^{s}$$

The parameter s adjusts the sensitivity of the timeliness measure to the ratio of the currency of the information relative to the shelf-life or volatility of that information. When such metrics can be defined, IQ can be assessed independently of users.

The Product and Service Performance Model for Information Quality (PSP/IQ model) groups the sixteen IQ dimensions into four IQ quadrants, sound, dependable, useful, and usable, as shown in Table 2 [3]. These quadrants are derived from the total quality management literature. Specifically, on one dimension, the PSP/IQ model captures aspects of IQ that are related to product quality and those related to service quality. On the other dimension, it distinguishes between conforming to specifications and meeting or exceeding information consumer expectations. These quadrants serve as a better foundation for analyzing and improving quality than the original four categories of IQ in Table 1. Several studies have used the PSP/IQ model, in combination with the IQA instrument, to consolidate the 16 dimensional measures into four overall measures, e.g., [6].

The PSP/IQ model and the IQA instrument are useful tools for benchmarking and analyzing IQ. The resulting measures of IQ provide the foundation for gap analyses [6]. One gap is the difference between an

Information Quality: Managing Information as a Product. Table 2. The PSP/IQ model

	Conforms to specifications	Meets or exceeds consumer expectations
Product quality	*Sound Information*	*Useful Information*
	Dimensions of Sound Information:	Dimensions of Useful Information:
	• Free-of-Error	• Appropriate Amount
	• Concise Representation	• Relevancy
	• Completeness	• Understandability
	• Consistent Representation	• Interpretability
		• Objectivity
Service quality	*Dependable Information*	*Usable Information*
	Dimensions of Dependable Information:	Dimensions of Usable Information:
	• Timeliness	• Believability
	• Security	• Accessibility
		• Ease of Manipulation
		• Reputation
		• Value-added

organization and a benchmark organization, which indicates the possibilities for improvement in that organization. Another gap is the difference between the assessments of IT professionals responsible for the information and the users of that information. When IT professionals rate IQ much higher, there are opportunities for better communication about IQ between information consumers and IT professionals.

The needs of information consumers, the first area of research in the IP approach to IQ, address the principle that the needs of information consumers are the primary purpose of improving information quality. Research in this first area addresses the information product itself and its quality as assessed by those who use the product.

Processes for Producing High-Quality Information

Like physical products, information products are produced by production/manufacturing processes. Like manufacturing processes for physical products, information manufacturing processes will only produce high-quality information if they are designed to do so. Thus, research on processes for producing high-quality information products, the second area of IP research, draws on parallels with manufacturing processes for physical products.

This area of research introduces two IQ roles beyond the information consumer described in the last section. Information collectors (also called information suppliers or providers) collect and provide the raw data for the information manufacturing process. The second role consists of those who perform the tasks of manufacturing the IP, which has been referred to as information custodians (also called information manufacturers or IT professionals).

One useful parallel with manufacturing is to apply the Total Quality Management (TQM) recommendations from quality gurus such as Deming, Crosby, and Juran that have resulted in major improvements in the quality of the products produced from manufacturing processes. For information processes, an IQ improvement cycle, similar to Deming's cycle, has been proposed. Called the Total Data Quality Management (TDQM) cycle, it involves the continuous and repeated application of four steps, define, measure, analyze, and improve IQ [13]. The previous section discussed research results that *defined* the dimensions of IQ, developed *measures* for them, and *analyzed* the gaps between an organization's IQ and benchmark IQ measures.

To *improve* the quality of information products, the processes that produce those information products must be defined, measured, analyzed, and improved. Research efforts in this area have focused on modeling information manufacturing processes in a way that permits analysis of process design alternatives and measurement of the IQ of the information products produced. One such research effort is the Information Manufacturing System (IMS), which is a method for developing a network model of information flow [1]. The model consists of five types of components that capture the steps in an information manufacturing system. These five are (i) the data vendor block represents a source of data, (ii) the processing block represents a step that adds value to the information, (iii) the data storage block represents a database, (iv) the quality block represents a step that enhances the quality of the information for information consumers, and (v) the customer block represents the delivery of an information product to an information consumer. At each step in the network of components, various dimensions of IQ are computed given the quality of data from the data vendor block and the quality effects of each of the processing and quality blocks.

Modeling information manufacturing processes in a way that captures the details needed to compute IQ metrics and to analyze process design alternatives is an active area of current research. One extension to the IMS modeling approach adds new modeling components, e.g., an information system boundary block and an organizational boundary block which capture handoff points at which the quality of information may be affected [10]. It also elaborates, and more formally defines, the data needed to specify each block so that IQ metrics can be computed. A variety of other extensions are also under development, e.g., [9], including software tools that assist with the modeling and metric computation tasks.

The Life Cycle of Information Products

The third principle of managing information as a product is to explicitly manage the life cycle of the IP, similar to how the life cycle of a physical product is managed. For physical products, the product is continually improved and extended, so that the product gradually becomes more useful to customers, more reliable, and less costly to produce, which in turn extends the life of the product. Few organizations treat their information as though it had a life cycle.

Instead they view information as a by-product of a computer system or event, and assume the life cycle of the information is the same as the life-cycle of that computer system or event. Information typically is upgraded or its characteristics changed as information consumers make requests, rather than the proactive management of upgrades typical for physical products.

While IQ researchers and practitioners acknowledge the need for managing an IP's life-cycle, there is only a little research addressing this issue. For example, there is some research addressing the optimal or recommended frequency for updating data. The lack of research on managing the life cycle of an IP may be related to the state of research on the governance structures needed for actively managing information as a product.

Governance Structures for Managing Information as a Product

The fourth area of research for managing information as a product focuses on the appropriate management and governance structures needed to support the IP approach. Wang et al.'s paper on managing information as a product [14] recommended appointing an Information Product Manager (IPM), a position similar to the Brand Manager role that organizations employ to manage products across the multiple functional areas of product development, manufacturing, marketing, and service that must all be coordinated to deliver high-quality products and manage their life cycle. Similarly an IPM would manage one or more IPs across information suppliers, information manufacturers, and information consumers. Thus, this is introducing a fourth information role, that of an information product manager. This role differs from that of a CIO, who typically is only responsible for the information manufacturing role and does not have responsibilities for information suppliers or information consumers. The CIO also must typically focus more on the hardware and software architecture and infrastructure than on the architecture and infrastructure of information. Thus, there is a need for a role that provides the cross-functional management structure needed to deliver high-quality information to information consumers.

The IPM role, however, is only one piece of the broader research area of information governance, management structures and best practices needed to adequately provide an IP approach in organizations. Such best practices also need to include the financial side of the IP approach [4], e.g., cost/benefit analysis of IQ improvements, the value and pricing of information, and the auditing of information manufacturing processes. There are still many opportunities for research and the exploration of best practices in the area of IQ governance.

Key Applications

Organizations are at varying stages of attempting to apply an IP approach to managing their information. Examples from several organizations including both for-profit and not-for-profit organizations illustrate the value of the four principles of the IP approach.

Principle 1: Understand the IP Needs of Information Consumers

As an investment bank examined the needs of its information consumers, it realized that it had both external customers who needed information as well as internal information consumers. For its external customers, it was providing adequate information for single requests, but was failing to consider that these customers interacted with multiple divisions, and thus it was failing to provide coordinated and aggregate information. For internal information consumers, it was treating their information needs as by-products of its events with external customers, and largely failed to consider or meet their needs.

A retail chain that sells eyewear had not examined the information needs of its grinders and thus was not treating the information sent from its opticians in retail outlets to the centralized lens grinders as an information product. Its investigation into the information needs of grinders revealed that many of its external customers' complaints about the quality of their eye products were not due to poor quality grinding but to the quality of the information products sent from the opticians to the grinders. When the opticians understood the grinders' needs as an information product, many of the problems with the quality of the eye products were eliminated.

A company that resells data to retailers worked to understand the information needs of its customers and then developed internal information production processes that could tailor its IPs to individual customers. By its attention to its internal and external information customers, it was able to deliver superior information products. A hospital tailored the IQA instrument to its

needs and used it to better understand when it was meeting or failing to meet its information consumers' needs.

Principle 2: Manage Information as a Product of a Well-Defined Production Process

To provide information about customers' accounts across their activities with several divisions, the investment bank maintains a centralized customer account database. The production process for this information, however, was flawed, so that the IP's produced did not meet the needs of information consumers. As a result, information consumers in the various divisions created their own customer account databases with information production processes tailored to their needs. While these individual databases met local needs, the global needs for account information across divisions were not being met. The investment bank redesigned its information production processes to produce higher quality information and information tailored to the divisions, so that local needs were better met using information from the global customer account database.

The hospital that used the IQA instrument has also attended to improving its information production processes. It has used a modified version of the IMS modeling technique to model its information processes [2].

Principle 3: Manage Information as a Product with a Life Cycle

A petrochemical company produced a material safety data sheet (MSDS), an IP, for each of its chemicals. While the process for producing the initial MSDS was excellent, it was not managed as an IP with its own life cycle, but rather was updated as a by-product of making changes to the chemical. New hazards, however, were discovered as the chemical was used, and the company was exposed to legal risks when it did not issue a revised MSDS for newly discovered hazards. Because it treated the MSDS as a by-product of chemical development, it did not have an information production process for tracking new hazards and revising the MSDS. To address this problem, it started to treat the MSDS as an IP with its own life cycle, production process, and governance structure. The investment bank was required to maintain risk profiles for its customers. Like the petrochemical company, it did not treat these risk profiles as an IP, and thus did not have a well-defined process for updating them, exposing it to legal risks.

Principle 4: Develop a Governance Structure to Manage Information Production Processes and the Resulting Information Products

Although it is nearly 10 years since the publication of "Manage Your Information as a Product" [14], organizations are still struggling with finding governance structures appropriate for ensuring that information consumers receive information that is fit for their use. Individuals have been assigned roles similar to an IPM, but comprehensive governance structures are not yet common. For example, the petrochemical company appointed someone to develop and manage a process for keeping each MSDS up-to-date with emerging information about hazards. At the company that resells data, the CIO's role was changed to focus more on information processes and products. The hospital mentioned in principles 1 and 2 not only uses the IQA instrument and develops models of its information production processes, but also has an IQ group whose responsibility is to monitor IQ and develop solutions to IQ problems including developing governance structures. At another hospital, there is an information quality analyst role whose responsibilities include uncovering and investigating IQ problems.

Future Directions

The four research areas and principles of the IP approach were presented in approximate order of research maturity. The needs of information consumers in terms of the dimensions of information quality are well-defined; reliable measures of these dimensions have been developed; and methods for analyzing and improving information quality along these dimensions are available. These dimensions continue to be used in research studies and organizational practices. They provide a complete set from which most information quality research studies and practical IQ improvement projects select a subset that is most relevant for the particular IQ effort being undertaken.

Methods for modeling and improving information manufacturing processes are currently an active area of research. These studies borrow heavily from the manufacturing and TQM literature, but also tailor the application of manufacturing research results to address the different characteristics of information as a product. While this is an active area of current research, there is still much to be done to provide methods and techniques that can easily be used in organizations to improve the quality of IP as delivered

to information consumers by improving the information production processes producing those IP.

Both the IP life cycle and the governance of IQ are areas with limited current research, but much potential for research. While organizations know how to manage the development and improvement of physical products, the same is not yet true for information products.

More research is also needed that integrates across two or more of these four research areas. For example, more research is needed that integrates the well-defined IQ measures of the IQ dimensions into studies of information production process improvements, e.g., see [5] as an early example of integrating process measures with the IQ outcome measures. Similarly, IP life cycles should be integrated into information production process models. IQ governance studies are needed that includes regular outcome measures to be used to direct efforts at process analysis and improvement. Governance studies are also needed that will develop methods for costing and justifying investments in IQ improvements. The purpose of these research efforts is to develop theories, methods, tools, and recommendations that will help organizations actively manage their information and thus ensure the delivery of high quality information to information consumers.

Cross-references

- ▶ Data Quality
- ▶ Information Quality
- ▶ Information Quality Assessment
- ▶ Information Quality Decision and Making
- ▶ Information Quality Policy and Strategy
- ▶ Information Quality Problem Solving

Recommended Reading

1. Ballou D.P., Wang R.Y., Pazer H., and Tayi G.K. Modeling information manufacturing systems to determine information product quality. Manage. Sci., 44(4):462–484, 1998.
2. Davidson B., Lee Y.W., and Wang R.Y. Developing data production maps: meeting patient discharge data submission requirements. Int. J. Healthc. Technol. Manag., 6(2):223–240, 2004.
3. Kahn B.K., Strong D.M., and Wang R.Y. Information quality benchmarks: product and service performance. Commun. ACM, 45(4ve):184–192, 2002.
4. Lee Y.W., Pipino L., Funk J., and Wang R.Y. Journey to information quality. MIT Press, Cambridge, MA, 2006.
5. Lee Y.W. and Strong D.M. Knowing-why about data processes and data quality. J. Manage. Inf. Syst., 20(3):13–39, 2004.
6. Lee Y.W., Strong D.M., Kahn B.K., and Wang R.Y. AIMQ: a methodology for information quality assessment. Inf. Manage., 40(2):133–146, 2002.
7. Pipino L.L., Lee Y.W., and Wang R.Y. Data quality assessment. Commun. ACM, 45(4ve):38–46, 2002.
8. Proceedings of the International Conference on Information Quality (ICIQ). (1996 and yearly since then), available at: http://mitiq.mit.edu and at http://mitiq.mit.edu/ICIQ.
9. Scannapieco M., Pernici B., and Pierce E. IP-UML: towards a methodology for quality improvement based on the ip-map framework. In Proc. Int. Conf. on Information Systems, 2002, pp. 279–291.
10. Shankaranarayan G., Ziad M., and Wang R.Y. Managing data quality in dynamic decision environment: an information product approach. J. Database Manage., 14(4):14–32, 2003.
11. Strong D.M., Lee Y.W., and Wang R.Y. Data quality in context. Commun. ACM, 40(5):103–110, 1997.
12. Strong D.M., Lee Y.W., and Wang R.Y. Ten potholes in the road to information quality. IEEE Comput., 30(8):38–46, 1997.
13. Wang R.Y. A product perspective on total data quality management. Commun. ACM, 41(2):58–65, 1998.
14. Wang R.Y., Lee Y.W., Pipino L.L., and Strong D.M. Manage your information as a product. Sloan Manage. Rev., 39(4): 95–105, 1998.
15. Wang R.Y. and Strong D.M. Beyond accuracy: what data quality means to data consumers. J. Manage. Inf. Syst., 12(4):5–34.

Information Quality and Decision Making

InduShobha N. Chengalur-Smith
University at Albany – SUNY, Albany, NY, USA

Synonyms

Data Quality

Definition

Decision-making is an inexact science, and one of the reasons is the utilization of data of imperfect quality in the decision process. If the user is aware of the caliber of the data and incorporates this information in the decision process, the effectiveness of the process is expected to increase.

Historical Background

A decision can be thought of as a set of actions related to and including the choice of one alternative rather than another. A rational model of decision-making is one in which decision-makers consider all aspects of all alternatives before making a decision. However actual decision-making often falls short of this ideal model because knowledge and experience of the consequence

is incomplete, there is limited amount of time to explore all alternatives, and humans do not calculate perfectly. Hansson [8] stated that all decisions are made under uncertainty and he presented four components of great uncertainty: uncertainty of options, uncertainty of the consequences of the options, uncertainty of reliability of information and uncertainty of values. The third category, uncertainty of reliability of information, is the focus of this entry.

The decision process is sensitive to task complexity, time pressure and other contextual factors. Task complexity itself is a function of several variables such as the number of alternatives, the number of attributes, and time pressure. If humans have a processing limit of 7 ± 2 items, as believed, the decision process quickly runs into information overload, even without the addition of data quality information. However, increases in decision complexity are often compensated for by a selective use of information or by shifting to more simplistic strategies. Exactly how this plays out, given the additional data quality information that needs to be evaluated, has not yet been conclusively determined. Thus, there could be plausible arguments for and against the use of data quality information in decision making.

Payne et al. [12] proposed that people adopt a decision strategy on the basis of a cost benefit framework, i.e., individuals compromise between making the best possible decision and minimizing their cognitive effort in making that decision. Decision making strategies could be (1) alternative-based or attribute-based, (2) compensatory or non-compensatory, and (3) consistent or selective. Consider, for instance, a satisficing strategy, where alternatives are considered one at a time, in the order in which they are presented. Each attribute is compared to a pre-determined cut-off level and the first alternative whose attributes all meet the cutoffs is selected and the process is stopped. Such a strategy is alternative-based (multiple attributes about an alternative are considered before the next alternative is processed), non-compensatory (no trade-offs are made among alternatives), selective (all the alternatives are not evaluated using the same amount of information). On the other hand, consider a "weighted additive" strategy that assigns relative importances (or weights) to each attribute and evaluates every alternative by multiplying the value of the attribute by its weight and summing up these weighted values for all attributes. Although it is another alternative based strategy, this is a compensatory strategy because tradeoffs are made among alternatives. It is also a consistent strategy because all the alternatives are evaluated using the same amount of information and the alternative with the highest overall evaluation is chosen.

It has been hypothesized that providing information about the quality of the data attributes used by the decision maker could be beneficial. It could also be argued that such data simply creates an additional burden, both on the data provider and the decision maker, without significant payback in improved decision making. The current research in this arena is focused on trying to determine the conditions under which providing data quality information may be useful to decision makers.

Foundations

Data quality is widely considered to be multi dimensional. Wang and Strong [15] identified 15 dimensions of data quality that they factored into 4 distinct groups: intrinsic, contextual, usability and accessibility and determined that accuracy (used interchangeably with reliability) was the most important attribute. It is expected that not only the type of data quality attribute provided but also the format in which the data quality information is recorded and presented would play a role in the decision process.

Data regarding the quality of stored data can be thought of as one type of metadata. In fact, some database systems already do this indirectly through the inclusion of a date field. The record of the time when the data was collected reflects one component of the quality of the data, namely currency. It is also possible to have information regarding data quality (e.g., accuracy) at the level of the individual data item. In certain situations this is warranted and can be achieved via the use of data tags [14]. Although this would be complex and costly, it may be necessary if the processes generating the data items were erratic and highly volatile. At the other extreme one could have data regarding the quality of an entire file or relational table. This approach would be most appropriate where the entire file was generated by a single, rather stable, process. An intermediate approach would be to record data quality information at the level of each data field. This approach is most appropriate where the data being utilized originated from a number of processes which may have substantially different data quality capabilities. Such an approach is neither as storage-intensive as item-level data nor as generic as file-level data. If it is assumed that the quality of data items in a

particular domain is the same, although different domains may have different quality, creating such metadata may not be very labor intensive. Although at this time typical databases do not include the kind of data quality information envisioned, since data about data quality are metadata, it would be logical to include such information in the data dictionary.

Even though the inclusion of data quality information has become technologically feasible, the collection of such information is still a challenging undertaking. Quality characteristics of data may be declared through qualitative means such as a "data source" field or through a quantitative system such as a reliability rating. Clearly, providing data quality information on a two-point ordinal scale is more manageable than providing it on a 100-point continuum. But if the latter option leads to more effective decision-making, then the considerable effort required may be worth it. Thus the availability of information is not enough. It needs to be presented in a form that will promote effective decision making. The type of information regarding the data's quality that would be most helpful to users may depend on the sophistication of the users, but is more likely to be a function of the decision process or strategy.

Empirical evidence on the effective use of information during the decision making process has been mixed. Ben Zur and Breznitz [1] found that people under time pressure place more emphasis on negative information about alternatives. The way information is displayed (i.e., on a verbal or numerical scale) has also been found to affect decision behavior [13]. Even within a numerical format, decision making is sensitive to different displays [9]. Thus both content and style of the metadata could have an impact on the decision process. In the public policy arena, Grether et al. [7] found that consumers ignore less relevant information in complex information environments. On the other hand, Gaeth and Shanteau [6] indicated that expert decisions were adversely affected by irrelevant factors, although this could be overcome by training. Several other studies, including [10], found that increasing the amount of attribute information about alternatives increases subjects' confidence in their decisions but also increases the variability of responses. Anecdotal evidence showed that higher information loads lead to simpler processing strategies and hence more consistent decisions.

Having unequivocal measures of the effectiveness of the decision making process is imperative because of the difficulty of identifying what constitutes a good decision. Ideally a decision aid (such as the inclusion of data quality information) should lead to more accurate results. When there are no correct answers, consistency is a desirable outcome of the decision process. MacGregor et al. [11] define two forms of consistency. One is the tendency to produce the same results when applied by the same user under identical circumstances, and the other is the tendency to produce similar results in the hands of different users. MacGregor et al. [11] found that the more structured the decision aid, the greater the improvement in accuracy and consistency of the resulting decision.

Chengalur-Smith et al. [4] expanded on this concept to measure the impact of providing data quality information on decision processes and defined three key measures of the outcome of the decision:

1. Decision Complacency: Complacency refers to the degree to which data quality information is ignored. If the decision-maker does not change the originally preferred alternative after viewing data quality information, he/she has exhibited decision complacency. Complacency can be regarded as a measure of futility, for it implies that providing data quality information has not impacted the final decision. Obviously low levels of decision complacency are the most desirable outcome.
2. Decision Consensus: Consensus examines the degree to which decision-makers can converge on a decision in the presence of data quality information. Often a number of decision-makers will be involved in the same decision process. Consensus explores the impact of data quality information upon the ability of such groups to maintain the prior degree of agreement concerning the preferred course of action. It is a measure of group response to this new incremental information.
3. Decision Consistency: Consistency is designed to capture agreement across all alternatives, when data quality information is provided. Note that the previous two measures, complacency and consensus, focused on only the top-ranked or preferred alternative. In a sense, consistency is an extension of complacency to the entire set of alternatives. Once again a high value for this measure would indicate that major re-orderings in the ranks did not occur.

Note that complacency (not changing the originally preferred alternative in the presence of data quality

information) and consensus (ability to converge on an alternative using data quality) deal only with the issue of the top-ranked alternative. Although decision-makers would most often be interested in the top ranked alternative, there are numerous decision contexts where the interest is in the ranking of all alternatives (e.g., the allocation of merit raises across an entire department). In general, complacency and consensus should be considered hierarchically, i.e., consensus should only be evaluated after non-complacency is established.

Key Applications

The widespread use of data warehouses has highlighted the necessity of data hygiene. However, it is infeasible to thoroughly cleanse the data, particularly when the ways in which the data may be used is in flux. A viable alternative may be to tag the fields in the warehouse with data quality information. Some of the issues that need to be considered when designing such a warehouse have been discussed earlier. The results from preliminary experimental studies described below indicate that it would be beneficial to include data quality tags when the data warehouse is used by managers on an ad hoc basis.

Future Directions

Although some experimental research has established that data quality information is used by decision makers, more work is required to determine exactly how such information is used. This would help in identifying the best format in which data quality information should be presented to users. The results of such investigations would also provide guidelines for designers of data warehouses and those that implement warehouse applications.

Experimental Results

Several experiments have been conducted to provide insight into the type of data quality information that would be most effective to support decision processes. A preliminary question that needed to be addressed was the proposition that the type of data quality information that is appropriate may be a function of individual preferences. If individuals processed data quality information in substantially different ways, incorporating data quality information may be counter-productive. A study that used learning styles as a surrogate for individual differences in the mindsets of professionals in an organization found no significant differences that were attributable to learning styles using the three key measures outlined above [4]. This makes for simpler implementation since this implies that the data quality information for a database does not necessarily have to be tailored to individuals.

A series of controlled experiments [2–4] investigating the impact of providing data quality information to decision makers found that the impact depended on factors such as the complexity of the task, the decision strategy and the format in which the information was provided. Preliminary results suggested that when decision makers are confronted with clearly differentiated alternatives, the inclusion of data quality information impacted the selection of a preferred alternative while maintaining group consensus. A follow-up study examined these factors while also taking experience and time pressure into account [5]. The findings suggested that managers with little to no domain-specific experience were more likely to use the data quality information provided. In addition, the results indicated that those who felt time pressure during the decision making process were more likely to use the data quality information provided. Finally, older decision makers were more likely to use data quality information than younger ones. However the level of consensus among the decision makers declined, indicating that improvements gained in individual decision processes through the inclusion of data quality information may be at the expense of group consensus.

Cross-references

- ▶ Data Cleaning
- ▶ Data Quality Assessment
- ▶ Data Quality Dimensions
- ▶ Data Warehouse
- ▶ Data Warehouse Life-cycle and Design
- ▶ Data Warehouse MetaData
- ▶ Data Warehousing Systems: Foundations and Architectures
- ▶ Information Quality Assessment
- ▶ Information Quality Policy and Strategy
- ▶ Information Quality: Managing Information as a Product
- ▶ Meta Data
- ▶ Metadata Repository
- ▶ Quality of Data Warehouses

Recommended Reading

1. Ben Zur H. and Breznitz S.J. The effects of time pressure on risky choice behavior. Acta Psychologica. 47:89–104, 1981.

I

2. Chengalur-Smith I., Ballou D.P., and Pazer H. The impact of data quality tagging on decision complacency. In Proc. 2nd Conf. on Information Quality, 1997, pp. 209–221.
3. Chengalur-Smith I., Ballou D.P., and Pazer H. The impact of data quality information on decision making: an exploratory analysis. IEEE Trans. Knowledge and Data Eng, 1999, pp. 853–864.
4. Chengalur-Smith I. and Pazer H. Decision complacency, consensus and consistency in the presence of data quality information. In Proc. 3rd Conf. on Information Quality, 1998, pp. 88–101.
5. Fisher C.W., Chengalur-Smith I.N., and Ballou D.P. The impact of experience and time on the use of data quality information in decision making. Inform. Syst. Res., 14(2):170–188, 2003.
6. Gaeth G.J. and Shanteau J. Reducing the influence of irrelevant information on experienced decision makers Organ. Behav. Hum. Perform., 33:263–282, 1984.
7. Grether D.M., Schwartz A., and Wilde L.L. The irrelevance of information overload: An analysis of search and disclosure. Southern California Law Rev., 59:277–303, 1986.
8. Hansson S.O. Decision making under great uncertainty. Philos Social Sci., 26(3):369–386, 1996.
9. Johnson E.J., Payne J.W., and Bettman J.R. Information displays and preference reversals. Organ. Behav. Hum.Decision Process., 42:1–21, 1988.
10. Keller K.L. and Staelin R. Effects of quality and quantity of information on decision effectiveness. J. Consum. Res. 14:200–213, 1987.
11. MacGregor D., Lichtenstein S., and Slovic P. Structuring knowledge retrieval: An analysis of decomposed quantitative judgments. Organ. Behav. Hum. Decision Process., 42:303–323, 1988.
12. Payne J.W., Bettman J.R., and Johnson E.J. The adaptive decision maker. Cambridge University Press, 1993.
13. Stone D.N. and Schkade D.A. Numeric and linguistic information representation in multivariate choice. Organ. Behav. Hum. Decision Process., 49:42–59, 1991.
14. Wang R.Y. and Madnick S.E. A polygon model for heterogeneous database systems: The source tagging perspective. In Proc. 16th Int. Conf. on Very Large Data Bases, 1990, pp. 519–538.
15. Wang R. and Strong D. Beyond Accuracy: What Data Quality Means to Data Consumers. J. Manage. Inform. Syst., 4:5–34, 1996.

Information Quality Assessment

Leo L. Pipino
University of Massachusetts, Lowell, MA, USA

Synonyms

Information; Data

Definition

This entry uses the terms data and information interchangeably. The classical distinction is that data are raw facts whereas information is data in context or data that have been processed. Nevertheless, other than at an abstract level, it is a distinction that is often not made and one finds that the terms are used interchangeably. It is important to note that one individual's information can be data to another individual. This entry will also use the terms information quality dimension and information quality variable interchangeably.

Further, this chapter defines information of quality as information that is fit for use (or data of quality as data that is fit for use). This means that context and use plays an important role in evaluating information quality. For example, the instantaneous changes in a stock's price may be of importance to the stock trader who may trade stocks on a minute by minute basis. This instantaneous information, however, will be of minor importance to a long term investor whose strategy is "buy and hold" in which long term price trends are of more interest and use.

The process of measuring data quality must confront two major questions: (i) Exactly what should be measured? and (ii) What is the metric that will be used to measure the variable? Often, answering the first question is much more difficult than answering the second.

This entry does not address the area of software quality or the quality of information systems. The former has been examined extensively in the Computer Science literature and the latter has received a great deal of attention in the Management Information Systems literature.

Historical Background

Historically, prescriptions for the measurement of information quality have focused on specific variables. The often cited variables of accuracy, timeliness, completeness, and consistency are examples. Indeed, what dimensions to measure and the definition of what metrics to use are still debated today. This entry will not present a detailed recapitulation of the history and literature of information quality measurement. Rather, it will present one approach – broader in scope than simply measuring specific variables – that can be an effective way to measure data and information quality.

Foundations

As alluded to in the above paragraph, the typical approach to measurement is to measure specific variables or dimensions of information quality. This chapter advocates a broader approach that views the

assessment of information quality in an organization to include assessment of the management of the process of insuring information quality. Within this framework, the chapter also widens the scope of what is to be measured.

An assessment of management involvement and management practices is essential. These would include, but are not limited to, examining whether policy and procedures are in place, if learning is proactively fostered in the organization, if proper configuration and control is in place, and whether or not audits are periodically conducted.

Measurement of the information product includes evaluations of fitness for use, such as the existence and maintenance of metadata, whether or not data integrity constraints are in place, and assessments of many of the traditional quality dimensions. These evaluations would be more qualitative than quantitative in nature. An example of an instrument that has been used by a major healthcare facility to help in performing this assessment is the Data Quality Practice & Product Assessment Instrument described in [8].

One must, however, also measure the quality of information along specific dimensions at a more detailed level than described above. Again, the approach suggested here to measure specific information quality dimensions has a wider perspective than is typical. It is based on previous work [8,9]. A variant of this approach, for example, has been used in practice by a major teaching hospital.

The basic approach is to assess (i) subjective perceptions of information quality, that is, the perceived quality of information along some specific quality dimensions and (ii) objective quantitative measures of the same dimensions, when measurable.

The subjective assessments measure the perception of different stakeholders. Specifically three stakeholders are identified:

1. The collector – the individual (or group) that obtains and/or enters the data.
2. The custodian – the individual (or group) that is responsible for storing, retrieving, and providing the data.
3. The consumer – the user of the data.

Each may (and most likely) will have different perceptions of the quality of specific data and information. These subjective measures can provide comparisons across stakeholders and, as a consequence, provide valuable information regarding the disconnects among the different stakeholders with respect to specific data.

For many variables, these perceptions can also be compared to quantitative measures of the same variable. Here, the comparison may show, for example, that stakeholders think data is of good quality along a certain dimension when, in fact, the data is actually poor. Of course, the comparison can also show agreement between the quantitative measure and the subjective perception.

What should be measured? As mentioned earlier, what to measure can be more difficult to decide than how to measure it. The decision will be context dependent and must be determined by the organization and what variables (data quality dimensions) are important to it and its goals, objectives, and environment. The variables accuracy, timeliness, and completeness rank high among those mentioned often. Exactly how these are defined is important and the granularity of the definition and, consequently the measurement, must be carefully considered. For example, accuracy is always mentioned in the same breadth as data quality. Exactly how can one define accuracy. Assume that the unit of measurement is the record. Further assume that records have missing values, that is, they are incomplete. Will the record be considered inaccurate or will this type of problem be defined as incompleteness and accuracy defined more as "free from error," that is that fields in the record with values (those that are not missing) are correct. This chapter will not answer this question. These are questions that the organization and its analysts must answer in the specific context of the organization. Different organizations will answer the question somewhat differently. This simple example points out the issue and helps focus on the importance of the definitions in determining what is to be measured.

The metric itself may be easier to define. For illustration purposes, if the incompleteness definition given above is used, then a simple ratio of complete to total records can be used. Assuming that in most databases the number of complete records far outweigh the records with missing data, the formulation of one minus the above ratio could be used.

Although not an exhaustive nor a completely independent set, a comprehensive set of dimensions from which one can choose has been provided in [13]. The list is not meant to exclude other possible variables. A subset from this list or other variants can be used. The important point is to clearly define what is to be measure.

The subjective perceptions can be measured using questionnaires whose questions are rated numerically using a Likert type scale. One example that has been used in practice is the Information Quality Assessment instrument (IQA) [8] or a subset of the IQA. For example, such statements as shown below are evaluated in the context of a specific organization and data set or database:

1. The information is incomplete.
2. The information is believable.
3. The information is correct.

In the case of the IQA, the respondent (any of the stakeholders) would provide an evaluation on a scale from 0 to 10 with 0 representing complete disagreement with the statement and 10 representing complete agreement. Of course, one could use a different numerical range for the evaluations. It is important to reemphasize that with the IQA the respondents are subjectively assessing the pool of information or database(s) which the organizations has chosen to evaluate. It is not an evaluation of one item or piece of solitary information.

For those variables that can be quantified, metrics can be defined and quantitative measurements can be performed. For example, one might physically measure the number of incomplete records and use a metric similar to the following:

Degree of Completeness = 1 – Number of Incomplete Records/Total Number of Records

For other dimensions, such as the believability of information, it may be decided that only the subjective perceptions will be used. Alternatively, if a reasonable definition of, for example, believability can be specified as a function of other quantitatively measured variable, and this is a big if, then some type of aggregation function can be used. An example might be a weighted average of the numerical values of the quantitative variables where the weights indicate relative importance, range from zero to one, and sum to one. Such aggregation, if done at all, must be performed with care since one must always be attentive to inappropriately combining scales of different types (ratio, interval, ordinal, and nominal [7]). This can lead to inappropriate use. Accepting the resulting precise numerical values as ratio or interval data could lead to poor decisions on the part of the decision maker.

In sum, at the detailed measurement level, the approach recommended here is a comparative approach wherein after what to be measured has been determined:

1. Subjective perceptions are measured.
2. Objective/quantitative measurements are performed. This would then lead to:
3. Comparison of results.
4. The taking of action where necessary to improve the data quality.

Of course, either step (1) or (2) can be used independently, that is, as the sole measure.

Recent work merits mention in a chapter on the measurement of data quality. It has been suggested [5] that the construct of utility be incorporated into the information quality measurement. Here utilities would indicate the relative value or importance of certain records within the set. These would be taken into account in generating quantitative assessments for a specific dimension. Note that, as modeled in [5] utility is not a measure of the relative importance or value of different variables (dimensions) but rather the relative importance of the records within a set measured along one dimension. This is ongoing research but can easily be incorporated into the metrics used in step 2 above.

Key Applications

The key application is the assessment of the quality of information that an organization uses.

Cross-references

- ▶ Data Deduplication
- ▶ Data Quality Assessment
- ▶ Data Quality Dimensions
- ▶ Information Quality Policy and Strategy

Recommended Reading

1. Ballou D. and Pazer H. Modeling data and process quality in multi-input, multi-output information systems. Manage. Sci., 31(2):150–162, 1985.
2. Ballou D., Wang R., Pazer H., and Tayi G. Modeling information manufacturing systems to determine information product quality. Manage. Sci., 44(4):462–484, 1998.
3. Batini C. and Scannapieco M. Data Quality: Concepts, Methodologies and Techniques. Springer, New York, 2006.
4. Codd E. The Relational Model for Database Management: Version 2. Addison-Wesley, Reading, MA, 1990.
5. Eden A. and Shankaranarayanan G. Understanding impartial versus utility-driven quality assessment in large datasets. In Proc. 12th Conf. on Information Quality, 2007, pp. 265–279.
6. Huang K., Lee. Y., and Wang R. Quality Information and Knowledge. Prentice-Hall, Englewood Cliffs, NJ, 1999.

7. Krantz D., Luce R., Suppes P., and Tversky A. Foundations of Measurement: Additive and Polynomial Representation. Academic Press, New York, 1971.
8. Lee Y.W., Pipino L.L., Funk J.D., and Wang R.Y. Journey to Data Quality. MIT Press, Cambridge, MA, 2006.
9. Pipino L., Lee Y., and Wang R. Data quality assessment. Commun. ACM., 45(4):211–218, 2002.
10. Redman T. Data Quality: The Field Guide. Digital Press, Belford, MA, 2001.
11. Strong D., Lee Y., and Wang R. Data quality in context. Commun. ACM., 40(5):103–110, 1997.
12. Wang R., Lee Y., Pipino L., and Strong D. Manage your information as a product. Sloan Manage. Rev., 39(4):95–105, 1998.
13. Wang R. and Strong D. Beyond accuracy: what data quality means to data consumers. J. Manage. Inf. Syst., 12(4):5–34, 1996.

Information Quality Policy and Strategy

Yang W. Lee
Northeastern University, Boston, MA, USA

Synonyms

Data Quality

Definition

Information quality policy and strategy is an emerging area of study and practice. Information quality policy and strategy involve managing both information product and information practice. A recent piece of literature structured information quality policy around ten key guidelines and provided a questionnaire on four key areas of information product: source data, metadata, system-based data integrity, and data quality, all of which were assessed by all stakeholders. The questionnaire also included the assessment of strategic, managerial, and operational level of information practice [3].

Conventionally, managing information quality means ensuring data integrity of stored data in a database and managing authoritative access and security. Increasingly, organizations manage and use heterogamous databases in dispersed strategic business units. With globalization trends, mergers and acquisitions, and virtual collaboration, managing information entails much more beyond managing data in a stored database within a single organization. Managing information in today's organization, therefore, means managing the comprehensive information process: data collection, storage and utilization processes [3]. High-quality information is used to compete in the market place and is a new strategically competitive asset [5]. To improve the quality of information available to information consumers, organizations need to manage information with an established policy and strategy.

Historical Background

While there are some early seminal papers, information quality (IQ) only emerged as a recognized field of information technology (IT) research in the mid-1990s. For example, the first International Conference on Information Quality was held in 1996 at MIT. In 1998, Wang et al. [6] published "Manage Your Information as a Product," which argued for the concept of treating information as a product which is intentionally managed and produced for consumers who have expectations of its quality. In 2006, Lee et al. [3] published "Journey to Data Quality," which included the first set of generic information quality policies, arguing for the principles of managing information with an institutionalized set of policies that structures decisions on information.

Foundations

This section provides ten policy guidelines that should form the basis of an organizational data quality policy and strategy [3].

Treat Information as a Product, Not By-Product

Treating information as a product, not by-product, is the overarching principle that informs the other guidelines [6]. This principle must be continually communicated throughout the organization. It must be a part of the organization's ethos. More details on the approach are available in Wang et al. [6].

Establish and Maintain Information Quality as a Part of the Business Agenda

An effective policy requires that an organization maintain information quality as part of its business agenda. The organization must understand and document the role of data in its business strategy and operations. This requires recognition of the data needed by the firm's business functions to complete its operational, tactical, and strategic tasks. It should be clearly understood that this data is fundamental to the competitive positions of the firm.

The responsibility for ensuring that data quality is an integral part of the business agenda falls on senior

executives. Senior executives should understand their leadership is critical to success and that they must be proactive in fulfilling the leadership role. They must take an active part in ensuring data quality in the organization so as to utilize quality information as a strategically competitive advantage.

Ensure that Information Quality Policy and Procedures are Aligned with Business Strategy, Business Policy, and Business Processes

The primary function of data is to support the business. To do so, a data quality policy must be in place and consistent with an organization's business policy. This implies that a broad view of what constitutes data policy is necessary. An integrated, cross-functional viewpoint is a prerequisite to achieving an aligned data quality data quality policy. Proper alignment will also foster the seamless institutionalization of data quality policy and procedures into the fabric of the organization's business processes.

Improper alignment or too narrow of a focus can cause data quality problems and prove disruptive to the organization. It can lead to conflict which could otherwise be avoided. For example, a view that restricts data quality policy to technical details of data storage and ignores the uses of the data at different levels of decision making would result in misalignment and subsequent problems. In short, data quality policy should reflect and support business policy. It should not be independent or isolated from business activities.

Establish Clearly Defined Information Quality Roles and Responsibilities as Part of Organizational Structure

Clear roles and responsibilities for data quality must be established. There should be specific data quality positions within the organization, not ad hoc assignments to functional areas in times of crisis. Additionally, one of the fundamental goals of data quality function should be to identify data collectors, data custodians, and data consumers as well as make these members of the organization aware of which of these roles that they and others play.

Roles and responsibilities specific to the data quality function range from the senior executive level to the analyst and programmer level. Current examples of titles in practice include chief Data Officer, Senior VP of Information Quality, Data Quality Manager, and Data Quality Analyst.

Ensure Data Architecture is Aligned with Enterprise Architecture

The organization should develop an overall data architecture that is aligned with and supports its enterprise architecture. Enterprise architecture means the blueprint of the organization-wide information and work infrastructure. The data architecture helps promote a view of the information product which is consistent across the enterprise. This promotes accessibility of information across the organization.

The data architecture reflects how the organization defines each data item, how data items are related, and how the organization represents this data in its systems. It also establishes rules and constraints for populating individual data items. All of theses are essential elements of any product – a definition of the item, statements indicating how it is related to other entities in the business, and what constraints exist for populating it in various databases.

To ensure that the data architecture is and remains aligned with the enterprise architecture entails a number of specific and detailed tasks. For example, establishing a data repository is critical to the development and maintenance of viable data architecture. The organization should generate a set of metadata that clearly defines data elements and provides a common understanding of the data. This enables efficient data sharing across the organization.

Be Proactive in Managing Changing Data Needs

The data needs of consumers change over time. The term *consumers* means any individuals and enterprises internal or external to the organization who use the data of their organization. To maintain a high degree of data quality, the organization must be sensitive to ever-evolving environments and changing needs. This entails continual scanning of the external environment and markets along with the changing needs of internal data consumers.

It is also critical to the maintenance of high data quality that the rationale for changes in policy and procedures is clearly and promptly communicated throughout the organization. Steering committees and forums can play a crucial role in the dissemination of this information.

The organization must also be cognizant of local variances as they relate to global data needs. In today's multinational and global firms, long-standing local culture and customs exist. These local differences

must be clearly identified and policy adapted to incorporate these local variances accordingly.

Have Practical Data Standards in Place

It is easy to get people to agree that standards should be in place. It is much more difficult to reach agreement on what should be standardized and the prescription of the standard. At times, it is difficult to obtain agreement on the definitions of the most basic data elements. Data standards entail many areas of data practice. Addressing the following questions will help inform the process for instituting practical data standards internally. If external standards are to be used, which ones should be chosen? Does the standard apply locally or globally? In either case, how and when will it be deployed? What processes are appropriate to monitor and adopt changing standards over time? How should they be documented and communicated?

The process an organization uses to determine whether to choose external or internal standards, and under what conditions, should be continually developed. The preferred solution would be to use a standard which has been created and maintained by an organization other than the organization using the standard. If this is possible and acceptable, then existing standards to adopt must be determined. This may be a choice between international, national, or industry standards or between two national standards. For example, the International Standards Organization (ISO) has a standard set of country codes. One's first choice would be to choose the international standard. Other things being equal, this provides the greatest degree of flexibility and lowest costs to the firm.

Plan for and Implement Pragmatic Methods to Identify and Solve Data Quality Problems, and Have in Place a Means to Periodically Review Data Quality Practice and the Data Product

Pragmatic methods based on diagnostic methods and measures should be promoted as standards across the organization. Example methods include system-based data integrity, data quality assessment, and data product and practice assessment methods.

Periodic reviews of the data quality environment and the overall data quality of the organization is a must. One method to achieve this is to assess the data quality practice and the firm's data quality using a standard assessment survey. Multiple assessments over time with a standard survey will provide a series of assessments against which the firm can measure its progress. Further, if an evaluation of an exemplary organization is available, the firm can use this exemplar's results as a benchmark and evaluate its performance against it.

A practice related to the auditing task involves certification of data. Data of both transaction systems and data warehouses must be included in the certification process. A robust process of data certification will ensure that data practice and data product will be of high quality. As a consequence, a data certification policy must be put in place. The policy should stipulate that sources for the information be clearly identified; including definitions, valid values, and any special considerations about the data.

Foster an Environment Conducive to Learning and Innovating with Respect to Data Quality Activities

Initiatives that contribute to a learning culture would include the use of a high-level data quality steering committee, the installation of forums across the organization, the stress on continuous education and training. All should be strongly supported and conveyed by senior management.

The organization should put in place a policy of rewards, and if necessary, sanctions, that are consistent with achieving a high level of data quality and maintaining this high level. What should not be discouraged is the identification and reporting of data quality problems, even if these prove embarrassing to management. Employees should feel free and safe to report data quality problems openly. A formal mechanism for reporting such problems can help in this regard. Establishing a forum for data collectors, data custodians, and data consumers to voice and share data quality problems and discuss data quality issues will be of great value. This will also aid in the dissemination of the experiences of the collectors, custodians, and consumers and foster an appreciation for each other's perspectives and roles.

It is especially important for data collectors to understand why and how consumers use the data. Data collectors also must adhere to the data quality principles as they enter data even though they are not the prime beneficiaries of such activity and do not usually get measured on this portion of the data entry task. Data consumers are critical to providing appropriate feedback on the quality of the data they receive, including whether it is needed data. The function would include apprising the data collectors of changes in data needs. Data

custodians must recognize that although they are not directly responsible for the data, they must understand its purpose and use. As with data collectors, data custodians should be aware of how and why data consumers use the data.

Establish a Mechanism to Resolve Disputes and Conflicts Among Different Stakeholders

Data policy and jurisdictional disputes, data definition disagreements, and data use debates will occur. An organization must have in place a mechanism to resolve theses differences and conflicts. These could be a set of hierarchically related mechanisms corresponding to different levels of the firm. Their form is not as important as their existence and that they have a clear charter specifying their responsibilities and authority.

Any of a number of conflict resolution techniques and mechanisms can be used. The specifics must be consistent with an organization's business strategy and culture. Examples are the use of steering committees, data quality boards, data quality work groups, and data quality councils. The scope of responsibilities, the degree of authority, and clearly defined lines of reporting and communications must be specified. Ambiguity in the specifications will render theses resolution mechanism ineffective.

It is most important that a historical record maintains disputes and their resolutions. This institutional memory will serve the organization well as it adapts to changing future environments.

Key Applications

Early adaptors are applying information policy and strategy in their organizations to great avail. Public sector agencies are advanced in their adoption of the policy and strategy since most public agencies approach information quality policy for their planning and implementation of information management that is guided by the mandated Information Quality Act. The public agencies such as the Intelligence Community, Environmental Protection Agency, Department of the Interior, and Department of Housing and Urban Development also align information quality policy with their Enterprise Architecture and Governance policy. In the private sector, most organizations approach information quality policy initially from their data warehouse or customer data management point of view that they need a policy to share the information at the enterprise level. Software vendors align some of the pragmatic policy issues as the underlying methodology which frames their tools; such as data integration, business intelligence, data migration, ETL, and data profiling tools.

Future Directions

The importance of data quality polices and strategies, good data practices and principles, effective data quality training, and communication within the organization cannot be overemphasized. This often overlooked aspect of data quality practice can mean the difference between those that are successful in this "journey to data quality" and those that are not [3].

The ten policy guidelines presented can serve as potential areas for future research. The needs of information consumers are well-defined; reliable measures of data quality assessment have been developed; and methods for analyzing, diagnosing, and improving information quality are readily available. The data products are studied and used in organizational practices. However, the data practice area including policy and strategy is in an early stage and will mature in the future as more firms demand policy and strategy for improving and aligning their information quality agenda with their business agenda.

More research is also needed to integrate the perspective of data product and data practice. To integrate both perspectives effectively, researchers will need to apply multi-disciplinary methods and theories. For example, "context-reflective data quality problem-solving" [2] uses various disciplinary areas and integrates data product and data practice. Another example [4] is seen with "process-embedded data integrity" which integrates data product and data practice. Studies focused on IQ policy and strategy studies are also needed which include organizational outcome measures to be used for directing efforts at policy analysis and strategy forming level. Strategy studies are also needed to develop methods for costing and justifying investments in IQ improvements and assessing competition in the market. The purpose of these research efforts is to develop theories, methods, tools, and policies to help organizations proactively manage their information. Thus, ensuring the delivery of quality information to information consumers and the provision of quality information as a strategic asset; not only to its own organization, but to partner organizations and the extended enterprise of the globalized economy as well.

Cross-references

► Information Quality Assessment
► Information Quality and Decision Making
► Information Quality: Managing Information as a Product

Recommended Reading

1. Ballou D.P., Wang R.Y., Pazer H., and Tayi G.K. Modeling information manufacturing systems to determine information product quality. Manage. Sci., 44(4):462–484, 1998.
2. Lee Y.W. Crafting rules: context-reflective data quality problem-solving. J. Manage. Inf. Syst., 20(3):93–119, 2004.
3. Lee Y.W., Pipino L., Funk J., and Wang R.Y. Journey to Information Quality. MIT Press, Cambridge, MA, 2006.
4. Lee Y.W., Pipino L.L., Strong D.M., and Wang R.Y. Process-embedded data integrity. J. Database Manage, 15(1):87–103, 2004.
5. Matsumura A. and Shouraboura N. Competing with quality information. In Proc. 1st Conf. on Information Quality, 1996, pp. 72–86.
6. Wang R.Y., Lee Y.W., Pipino L.L., and Strong D.M. Manage your information as a product. Sloan Manage. Rev., 39(4):95–105, 1998.
7. Weil P. and Broadbent M. Leveraging the New Infrastructure: How Market Leaders Capitalize on IT. Harvard Business School Press, Cambridge, MA, 1998.

Information Repository

► Data Warehouse

Information Retrieval

Giambattista Amati
Ugo Bordoni Foundation, Rome, Italy

Synonyms

Document retrieval; Text retrieval

Definition

Information Retrieval (IR) deals with the construction of automatic systems that allow users to inquire about textual data of any kind through natural language queries. The retrieved information of IR systems may vary from a ranked list of relevant textual items of any kind, such as full documents or their excerpts, or can be distilled into more elaborated forms, such as document summaries or answers to questions. Information Retrieval is an empirical science studying the representation, storage and access to information, and covers a large number of interdisciplinary topics of theoretical computer science including information theory, machine learning, coding theory, probability theory, programming theory, computational semantics, natural language processing, logics and algebra. From a practical perspective research investigation in IR includes: data representation, storage and retrieval, such as indexing, data encoding and text compression, document and term classification and clustering, systems architecture, distributed systems, document-query matching functions (IR models); user-oriented studies and aspects of behavioral science, such as data visualization, browsing, user interfaces, system evaluation, user relevance feedback and automatic query-expansion. With the advent and diffusion of the Web and the dramatic increase of public available information sources IR research also focuses on efficiency in terms of query response time and storage space of indexes in a distributed setting, as well as on personalized search where results can be filtered and adapted to user's area of interest taking into account, for example, geographical knowledge and user's historical data.

Historical Background

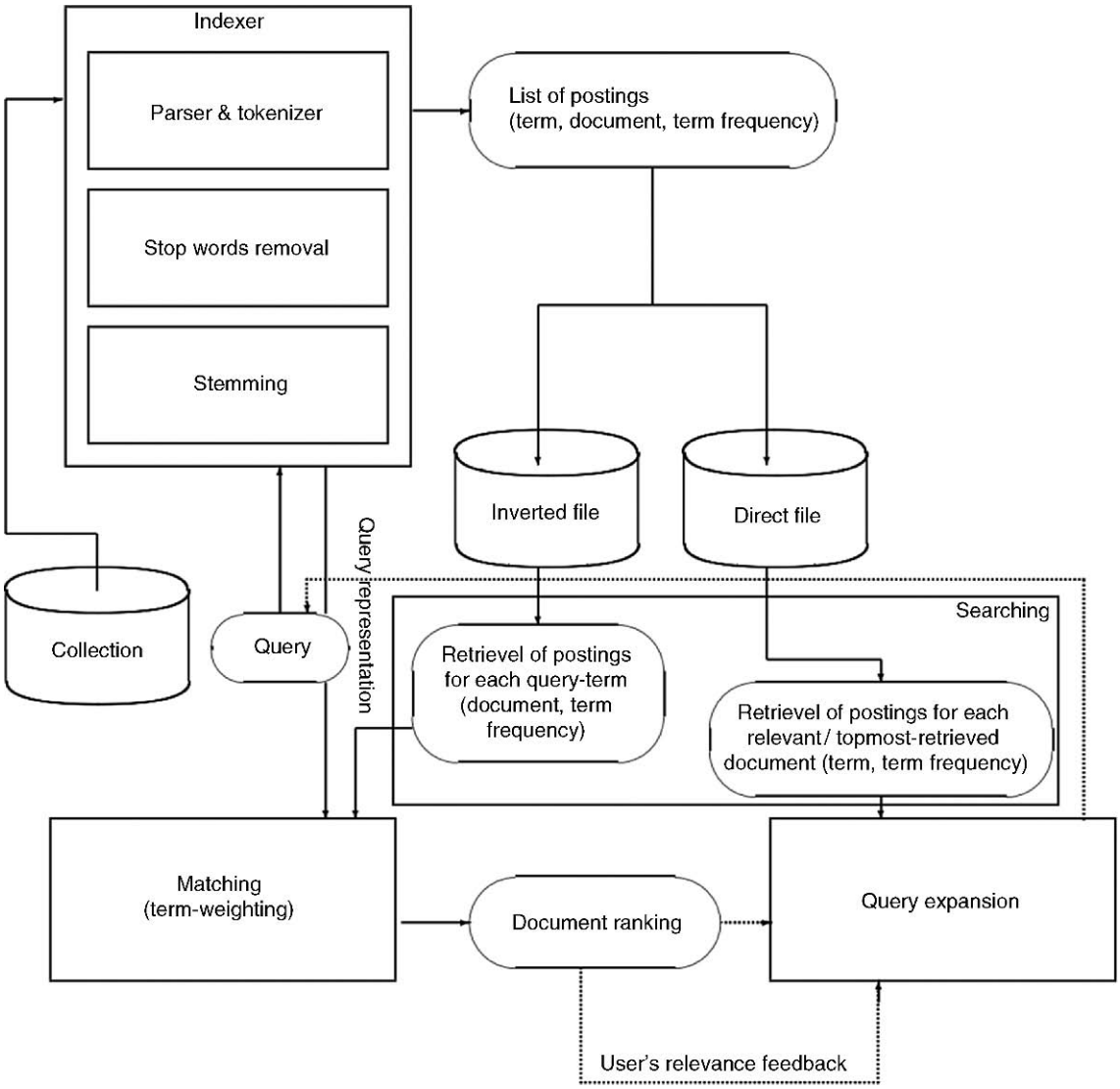
Information Retrieval has its origins in the 1950s to support the librarians activities of indexing and accessing textual collections. Hans Peter Luhn is one of the pioneers of IR [5] with his studies on automatic indexing, which concerns the assignment of significant keywords to documents. In the early stage of IR due to the limitations of computer capabilities document retrieval was restricted to satisfy the boolean exact match between the query-terms and document surrogates (title, subject headings or abstract). Luhn thought that the automation of indexing and abstracting was less prone to errors than human indexers. Before Luhn's original ideas, Shanny, the founder of the mathematical theory of communication, was the first to consider text generation as a "sequence of words" and processed as a discrete information source by a stochastic process (a discrete Markov process) [10]. Following Shanny's idea, Mandelbrot derived theoretically the empirical model by Eustop on word distribution and further studied by Zipf [14]. Mandelbrot showed that text generation is finding the least costly method of coding as obtained in the classical problem in communication theory [6]. The law of Estoup-Zipf-Mandelbrot establishes that the logarithm of rank of words by decreasing

frequency is in linear relation with this frequency. An important milestone in the history of Information Retrieval is the introduction of evaluation measures for IR by Cleverdon. in 1953 Cleverdon led a project of Librarian of Cranfield College of Aeronautics to assess retrieval with a "document source – question" method, that consists in collecting questions and their relevance judgments from individuals; the Cranfield test collection is still publicly available for experimentation in IR [3]. In 1960 Maron and Kuhns used the term "probabilistic indexing" in the theory of IR to model the concept of relevance explicated in terms of the theory of probability [7]. Nevertheless, it was the vector space model, a model not based on probability theory, that influenced most of the research for some decades. The SMART project, dedicated to the realization of one of the first IR systems, the SMART system, was initiated at Harvard University in 1961 by Gerard Salton, but actually most of the research was conducted at the Cornell University [9]. The SMART system used the cosine of the index vectors derived from documents and search requests to obtain for each document a coefficient of similarity with each search request. After Maron and Kuhns, a probabilistic model of relevance was introduced by Stephen Robertson and Karen Spärck Jones [8] that has led to the development of the OKAPI system. The probabilistic model is based on Spärck Jones' observation that the significance of a term in a document is due to its rareness in the collection, and it can be measured by the logarithm of the inverse of the relative frequency of the term in the collection (document frequency) [11]. In the early 1970s the "cluster hypothesis" was stated by Jardine and van Rijsbergen [4]. According to the cluster hypothesis both efficiency and effectiveness of search could benefit from clustering similar documents because such clusters are easily retrieved by the same search requests, while Salton thought that clustering would have reduced effectiveness in favor of a better response time.

The first main international conference on IR, ACM SIGIR, was held in 1978 [1]. The first commercial search engines appeared in the 1980s, while the 1990s saw the birth of the first web search engines. Since the beginning of the TREC (Text REtrieval Conference) conference series in 1992, a conference organized by the US government's National Institute of Standards and Technology and dedicated to large-scale evaluation of text retrieval methodologies, very large test collections of full-text documents and standards for retrieval evaluation are available. Evaluation is an important issue in IR, therefore TREC has a significant impact in research because provides an objective evaluation of fresh techniques and approaches, and because promotes new specialized retrieval tasks, as well as, the transfer of emerging new ideas into commercial systems and Web search engines.

Foundations

Information Retrieval systems have four main components: query representation and document indexing, term and document retrieval, query-document matching, relevance feedback processing (see Figure 1). Typically, a user submits a query made up of a few words and phrases, or several sentences. The internal representation of the original query and the text extracted from a document are both processed by the same tokenizer: a *parser* extracts the words taking into account specific properties of the language, then most frequent words and other words that have a functional role in the text (*stop-words*) are removed, and finally the constituents of the lexicon of the system (*tokens*) are created by removing linguistic suffixes or prefixes (*stemmer*), their multiplicity of occurrence in the text is recorded. The text of each document is indexed into two files: the *inverted* and the *direct* file. Both files contain the set of *pointers* of the collection, also knows as *postings*, that is the matrix containing all possible term-document pairs of the collection. With the inverted file one can access the set of pointers, e.g., (term, document, frequency), and thus the set of documents containing a given term together with some extra information about these pointers, that is the frequency of that term in these documents and possibly the positions occupied by that given term in each document. With the direct file, one can instead access the set pointers, e.g., (document, term, frequency), that is the set of terms contained in a given document together with the frequency of each term in that document. Positions of terms within documents are used to restrict search with the use of proximity operators. For example, one can submit the query "information retrieval" wishing to find and rank by relevance all and only all documents containing the word "information" followed by the word "retrieval." While *information AND retrieval* is the query to find and rank all and only all relevant documents containing both the words "information" and "retrieval" irrespective of the positions occupied by these words in the text. Frequencies are instead essential to obtain a ranked list of documents.



Information Retrieval. Figure 1. A conceptual architecture for an Information Retrieval system.

The direct file is used to perform other post-processing activities, such as *automatic query expansion*, or to *cluster* the retrieved documents into homogeneous or similar classes of documents, or to present results in different output formats (e.g., into an XML form).

Both inverted and direct file are stored in a compressed form, and it is possible to achieve a very good rate of compression with respect to the size of the original textual data. For example the inverted file of the TREC corpus WT10g containing 1,692,096 documents of 10 GB of text and 280,571,311 pointers, can be compressed in about 385.5 MB, each pointer (term-document, term frequency) requiring an average of 11.5 bits of information only. In general, the most frequent words are declared in a special list of words, the *stoplist*, and are not indexed because they require the storage of too many pointers (the size of the collection in the worst case). A comprehensive study of compression algorithms for text retrieval can be found in the book *Managing Gigabytes* [13].

The kernel of an information retrieval system is the query-document matching model, producing the document-scores for the given query, also known as the *retrieval status value* (RSV) of the documents for that query. The matching model is the theoretical component responsible of the effectiveness and the quality of the retrieval.

Similarly to any other empirical science, Information Retrieval makes inferences by analysis of the empirical data consisting of three phases: *model specification*, *parameter estimation* and *model evaluation*. However, unlike other empirical sciences, IR has two types of data: the postings, that is the elements of the term-document matrix, and the user relevance data set, that are in general provided by a set of document-query relevance assessment pairs. The relevance data set can be used to accomplish two different tasks: query re-formulation for improving document retrieval or the evaluation of system performance.

Due to the existence of relevance feedback data, IR has two different kinds of models: the *query-language model* and the *term-document model*. The query models aim at providing the weights of the query terms irrespective of the observed document. The term-document model instead compares and weights the frequency of the term within a document with respect to its frequency in the collection.

For example, the query "What is a prime factor?", which is transformed into the query "prime factor" after stop-words removal and stemming, can be simply represented as the vector (prime = 1, factor = 1). After a first pass retrieval in absence of user's feedback, one may deem the first retrieved documents as relevant and assume that the terms contained in this small portion of retrieved documents as a sample of the term population relative to the topic "prime factor." Then, the query-language model will assign new weights to the original query-terms and add new terms to the query. For example, retrieving just the first three documents the new query might be (prime = 1.3581, factor = 1.2327, integ = 0.2154, number = 0.1778, primal = 0.0941,...). The term-document weights assigned by the term-document model will be resized according to query-term weights, that is as the inner product of these two weight vectors.

A theory on evaluation of IR systems is mainly developed in van Rijsbergen's book [12]. The effectiveness of an IR system is evaluated by two standard mea sures: *recall* and *precision*, that are defined as follows:

$$Recall = \frac{\mu(Rel \cap Ret)}{\mu(Rel)}$$

$$Precision = \frac{\mu(Rel \cap Ret)}{\mu(Ret)}$$

where $Ret = \{d|d \text{ is retrieved}\}$ and $Rel = \{d|d \text{ is relevant}\}$, and $\mu = |\cdot|$ is the counting measure. Then, $|Rel \cap Ret|$ is the number of relevant and retrieved documents, $|Ret|$ is the number of retrieved documents, and $|Rel|$ is the number of relevant documents.

Key Applications

Main applications of Information Retrieval concern the construction of search engines either for specific domains, like biomedicine and genomics, law, web, blogs, or adapted to particular types of document structure (e.g., XML or hypertext documents) search engines, or dedicated to multimedia and digital libraries.

Future Directions

Notwithstanding the increase of computer processing power, most of IR applications deal with very large collections that cannot be in general processed by only one server. Both efficient implementation and distributed versions of IR systems are required. The design of efficient partitions of very large indexes that need to be distributed over a cluster of machines is an important research topic. Also, large community of users may wish to share their information and knowledge, but not local indexes, and therefore merging local search results to obtain one effective global document ranking is a challenging research issue (*data fusion*). Most of the new technology for information retrieval is becoming more and more an asset of the most popular web search engines, and therefore it is mainly the market that influences new academic research directions. Although social network analysis has achieved a quite mature technology based on variations of Markov chain models with stationary probability distributions, a new social phenomenon is now emerging in the web, the *social tagging*; systems also provide collaborative tools to Internet users to store and search structured comments of web pages. Query search on movies, music or books will be in the very next future conditioned, for example, by opinions or by recommendation. More generally, search will be more and more personalized and tailored to one's own profile or to all other profiles similar one's own. The major techniques to discover statistical relationships for hypertext documents and hyperlinks can be found in Chakrabarti's book [2].

Data Sets

Most of data sets and test collections can be found at the TREC (NIST) web site http://trec.nist.gov/data.

html. There are several types of test collections concerning different range of IR applications. To cite few of them, there are the *ad hoc* collections, that are dedicated to the standard document retrieval based on a primitive notion of user's relevance, *web* test collections, that are dedicated to web retrieval, *Blog Track* test collections, that contains a large collection of permalinks from the blogosphere.

URL to Code

There are several open source IR systems on the web. In the following there is a list of the most popular and advanced academic IR research systems.

1. Indri and Lemur search engines developed by the Carnegie Mellon University and the University of Massachusetts, Amherst, United States: http://www.lemurproject.org/
2. Terrier developed by University of Glasgow, United Kingdom: http://ir.dcs.gla.ac.uk/terrier/.
3. Wumpus developed by University of Waterloo, Canada: http://www.wumpus-search.org/
4. Zettair by the RMIT Univerisity of Melbourne in Australia: http://www.seg.rmit.edu.au/zettair/

Other IR systems are the open source Apache Lucene (http://lucene.apache.org/) and the SMART system (ftp://ftp.cs.cornell.edu/pub/smart/).

Cross-references

- ► Biomedical Scientific Textual Data Types and Processing
- ► Clustering
- ► Digital Libraries
- ► Information Retrieval Evaluation Measures
- ► Information Retrieval Models
- ► Information Retrieval Operations
- ► Query Expansion Models
- ► Relevance Feedback
- ► Text Indexing Techniques

Recommended Reading

1. Annual International SIGIR Conference, Proceedings of the ACM Special Interest Group on Information Retrieval Conference. http://www.sigir.org/.
2. Chakrabarti S. Mining the Web: Discovering Knowledge from Hypertext Data. Morgan-Kauffman, 2002.
3. Cleverdon C. The cranfield test on index language devices. Aslib Proc., 19:173–192, 1967.
4. Jardine N. and van Rijsbergen C.J. The use of hierarchic clustering in information retrieval. Inform. Storage Retr., 7 (5):217–240, 1971.
5. Luhn H. A statistical approach to mechanized encoding and searching of literary information. IBM Journal of Research and Development, 1:309–317, 1957.
6. Mandelbrot B. An informational theory of the statistical structure of language. In Communication Theory, the Second London Symposium. W. Jackson (ed.). Butterworth, London, 1953, pp. 486–504.
7. Maron M.E. and Kuhns J.L. On Relevance, Probabilistic Indexing and Information Retrieval. J. ACM 7(3):216–244, 1960.
8. Robertson S.E. and Sparck-Jones K. Relevance weighting of search terms. J. Am. Soc. Inform. Sci., 27:129–146, 1976.
9. Salton G. and Lesk M.E. The SMART automatic document retrieval systems – an illustration. Commun. ACM, 8 (6):391–398, 1965.
10. Shanny C. A mathematical theory of communication. Bell Syst. Tech. J., 27:379–423 and 623–656, 1948.
11. Sparck J. K. A statistical interpretation of term specificity and its application in retrieval. J. Doc., 28(1):11–21, 1972.
12. Van Rijsbergen C. Information Retrieval, 2nd edn. Butterworths, London, 1979.
13. Witten I.H., Moffat A., and Bell T.C. Managing Gigabytes. 2nd edn. San Francisco, California, 1999.
14. Zipf G. Human behavior and the principle of least effort. Addison-Wesley, Reading, Massachusetts, 1949.

Information Retrieval Models

Giambattista Amati
Ugo Bordoni Foundation, Rome, Italy

Synonyms

Document term weighting; Ad hoc retrieval models; Term-document matching function

Definition

An Information Retrieval (IR) model selects or ranks the set of documents with respect to a user query. Text in documents and queries is represented in the same way, so that document selection and ranking can be formalized by a matching function that returns a Retrieval Status Value (RSV) for each document of the collection. Most IR systems represent document contents by a set of descriptors, called terms, belonging to a vocabulary **V**.

Main IR models define the query-document matching function that weights the query terms occurring in a document according to four main approaches:

- The estimation of the probability of user's relevance *rel* for each document **d** and query **q** with respect to a set $R_{\mathbf{q}}$ of training documents

$$\text{Prob}(rel|\mathbf{d}, \mathbf{q}, R_{\mathbf{q}})$$

- The computation of a similarity function between queries and documents in a vector space

$$SIM(\mathbf{d}, \mathbf{q})$$

- The estimation of the probability of generating the document **d** given a query **q**,

$$\mathbf{p}(\mathbf{d}|\mathbf{q})$$

- The information carried by the query terms in the document, that is that is the number of bits necessary to code X_i occurrences of the query terms $\mathbf{t}_i \in \mathbf{q}$ in the document:

$$-\log_2 \text{Prob}(\mathbf{d}|X_1,\ldots,X_{\mathbf{q}})$$

Historical Background

The history of Information Retrieval goes in parallel to the development of IR models. In general, an IR system is mainly identified with retrieval function employed to rank documents, because it is the retrieval *effectiveness* that matters in IR systems. In the early days, research focused on methods for automatic indexing in contraposition to the manual activities made by librarians. Early works concerned the construction of effective methods for keywords selection to represent succinctly the documents, and document-query matching thus was performed by Boolean search of the query terms in the index of such surrogates of the documents. A full exploitation of a probabilistic model was achieved in the 1990s with the birth of the BM25 ranking formula [7]. Before BM25, most of the theoretical investigation was devoted to the vector space model, the first parameter free model for ranking documents [8]. It was only in the late 1990s that other new models appeared on the scene, such as the language models [6] and the information theoretic models that include the Divergence From Randomness models [1]. Boolean model remains attractive because it requires less computational cost both in terms of size of the indices and response time, and quite a few models tried to extend the Boolean model with fuzzy or logical operators, yet the potentialities of such algebraic and logical models have not been fully exploited.

Foundations

IR models can be classified into four types: probabilistic models, algebraic and logical models, information theoretic models and Bayesian models.

Probabilistic models require a training set of data consisting of set of documents which are assessed relevant to a set of queries by users. Algebraic models assume that both queries and terms are represented by vectors, and that the similarity between queries and documents is obtained through a specific normalization of the inner product of these two vectors. In logical models queries and terms are represented by propositions, and the dependency between queries and documents is given by an entailment operator. Information theoretic models are based on the notion of coding cost of terms in the documents, the most informative documents being those generated by the least probable configurations of terms. Bayesian models process two sources of evidence for terms, the term frequencies in the document and in the collection, that are combined to produce an estimate of the probability of the term in the document.

Binary independence model (BIR) and BM25 are probabilistic models. Vector space, Boolean, fuzzy and logical models belong to the algebraic and logical models. The 2-Poisson model [5], and Divergence From Randomness models (DFR) belong to the class of information theoretic models. Language models are instead derived by Bayesian methods.

Probabilistic Indexing

As first IR models, it is interesting to consider the automatic indexing techniques used in the origins of IR. In statistics, observations of empirical data are predicted by stochastic models, that is a probabilistic model which takes into account the presence of some randomness in its variables. In particular, the IR model predicts a probability distribution of possible estimates for term frequencies. An hypothesis that describes the distribution of frequencies is formulated, and the values of inherent parameters of the distribution are estimated by fitting the empirical data to the specified model. Finally, the hypothesis on the distribution form (e.g., Poisson, gaussian, etc.) is accepted or rejected according to an error associated to the fit.

A stochastic model of Information Retrieval models aims to predict a frequency **tf** of a term **t** in a document **d**, that is

$$\mathbf{p}(X_{\mathbf{t}} = \mathbf{tf}, \theta_1,...,\theta_k)$$

where $\mathbf{p}$ is the distribution in the parameters $\theta_1,...,\theta_k$. Both the distribution form and the parameters depend on the observed word.

Once the probability of the frequency **tf** of a term **t** in a document **d** is obtained, then it is also possible to obtain the *probability of relevance of a document* **d** given the term frequency $X_{\mathbf{t}}$:

$$\mathbf{p}(\mathbf{d}|X_{\mathbf{t}} = \mathbf{tf}, \theta_1,...,\theta_k)$$

For example, the Two-Poisson assumes that the distribution of a significant word is a mixture of two Poisson models with mean frequencies $\lambda_{\mathbf{t}} \ll \mu_{\mathbf{t}}$, and the estimate of the probability of the document given the term frequency **tf** is:

$$\mathbf{p}(\mathbf{d}|X_{\mathbf{t}} = \mathbf{tf}, \beta_{\mathbf{t}}, \lambda_{\mathbf{t}}, \mu_{\mathbf{t}}) = \frac{1}{1 + \beta_{\mathbf{t}} \cdot e^{\mu_{\mathbf{t}} - \lambda_{\mathbf{t}}} \left(\frac{\lambda_{\mathbf{t}}}{\mu_{\mathbf{t}}}\right)^{\mathbf{tf}}} \quad (1)$$

with $\lambda_{\mathbf{t}} \ll \mu_{\mathbf{t}}$, and where the values of the parameters $\beta_{\mathbf{t}}$, $\lambda_{\mathbf{t}}$ and $\mu_{\mathbf{t}}$ are learned from data and depend on the observed term **t**.

Information Theoretic Models

The probability estimated by information theoretic models is based on the generation of a document as an unordered partition of terms. Each term $\mathbf{t}_i$ of a vocabulary **V** has a prior probability p_i of occurrence in an arbitrary document of the collection, and a document **d** of length k is conceived as a partition satisfying the constraint $\mathbf{tf}_1 + ... + \mathbf{tf}_{\mathbf{V}} = k$ over **V** cells, where $\mathbf{tf}_i$ is the term frequency in the document.

Similarly to the Two-Poisson model the probability of generating a document satisfy the condition

$$\sum_{\mathbf{tf}_i \geq 0} p(X_1 = \mathbf{tf}_1,...,X_{\mathbf{V}} = \mathbf{tf}_{\mathbf{V}}) = 1$$

The connection with Shannon's theory of information is obtained by regarding *a document as a message to transmit*. Shannon explains the notion of information in terms of cost of transmission of a message, which is $-\log_2 p(X_1 = \mathbf{tf}_1,...,X_{\mathbf{V}} = \mathbf{tf}_{\mathbf{V}})$, that is inversely related to the number of choices one has to generate the message with respect to the universe of possible messages: the larger the uncertainty, the larger the information.

If the term independence is assumed, according to which only term frequency counts and positions of terms in documents are irrelevant, a document is treated as an ensemble and is said to be a *bag of words*. In such a case, the probability of generating a document **d** is given by the multinomial distribution:

$$p(X_1 = \mathbf{tf}_1,...,X_{\mathbf{V}} = \mathbf{tf}_{\mathbf{V}}) = \binom{k}{\mathbf{tf}_1 \mathbf{tf}_2 ... \mathbf{tf}_{\mathbf{V}}} p_1^{\mathbf{tf}_1} p_2^{\mathbf{tf}_2} \cdots p_{\mathbf{V}}^{\mathbf{tf}_{\mathbf{V}}}$$

It can be shown that

$$\lim_{k \to \infty} \frac{-\log_2 p(X_1 = \mathbf{tf}_1,...,X_{\mathbf{V}} = \mathbf{tf}_{\mathbf{V}})}{k} = D(p_{ML}||p) \quad (2)$$

where $p^i_{ML} = \frac{\mathbf{tf}_i}{k}$ and $D(p_{ML}||p) = \sum_{\mathbf{t}_i \in \mathbf{V}} p^i_{ML} \log_2 \frac{p^i_{ML}}{p_i}$ is the *Kullback-Leibler divergence*.

Exploiting the additivity of the divergence, one can computes the contribution of the query terms to the document:

$$-\log_2 p(X_1 = \mathbf{tf}_1,...,X_{\mathbf{V}} = \mathbf{tf}_{\mathbf{V}}|\mathbf{q}) = \sum_{\mathbf{t}_i \in \mathbf{q}} \mathbf{tf}_i \log_2 \frac{\mathbf{tf}_i}{k \cdot p_i} \quad (3)$$

Similarly to TF-IDF used in the vector space model, also (3) can be used for retrieval.

Different ways to compute the probability $-\log_2 p(X_1 = \mathbf{tf}_1,...,X_{\mathbf{V}} = \mathbf{tf}_{\mathbf{V}}|\mathbf{q})$ and normalize this information with respect the length of the documents is provided by the Divergence From Randomness models.

Vector Space Models

In the vector space model terms and documents are treated as vectors and relevance document scores are obtained by the similarity function

$$SIM(\vec{\mathbf{q}}, \vec{\mathbf{d}}) = \frac{\vec{\mathbf{q}} \cdot \vec{\mathbf{d}}}{||\vec{\mathbf{q}}|| \cdot ||\vec{\mathbf{d}}||}$$

which is the *cosine* function of the angle between the query-document vectors, where the numerator is the inner product of the two vectors and $||\vec{x}|| = \sqrt{\sum_i x_i^2}$ is the norm of a vector. Any term-document weighting model can be employed in the vector space model, differences depend on the ways queries and documents are represented. The vector space model thus is a way to normalize the term-document weight with respect to the length of the document and this normalization is achieved by the cosine function, that is by dividing the inner product by the norms of the vectors.

Assuming each term weight w_i is the number of the bits necessary to encode the $\mathbf{tf}_i$ occurrences of the term in the document, that is $w_i = -\log_2 p(X_i = \mathbf{tf}_i)$, it is easy to show that the inner product can be rewritten as inversely related to the probability of occurrence of the query-terms in the document. According to the *term independence assumption*, the inner product is inversely related to the product of the probabilities of occurrence of each query-term in the document:

$$\sum_i q_i \cdot w_i = -\log_2 \prod_i p(X_i = \mathbf{tf}_i)^{q_i}$$

If $\lambda_i = \frac{\mathbf{n_i}}{\mathbf{N}}$ is the relative document frequency of a term, that is the ratio of the number $\mathbf{n_t}$ of documents in which the term occurs and the number $\mathbf{N}$ of documents of the collection, then the probability $p(X_i = \mathbf{tf}_i)$ can be given by the *geometric distribution*:

$$p(X_i = \mathbf{tf}_i) = \left(\frac{\lambda_i}{\lambda_i + 1}\right)^{\mathbf{tf}_i}$$

that leads to the representation of a document provided by the the first IR vector space model:

$$\overrightarrow{\mathbf{d}} = \left(\mathbf{tf}_1 \cdot \log_2\left(1 + \frac{\mathbf{N}}{\mathbf{n_{t1}}}\right), \ldots, \mathbf{tf}_n \cdot \log_2\left(1 + \frac{\mathbf{N}}{\mathbf{n_{t}}_n}\right)\right)$$

The TF-IDF model weighting of the vector space model is probabilistic, or information theoretic, in its nature, and the *term independence* is an implicit assumption of the model. An estimate of the value of the document relevance is then obtained by dividing the information by the norms of the two vectors $\overrightarrow{\mathbf{d}}$ and $\overrightarrow{\mathbf{q}}$.

The drawback of this model is the cost of computing document norms. If norms are computed in batch, then a dedicated index must be maintained and updated when documents are added or deleted. On the other hand, if norms are computed online, then one needs to access the direct file to retrieve all the term frequencies $\mathbf{tf}$ of the document.

Bayesian Models

The application of Bayes' theorem to IR concerns the combination of two term frequencies coming from two sources of evidence: the *relative term frequency* p_C *in the collection*, that is regarded as an estimate of the *prior probability* of occurrence of the term in an arbitrary document of the collection, and *term frequency* $p_{ML} = \frac{\mathbf{tf}}{\mathbf{l(d)}}$ *in the document*, that is the *maximum likelihood estimate* of the term in the document.

Bayesian models estimate the probability θ_i of a term $\mathbf{t}$ in a document $\mathbf{d}$, considering the θ_i as parameters of the posterior probability

$$\text{Prob}(\theta_1,\ldots,\theta_n|\mathbf{d}) = \frac{L(\mathbf{d}|\theta_1,\ldots,\theta_n) \cdot \pi(\theta_1,\ldots,\theta_n)}{\text{Prob}(\mathbf{d})}$$

where $L(\mathbf{d}|\,\theta_1,\ldots,\theta_n)$ is the likelihood, $\pi(\theta_1,\ldots,\theta_n)$ is a prior distribution for the parameters θ_i and $\text{Prob}(\mathbf{d}) = \int_0^1 \ldots \int_0^1 L(\mathbf{d}|\,\theta_1,\ldots,\theta_n) \cdot \pi(\theta_1,\ldots,\theta_n) d\theta_1 \ldots \theta_n$.

If a document is regarded as a set of words, so that permutations of sequences of terms from the vocabulary $\mathbf{V}$ have the same probability distribution (*exchangeable sequences*), then the posterior probability has the form of the product of the multinomial distribution and of a unique prior distribution π:

$$\text{Prob}_{\mathbf{d}}(\theta_1,\ldots,\theta_n|\mathbf{d}) = \frac{\binom{\mathbf{l(d)}}{\mathbf{tf}_1\mathbf{tf}_2\ldots\mathbf{tf}_n} \theta_1^{\mathbf{tf}_1}\theta_2^{\mathbf{tf}_2} \ldots \theta_n^{\mathbf{tf}_n} \cdot \pi(\theta_1,\ldots,\theta_n)}{\text{Prob}(\mathbf{d})}$$

In Bayesian methodology, the prior is chosen in order to have both prior and posterior distribution with the same functional form (*conjugate prior*). The conjugate form to the multinomial distribution is given by the *Dirichlet's distribution*, that is the posterior probability is:

$$\text{Prob}_{\mathbf{d}}(\theta_1,\ldots,\theta_n|\mathbf{d}) \propto \theta_1^{\mathbf{tf}_1+A_1-1}\theta_2^{\mathbf{tf}_2+A_2-1} \ldots \theta_n^{\mathbf{tf}_n+A_n-1} \quad (4)$$

where $\sum_i A_i = A$ are the parameters. The *expectations* $\hat{\theta}_i$ of variables θ_i comes to be:

$$\hat{\theta}_i = \frac{\mathbf{tf}_i + A_i}{\sum_i(\mathbf{tf}_i + A_i)} = \frac{\mathbf{tf}_i + A_i}{\mathbf{l(d)} + A}$$

The Bayesian method constitutes the bulk of the language modeling approach to IR [2,4] (see *language modeling*). However, the values for the parameters A_i need to be learned. The parameters A_i can be further reduced to a single parameter μ by setting the values of A_i to the expected term frequency of $\mathbf{t}_i$ in a document of a fixed length μ, that is $A_i = \mu \cdot p_i$, where p_i is $\frac{\mathbf{TF}}{\mathbf{TFC}}$ is the frequency in the collection (and thus $\mu = \sum_i A_i$). The expectation of (4) thus smoothes the maximum likelihood estimate (MLE) $p_{ML} = \frac{\mathbf{tf}}{\mathbf{l(d)}}$ with the term frequency in the collection, p_i, that is:

$$\hat{\theta}_i = \frac{\mathbf{tf}_i + \mu \cdot p_i}{\mathbf{l(d)} + \mu}$$

The value of the parameter μ depends on several factors, but mainly on the collection and the query length. Another way of smoothing the raw MLE $p_{ML} = \frac{\mathbf{tf}}{\mathbf{l(d)}}$ of the likelihood is the Mercer-Jelinek smoothing technique for IR, that mixes linearly the two frequencies p_{ML} and p_i:

$$\lambda \cdot p_{ML} + (1 - \lambda) \cdot p_i$$

with $0 \leq \lambda \leq 1$.

Probabilistic and Binary Retrieval Models

The independence binary retrieval model (BIR) is based on the assumption that relevance is an event *rel* of the probabilistic space and that relevance can be learned directly from user's feedback. The data containing relevance information are initially gathered by sampling or by assessment on a first pass retrieval, then relevance information is processed to set the values of certain parameters associated with the BIR matching function. The BM25 model also derives from the BIR model.

The BIR model is built upon four probabilities $p_{\mathbf{q}}(\mathbf{t} = a | \, rel = b)$, where the subscript $\mathbf{q}$ denotes that relevance depends on a specific query, and $a, b \in \{0, 1\}$ are the boolean values *false* and *true* for the variables $\mathbf{t}$ (the term occurs in the documents or not) and for the variable *relevance* (the document is relevant or not) respectively. For the estimation of probability of document relevance BIR uses Bayes' Theorem, which relates the posterior probability of relevance ($p_{\mathbf{q}}(rel = b | \, \mathbf{t} = a)$) to the prior probability of relevance ($p_{\mathbf{q}}(rel = b)$) and the likelihood of relevance after observing a document ($p_{\mathbf{q}}(\mathbf{t} = a | \, rel = b)$).

Logical and Algebraic Models

Other approaches to IR modeling include those based on logics, fuzzysets, Bayesian inference networks and the Dempster-Shafer theory of evidence. All these models share the view that (stochastic and/or logical) dependency between sets of words can be captured by a probability defined on top of a (graphical or logical) link between boolean propositional sentences.

A Bayesian inference network is a directed, acyclic dependency graph (DAG) in which nodes represent propositions andedges represent dependence relations between these propositions [10].

Fuzzy models are based on fuzzy-set theory, whichdefines partial membership of elements to a set. Fuzzy models extend logical operators with partial set membership, and processes user queries in a similar way to the conventional Booleanmodel.

A common generalization of both Boolean and fuzzy model is Salton, Fox and Wu's Extended Boolean Information Retrieval Model [9], where the query-document similarity is defined in the L_p-spaces (p-norms).

The logical approach to IR is motivated by modelling retrieval as an inference process. The foundational assumption of logical models is that the semantics of the content of a document is related to a query through a conditional connective, which consists of a type of logical implication set out by Van Rijsbergen in 1986 [11]. One of the IR models based on probabilities on conditionals is obtained by a probability revision method called *imaging* [3].

Key Applications

IR models are applied to a wide range of fields and domains: digital libraries, biomedicine, web search engines, enterprise search, desktop search, search engines for blogs, for legal and other vertical domains, recommendation systems for content providers, information filtering and routing for the selective delivery of news or for detecting spamming and junk mails.

Future Directions

With the dramatic growth of document sources information needs to be retrieved in a distilled manner. The most popular search algorithms for WEB, like PageRank or HITS, clearly indicate that document quality and authoritative content depend on social and collaborative aspects of the community of the users. Users will more and more cite their favorite information sources, express and share their opinions with other in the same network of interests or for entertainment reasons.

Information Retrieval Models will thus naturally evolve along to dynamic search dimensions such as analysis of the social networks, time and context. Contexts include background knowledge about the domain, group of interests, relevant information sources and specific ontologies, as well as profiles with users background, interests, and preferences.

One of the most important research direction is the development of systems that integrates database and IR technology. Such an integration requires new query language, new indexing techniques but more importantly models that combines exact matching, proper of the database systems processing queries for

structured information, with partial matching, proper of the IR systems.

Experimental Results

The performance measures of the IR models are based on precision and recall. Significance tests on a set of queries must be conducted to assess the relative performance of models.

Data Sets

Most of data sets and test collections can be found from the TREC (NIST) web site (http://trec.nist.gov/data.html).

Cross-references

- ▶ Digital Libraries
- ▶ Indexing
- ▶ Information Retrieval Evaluation Measures
- ▶ Text Indexing Techniques
- ▶ Text Mining
- ▶ Web Search and Crawling

Recommended Reading

1. Amati G. and Van Rijsbergen C.J. Probabilistic models of information retrieval based on measuring the divergence from randomness. ACM Trans. Inform. Syst., 20(4):357–389, 2002.
2. Berger A. and Lafferty J. Information retrieval as statistical translation. In Proc. 22nd Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 1999, pp. 222–229.
3. Crestani F. and Van Rijsbergen C.J. Probability kinematics in information retrieval. In Proc. 18th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 1995, pp. 291–299.
4. Croft W.B. and Lafferty J. (eds.). Language Modeling for Information Retrieval. Kluwer Academic, 2003.
5. Harter S.P. A probabilistic approach to automatic keyword indexing. part I: On the distribution of specialty words in a technical literature J. ASIS, 26:197–216, 1975.
6. Ponte J. and Croft B. A language modeling approach in information retrieval. In Proc. 21st ACM SIGIR Conference on Research and Development in Information Retrieval, B. Croft, A. Moffat, and C.J. Van Rijsbergen (eds.). ACM, Melbourne, Australia, 1998, pp. 275–281.
7. Robertson S.E. and Walker S. Some simple approximations to the 2-Poisson model for probabilistic weighted retrieval. In Proc. 17th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval. Springer, Dublin, Ireland, June 1994, pp. 232–241.
8. Salton G. and McGill M.J. Introduction to Modern Information Retrieval. McGraw–Hill, New York, 1983.
9. Salton G., Fox E.A., and Wu H. Extended boolean information retrieval. Commun. ACM, 26(11):1022–1036, 1983.
10. Turtle H. and Bruce Croft W. Evaluation of an inference network-based retrieval model. ACM Trans. Inform. Syst., 9(3):187–222, 1991.
11. Van Rijsbergen C.J. A new theorethical framework for information retrieval. In Proc. 9th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 1986, pp. 194–200.

Information Retrieval Models/Metrics/Operations

▶ Biomedical Scientific Textual Data Types and Processing

Information Retrieval Operations

Edie Rasmussen
University of British Columbia, Vancouver, BC, Canada

Synonyms

Information retrieval processing

Definition

An information retrieval system for collections of unstructured text can be viewed as a set of processing modules, beginning with lexical analysis of documents and leading ultimately to a retrieval process in which user queries are matched against documents. The processes which make up an information retrieval system are referred to as information retrieval operations. In some cases, the end goal is not the retrieval of matching documents from a collection but an auxiliary application, such as document categorization, summarization, or filtering of information from an information stream. While the basic operations in information retrieval systems are similar, there can be considerable variability in the model or details of implementation, making evaluation of information retrieval system effectiveness an important step.

Historical Background

Early work in information retrieval began in the 1950s with the recognition that words in documents could be used as indicators of the content or meaning of the document. Hans Peter Luhn, in early work on

automatic abstracting and Keyword-in-Context indexing, recognized that words had differential value in representing the document's meaning [4,5], a concept that is fundamental to modern information retrieval systems. In the 1960s, Salton [7] and his research group developed the SMART system based on the vector space model, combining the basic operations for information retrieval in an information retrieval system. This allowed them to experiment with a variety of information retrieval operations, including stemming, term weighting, similarity functions, and relevance feedback.

Subsequent research has developed and refined information retrieval operations, and there has been increased emphasis on modular implementations which allow specific operations to be isolated. In 1992, Frakes and Baeza-Yates edited Information Retrieval: Data Structures and Algorithms [3], with the goal of making information retrieval more widely known, and making available pseudo-code and code for the basic information retrieval operations. More recently, open-source software such as the Lemur Toolkit has provided easy access to information retrieval systems which allow the isolation and modification of information retrieval operations [1].

Foundations

The basic operations in the building of an information retrieval system are the pre-processing of text, creation of index files, and processing of queries to provide a (usually) ranked list of potentially relevant documents.

Pre-processing of text involves parsing the text to create a list of index terms (lexicon) which will be stored for retrieval. In identifying the index terms, some words may be identified as Stopwords, i.e., common words that are not considered meaningful, and which are not indexed. Words with common suffixes may be identified through an operation called Stemming, and concatenated under a single index entry. While it is possible to select a subset of the words in a text as index terms, usually all words (except words identified as stopwords) in the text are used, so that this is sometimes referred to as a "bag of words" approach.

Early research demonstrated the value of using differential weights for the index terms associated with a document, and testing different models for assigning term weights continues to be an active research area. A long-standing approach was the tf*idf, or term frequency-inverse document frequency weighting scheme, in which every term in a document was assigned a weight proportional to its frequency in that document, and inversely proportional to its frequency in the collection as a whole. More recent weighting schemes include BM25 and weights derived from language models [6].

The resultant document-term matrix is sparse and it is not efficient for storage and processing, so the next set of operations involves building the index to store the terms and associated term weights. The inverted index, in which a postings list of document numbers and associated weights is given for each term, is commonly used. For large data collections, some form of index compression may also be applied [6,10].

Finally, a set of operations on the query results in delivery of the output to the user of the system. A query is presented to the system through a user interface, and it undergoes the same processing as the document collection to identify the index terms which it contains. The order in which query terms are processed may be considered to achieve efficiencies in processing. The query is matched against the documents in the collection using some similarity function, and the documents are ranked in order of similarity. A variety of similarity functions have been proposed and tested; the choice may depend on the underlying information retrieval model. For example, in the vector space model, the similarity function was usually the cosine function which measures the angle between the document and query vector.

Further operations may be carried out on the query in an attempt to improve it. In relevance feedback, information about documents in the ranked list which are either known to be relevant (from user input) or assumed to be relevant, is used to reweight the query terms and potentially offers performance improvement. In query expansion, terms are added to the query, automatically or with the assistance of the searcher, in an attempt to improve retrieval performance.

The retrieval process described above performs what is sometimes referred to as the ad hoc retrieval task, in which the query is matched against a relatively static collection of documents. In some situations it is the query which is (relatively) static, and the documents which are changing. Matching a query against a document stream is referred to as information filtering, and is useful in situations where information which is continually being created must be monitored (for example, from a newswire).

In some instances the end goal is not to produce documents in response to a query, but to perform

some other task through specialized processing of the indexed document collection, often in support of information retrieval. An example is document categorization, whereby documents being added to a collection are automatically assigned to what is predicted to be the most appropriate category for them, based on characteristics which have been observed for that category (such as patterns of term occurrence). In document clustering, document-document similarity based on term occurrence and term weights is used to divide a document collection into clusters or groupings of like objects. (Clustering differs from categorization in that the clusters are not a priori known.) Clusters may be created from retrieval output, in order to provide the user with documents which are organized by topic. Similarly, in document summarization, information about the document is used to identify the most important words, sentences or concepts it contains in order to build a summary of it, for instance, to use as a document surrogate to present to the searcher. Another post-retrieval operation is information visualization [11], in which a visual display is generated rather than, or in addition to, a ranked list. In some cases the visual display can be manipulated to provide further interpretation of the retrieval results.

Experimental Results

Almost as soon as computerized information retrieval systems were conceived, evaluation was seen as an important component. The best known of the early tests were the Cranfield experiments [8], so-called because they were conducted by Cyril Cleverdon and a group of researchers at the Cranfield College of Aeronautics, primarily as a test of indexing techniques. The first set of experiments, conducted in 1958–1962, provided controversial results, and in response to criticism of the methodology, a second set of experiments was devised by Cleverdon, with emphasis on rigor and a laboratory model. These second experiments, known as Cranfield II, led to the basic model for information retrieval experimentation in common use today: a document collection, a set of queries and associated relevance judgments, and measurement based on precision and recall. In terms of findings, the Cranfield experiments and a series of experiments on the SMART system [8] showed that the interest of the time in complex indexing systems was misguided, and that in general simpler indexing systems worked as well as more complex techniques. The laboratory model as exemplified by the Cranfield experiments made it possible to isolate individual information retrieval operations and evaluate performance with, for instance, different values or functions.

Through the 1970s and 1980s, the laboratory for information retrieval research expanded, with new test collections, query sets and relevance judgments. The collections grew steadily larger, though still falling far short of those found in operational systems. After 30 years of IR experimentation in the Cranfield model, there was confidence within the IR community that the basic information retrieval operations had been refined to achieve real performance improvements, although the transfer of the technology to the commercial section was extremely limited.

One of the strongest arguments for the lack of commercial success was skepticism about the scalability of performance improvements from the laboratory to large scale systems. Partly in response to this criticism, in 1992 the National Institute of Standards and Technology (NIST) hosted the first Text REtrieval Conference (TREC) [9]. TREC provided the infrastructure for large-scale IR evaluation, and resulted in the improvement and dissemination of many information retrieval operations, notably term weighting, as reports of the success of the BM25 model for term weighting were widely disseminated. TREC also supported the development of standard routines to analyze the results of query runs, minimizing the variance in analysis of data that existed, and making it easier to compare variations in information retrieval operations.

Cross-references

- ▶ Clustering for Post Hoc Information Retrieval
- ▶ Index Creation and File Structures
- ▶ Lexical Analysis of Textual Data
- ▶ Query Expansion for Information Retrieval
- ▶ Relevance Feedback
- ▶ Similarity and Ranking Operations
- ▶ Stemming
- ▶ Stopwords
- ▶ Summarization
- ▶ Visualization for Information Retrieval

Recommended Reading

1. Eckard E. and Chappelier J.-C. Free software for research in information retrieval and textual clustering, 2007. Available at: infoscience.epfl.ch/record/115460/files/Free_sofware_for_IR.pdf

2. Frakes W.B. and Baeza-Yates R. Information Retrieval: Data Structures and Algorithms. Englewood Cliffs, Prentice Hall, NJ, 1992.
3. Korfhage R.R. Information Storage and Retrieval. Wiley, New York, 1997.
4. Luhn H.P. The automatic creation of literature abstracts. IBM J. Res. Dev., 2:157–165, 1958. Available at: http://www.research.ibm.com/journal/rd/022/luhn.pdf
5. Luhn H.P. Keyword-in-context index for technical literature. Am. Documentation, 11(4):288 (8p), 1960.
6. Manning C.D., Raghavan P., and Schütze H. Introduction to Information Retrieval. Cambridge University Press, Cambridge, UK, 2008.
7. Salton G. The SMART Retrieval System: Experiments in Automatic Document Processing. Prentice-Hall, Englewood Cliffs, NJ, 1971.
8. Sparck Jones K. Retrieval system tests 1958–1978, In: Information Retrieval Experiment, K. Sparck Jones, (ed.). Butterworths, London, 1981. pp. 213–255.
9. Voorhees E.M. and Harman D.K. (eds.). TREC: Experiment and evaluation in Information Retrieval. MIT Press, Cambridge, MA, 2005.
10. Witten I.H., Moffat A., and Bell T.C. Managing Gigabytes: Compressing and Indexing Documents and Images (2nd edn.). Morgan Kaufmann, San Francisco, CA, 1999.
11. Zhang J. Visualization for Information Retrieval. Springer, New York, 2008.

Information Retrieval Processing

► Information Retrieval Operations

Information Seeking

► Information Foraging

Information Visualization

► Data Visualization

Information visualization on hierarchies

► Visualizing Hierarchical Data

Information Visualization on Networks

► Visualizing Network Data

INitiative for the Evaluation of XML Retrieval

Gabriella Kazai
Microsoft Research Cambridge, Cambridge, UK

Synonyms

INEX; XML information retrieval; Evaluation forum

Definition

The INitiative for the Evaluation of XML retrieval (INEX), launched in 2002, is an established evaluation forum for XML Information Retrieval (IR) with over 90 participating organizations worldwide. The initiative is sponsored by the DELOS Network of Excellence for Digital Libraries and supported by the IEEE Computer Society.

INEX encourages research in XML IR by providing an infrastructure to evaluate the effectiveness of XML IR systems. The infrastructure takes the form of a large XML test collection, appropriate scoring methods, and a forum for participating organizations to compare their results. The construction of the test collection is a collaborative effort, where participating organizations contribute by providing test queries and relevance judgments on the collection of XML documents. The constructed test collection provides participants a means for comparative and quantitative experiments.

Historical Background

The motivation for INEX stems from the increasingly important role of XML IR in many information access systems (e.g., in digital libraries and on the Web) where content is a mixture of text, multimedia, and metadata, formatted according to the adopted W3C standard of the eXtensible Markup Language (XML).

XML offers the opportunity to exploit the internal structure of documents in order to allow for more precise access, providing more focused answers to users' requests. XML IR thus breaks away from the traditional retrieval unit of a document as a single

large (text) block and aims to implement more focused retrieval strategies that return document components, i.e., XML elements, instead of whole documents in response to a user query. This focused retrieval approach is seen of particular benefit for information repositories containing long documents, or documents covering a wide variety of topics (e.g., books, user manuals, legal documents), where the user's effort to locate relevant content can be reduced by directing them to the most relevant parts of the documents. Providing effective access to XML based content is therefore a key issue for the success of these systems.

INEX aims to provide the means necessary for the evaluation of XML IR systems. It follows the predominant approach in IR of evaluating retrieval effectiveness using a test collection constructed specifically for that purpose.

A test collection usually consists of a set of documents, user requests (topics), and relevance assessments which specify the set of "right answers" for the requests.

In the field of IR, there have been several large-scale evaluation projects, including the Text REtrieval Conference (TREC) (http://trec.nist.gov/), the Cross-Language Evaluation Forum (CLEF) (http://www.clef-campaign.org/), and the National Institute of Informatics Test Collection for IR Systems (NTCIR) (http://research.nii.ac.jp/ntcir/), which resulted in established test collections and evaluation methodologies. These traditional IR test collections and methodology, however, cannot be directly applied to the evaluation of content-oriented XML retrieval as they do not consider structural aspects and, for example, provide relevance judgments only at the unit of the document. Furthermore, the evaluation is based on assumptions that do not hold in XML IR. For example, the evaluation of IR systems treats documents as independent and well-distinguishable separate units of approximately equal size. XML IR, however, allows for varying sized document components to be retrieved. It is also possible that multiple elements from the same document are retrieved, which cannot be viewed as independent units.

The evaluation of XML retrieval systems thus makes it necessary to build test collections and develop appropriate metrics where the evaluation paradigms are provided according to criteria that take into account the imposed structural aspects. INEX aims to address these goals.

Foundations

Since its launch in 2002, INEX has grown both in terms of number of participants and with respect to its coverage of the investigated retrieval tasks. Throughout the years, INEX faced a range of challenges regarding the evaluation of XML IR approaches. These include the question of suitable relevance criteria, feasible assessment procedures, and appropriate evaluation measures. Different theories and methods for the evaluation of XML IR were developed and tested at INEX, leading to a now stable evaluation setup and a rich history of learned lessons.

In 2002, INEX started with 36 active participating organizations and a small collection of XML documents donated by the IEEE Computer Society, totaling 494 MB in size and containing over eight million XML elements [3, pp. 1–17]. INEX 2002 run a single track, investigating ad hoc retrieval applied to XML documents based on the focused retrieval approach. In IR literature, ad hoc retrieval is described as a simulation of how a library might be used, and it involves the searching of a static set of documents using a new set of topics [15]. While the principle is the same, the difference for INEX is that the library consists of XML documents, the queries may contain both content and structural conditions and, in response to a query, arbitrary XML elements may be retrieved from the library.

Two subtasks were defined within the Ad hoc track based on the query types of Content-Only (CO) and Content-And-Structure (CAS). In the CO task, it was left entirely to the retrieval system to identify the most appropriate relevant XML elements to return to the user, while in the CAS task, systems could make use of the structural clues specified by the user in the query.

The queries were created by the participating groups, contributing 30 CO and 30 CAS topics to the test collection. For each topic, a title, a description and a narrative were specified, where the syntax of the title allowed the definition of target elements and so-called containment conditions (content word and containment element pairs).

The relevance criteria was defined along two dimensions, each with four possible grades for assessors to choose from. Assessors were asked to assign scores for both dimensions to all XML elements of the collection that contained relevant information.

Based on the collected relevance judgments, effectiveness scores were calculated by adopting Raghavan's

Precall measure [3, pp. 1–17]. Table 1 shows summary information on INEX 2002.

In 2003, the CAS subtask was separated into the Strict CAS (SCAS) and the Vague CAS (VCAS) strands so that the effect of interpreting a query's structural clues strictly or vaguely could be studied. The CO subtask remained unchanged, and so did the document collection.

The syntax of the topic title was modified based on the XPath standard (http://www.w3.org/TR/xpath), where a new *about()* function was introduced [4, pp. 192–199].

Due to the fact that the coverage dimension of the relevance criterion was found to be susceptible to misinterpretation, INEX 2003 renamed and redefined the relevance dimensions to Exhaustivity and Specificity.

A new measure, inex_eval_ng [10], was also introduced, allowing to take into account the possible overlap (e.g., paragraph and its container section) and the varying size of the XML elements. Table 2 shows summary information on INEX 2003.

By 2004, INEX had 43 active participating groups and four additional tracks: Relevance feedback, Heterogeneous collection, Natural language processing (NLP) and Interactive tracks. The ad hoc track ran only two of the subtasks defined in 2003: CO and VCAS. The format of a topic's title field was formally defined using the Narrowed Extended XPath I (NEXI) language. The purpose of the Relevance feedback track was to explore issues related to the use of relevance feedback in a structured environment [1]. The Heterogeneous track aimed to address challenges where collections of XML documents from different sources and with different DTDs or Schemas were to be searched. The NLP track investigated whether it was practical to use a natural language query in place of the formal NEXI topic title used in the Ad hoc track [9]. The Interactive track focused on studying the behavior of searchers when presented with components of XML documents that have a high probability of being relevant (as estimated by an XML IR system) [12].

Both the document collection and the relevance dimensions remained unchanged from 2003. The measure of Precall was used as in previous years to report the retrieval effectiveness scores of the participating search systems. Table 3 shows summary information on INEX 2004.

The ad hoc track at INEX 2005 continued studying the role of structure in user queries, and defined four separate strands of the VCAS subtask based on the strict

INitiative for the Evaluation of XML Retrieval. Table 1. Summary information on INEX 2002

INEX 2002	
Organizers:	N. Fuhr, N. Gövert, G. Kazai, M. Lalmas
Participants:	36 active (49 from 21 countries signed up)
Tracks:	Ad hoc retrieval with two subtasks
Ad hoc tasks:	Ad hoc retrieval based on CO (content-only) and CAS (content-and-structure) topics
Document corpus:	12,107 articles from IEEE Computer Society, 1995–2002
	Totaling 494 MB and over eight million XML elements
Topics:	30 CO and 30 CAS topics
Relevance:	Topical relevance (Irrelevant, Marginal, Fairly, Highly relevant),
	Component coverage (No coverage, Too small, Too large, Exact)
Metrics:	Precall (inex_eval) [3, pp. 1–17]
Proceedings:	[6]

INitiative for the Evaluation of XML Retrieval. Table 2. Summary information on INEX 2003

INEX 2003	
Organizers:	N. Fuhr, M. Lalmas
Participants:	30 active (40 from 18 countries signed up)
Tracks:	Ad hoc retrieval with three subtasks
Ad hoc tasks:	CO (content-only), SCAS (strict content-and-structure) and VCAS (vague content-and-structure)
Document corpus:	Same as in 2002
Topics:	36 CO and 30 CAS topics
Relevance:	Exhaustivity (Not, Marginally, Fairly, Highly exhaustive),
	Specificity (Not, Marginally, Fairly, Highly specific)
Metrics:	Precall (inex_eval) [3, pp. 1–17], and inex_eval_ng [10]
Proceedings:	[4]

INitiative for the Evaluation of XML Retrieval. Table 3. Summary information on INEX 2004

INEX 2004	
Organizers:	Overall: N. Fuhr, M. Lalmas
	Topic format: B. Sigurbjörnsson, A. Trotman
	Relevance assessment tool: B. Piwowarski
	Metrics: G. Kazai, A.P. de Vries
Participants:	43 active (55 from 20 countries signed up)
Tracks:	Ad hoc retrieval with two subtasks
	Relevance feedback track (C. Crouch, M. Lalmas)
	Heterogeneous collection track (T. Rölleke, Z. Szlávik)
	Natural language processing track (S. Geva, T. Sahama)
	Interactive track (A. Tombros, B. Larsen, S. Malik)
Ad hoc tasks:	CO (content-only) and VCAS (vague content-and-structure)
Document corpus:	Same as in 2002 and 2003
Topics:	40 CO and 35 CAS topics
Relevance:	Same as in 2003
Metrics:	Precall (inex_eval) [3, pp. 1–17]
Proceedings:	[6]

or vague interpretations of the structural conditions of a query [13]. The CO subtask has also diversified into six strands based on a combination of three subtasks (Focused, Thorough and FetchBrowse) and the use of the CO or CO+S (CO+Structure) type topics. The latter type expands the CO topic format with an additional CAS title field, where structural hints for the CO title can be expressed. The Focused task asked systems to return a ranked list of the most focused (specific and exhaustive) document parts, without returning overlapping elements. The Thorough task required systems to estimate the relevance of all XML elements in the searched collection and return a ranked list of the top 1,500 elements. The FetchBrowse task asked systems to return to the user the most focused, relevant XML elements clustered by the unit of the document containing the elements. Put another way, the task was to return documents with the most focused relevant elements highlighted within them.

All additional tracks started in 2004 run again in 2005, together with the new Document mining and Multimedia tracks. The Document mining track focused on the tasks of classification and clustering by developing methods that are able to exploit the XML markup for this purpose [2]. The Multimedia track was set up aiming at the evaluation of structured document retrieval approaches which are able to combine the relevance of the different media types into a single (meaningful) ranking that is presented to the user [14].

INEX 2005 obtained additional resources in the form of additional XML articles from the IEEE Computer Society, increasing the total size of the collection to 764 MB. In addition, the Multimedia track made use of an XML version of the Lonely Planet collection.

Other changes to the evaluation framework included new assessment procedures and new metrics. The assessment process was simplified to asking assessors to first highlight relevant passages and then assess the elements overlapping these passages. As a consequence, the Specificity dimension could be automatically measured on a continuous scale [0,1] by calculating the ratio (in characters) of the highlighted text (i.e., relevant information) and the total length of the element.

To report effectiveness scores, INEX 2005 adopted the eXtended Cumulated Gain (XCG) measures [5, pp. 16–29], which were developed specifically for graded (non-binary) relevance values and with the aim to allow XML IR systems to be credited according to the retrieved elements' degree of relevance. Table 4 shows summary information on INEX 2005.

In 2006, INEX finished with 50 active participating organizations and expanded to a total of nine tracks: Ad hoc, Relevance feedback, Heterogeneous collection, Natural language processing, Interactive, Multimedia, Document mining, Use case, and Entity ranking tracks. The Ad hoc track consisted of four subtasks: Focused, Thorough, Relevant in Context (FetchBrowse in 2005), and Best in Context tasks. The new Best in Context task asked systems to return a single best entry point (BEP) to the user per relevant document. Rather than dealing with information access to XML elements, the new Entity ranking track set as its task the retrieval of a list of entities of specific types (e.g., people, products, artifacts). The Use case track attempted to identify examples of how XML IR systems can be exploited by end-users for various purposes [11].

A major change in 2006 was the departure from the use of the IEEE document collection, which has been replaced by a collection of XML articles from the Wikipedia project.

INEX 2006 has also further simplified the assessment procedure by dropping the exhaustivity dimension of the relevance criteria.

INitiative for the Evaluation of XML Retrieval. Table 4. Summary information on INEX 2005

INEX 2005	
Organizers:	Overall: N. Fuhr, M. Lalmas
	Topic format: B. Sigurbjörnsson, A. Trotman
	Relevance assessment tool: B. Piwowarski
	Metrics: G. Kazai, A.P. de Vries, P. Ogilvie, B. Piwowarski
Participants:	41 active (47 signed up)
Tracks:	Ad hoc retrieval with ten subtasks
	Relevance feedback track (Y. Mass, C. Crouch)
	Heterogeneous collection track (R. Larson)
	Natural language processing track (S. Geva, T. Sahama)
	Interactive track (B. Larsen, A. Tombros, S. Malik)
	Multimedia track (R. van Zwol, G. Kazai, M. Lalmas)
	Document mining track (L. Denoyer, A-M. Vercoustre, P. Gallinari)
Ad hoc tasks:	CO.Focused, CO.Thorough, CO. FetchBrowse,
	COS.Focused, COS.Thorough, and COS.FetchBrowse,
	VVCAS (vague target and vague containment CAS),
	SVCAS (strict target and vague containment CAS),
	VSCAS (vague target and strict containment CAS), and
	SSCAS (strict target and strict containment CAS)
Document corpus:	16,819 articles from IEEE Computer Society, 1995–2004
	Totaling 764 MB, and over 11 million XML elements
	Includes an additional 4,712 new articles
Topics:	40 CO+S and 47 CAS topics
Relevance:	Exhaustivity (Not, Somewhat, Highly exh., Too small),
	Specificity (Continuous scale)
Metrics:	XCG metrics [6, pp. 16–29]
Proceedings:	[6]

A new passage-based recall and precision was adopted to report effectiveness scores for the Relevant in Context task, while XCG was employed for the Focused and Thorough tasks [7, pp. 20–34]. Two further measures, BEP-distance and EPRUM [7, pp. 20–34], provided the performance results for the Best in Context task. Table 5 shows summary information on INEX 2006.

INitiative for the Evaluation of XML Retrieval. Table 5. Summary information on INEX 2006

INEX 2006	
Organizers:	Overall: N. Fuhr, M. Lalmas
	Wikipedia collection: L. Denoyer, M. Theobald
	Topic format: A. Trotman, B.Larsen
	Task description: J. Kamps, C. Clarke
	Relevance assessment tool: B. Piwowarski
	Metrics: G. Kazai, S. Robertson, P. Ogilvie
Participants:	50 active (68 signed up)
Tracks:	Ad hoc retrieval with four subtasks
	Relevance feedback track (Y. Mass, R. Schenkel)
	Heterogeneous collection track (I. Frommholz, R. Larson)
	Natural language processing track (S. Geva, X. Tannier)
	Interactive track (B. Larsen, A. Tombros, S. Malik)
	Multimedia track (R. van Zwol, T. Westerveld)
	Document mining track (L. Denoyer, A-M. Vercoustre, P. Gallinari)
	Use case track (A. Trotman, N. Pharo)
	Entity ranking track (A.P. de Vries, N. Craswell)
Ad hoc tasks:	Focused, Thorough, Relevant in Context, and
	Best in Context
Document corpus:	659,388 articles of the Wikipedia project, covering a hierarchy of 113,483 categories, totaling over 60 GB (4.6 GB without images) and 30 million XML elements
Topics:	125 CO+S topics
Relevance:	Specificity (Continuous scale)
Metrics:	XCG metrics [6, pp. 16–29], passage-based recall and precision,
	BEP-distance and EPRUM [7, pp. 20–34]
Proceedings:	[7]

For INEX 2007, 100 groups registered to participate. A total of six track were run in 2007: The Ad hoc, Document mining, Multimedia, and Entity ranking tracks were continued, and two new tracks were started: Link the Wiki, and Book search. The Ad hoc track pitted XML element retrieval approaches against passage retrieval methods on three tasks: Focused, Relevant in Context and Best in Context. The Link the Wiki track aims at evaluating the state of the art in automated discovery of document hyperlinks. The Book search track builds on a collection of over 40,000 digitized books, marked up in XML. It aims to investigate book-specific relevance ranking strategies, user interface issues and user behavior, exploiting special features, such as back of book indexes provided by authors, and linking to associated metadata like catalogue information from libraries.

There were no changes in the document collection, which remained the Wikipedia XML corpus, and the relevance assessment criteria and procedures. The metrics from 2006 have been refined to allow for the evaluation of arbitrary passages as retrieval results and a new measure generalized precision and recall has also been introduced to measure retrieval effectiveness for the Relevant in Context task [8, INEX 2007 Evaluation Measures]. Table 6 shows summary information on INEX 2007.

INEX 2008 is to set to start in the Spring of 2008.

INitiative for the Evaluation of XML Retrieval. Table 6. Summary information on INEX 2007

INEX 2007	
Organizers:	Overall: N. Fuhr, A. Trotman, M. Lalmas
	Wikipedia collection: L. Denoyer
	Collection exploration: R. Schenkel, M. Theobald
	Topic format: B.Larsen, A. Trotman
	Task description: J. Kamps, C. Clarke
	Relevance assessment tool: B. Piwowarski
	Metrics: G. Kazai, B. Piwowarski, J. Kamps, J. Pehcevski, S. Robertson, P. Ogilvie
Participants:	100 signed up
Tracks:	Ad hoc retrieval with six subtasks
	Document mining track (L. Denoyer, P. Gallinari)
	Multimedia track (T. Westerveld, T. Tsikrika)
	Entity ranking track (A.P. de Vries, N. Craswell, M. Lalmas, J.A. Thom, A-M. Vercoustre)
	Link the Wiki track (S. Geva, A.Trotman)
	Book search track (G. Kazai, A. Doucet)
Ad hoc tasks:	Focused, Relevant in Context, and Best in Context using either XML element retrieval or passage retrieval approaches
Document corpus:	Same as in 2006
Topics:	130 CO+S topics
Relevance:	Same as in 2006
Metrics:	Passage-based recall and precision, generalized precision and recall, and BEP-distance [8, INEX 2007 Evaluation Measures]
Proceedings:	[8]

Key Applications

Evaluation is a key component of any system development as it allows to quantify improvement in performance. INEX provides an important resource to facilitate the evaluation of XML IR systems.

XML IR is a form of semi-structured text retrieval, which aims to exploit the inherent structure of documents to improve their retrieval, where the structure is given by the XML markup.

Some of the issues and proposed solutions within INEX are applicable to other areas of IR, such as passage, video and Web retrieval, where there is no fixed unit of retrieval and where the evaluation needs to handle overlapping fragments and users' post query browsing behavior.

Data Sets

Until 2004, the document collection consisted of 12,107 articles, marked-up in XML, from 12 magazines and 6 transactions of the IEEE Computer Society's publications, covering the period of 1995–2002, and totaling 494 MB in size, consisting of over eight million XML elements. On average, an article contains 1,532 XML nodes, where the average depth of the node is 6.9.

In 2005, the collection was extended with further publications from the IEEE Computer Society. A total of 4,712 new articles from the period of 2002–2004 were added, giving a total of 16,819 articles, and totaling 764 MB in size and over 11 million XML elements.

The overall structure of a typical article in the IEEE collection consists of a front matter, a body, and a back matter. The front matter contains an article's metadata, such as title, author, publication information, and

abstract. The article's body contains the actual content of the article. The body is structured into sections, subsections, and sub-subsections. These logical units start with a section title, followed by a number of paragraphs. In addition, the content has markup for references (citations, tables, figures), item lists, and layout (such as emphasized and bold faced text), etc. The back matter contains a bibliography and further information about the authors.

INEX 2006 and 2007 switched to a different document collection, consisting of 659,388 English articles, marked-up in XML, from the Wikipedia (http://en.wikipedia.org) project, totaling over 60 GB (4.6 GB without images) and 30 million XML elements. The collection's structure is similar to that of the IEEE collection's. On average, a Wikipedia article contains 161.35 XML nodes, where the average depth of an element is 6.72.

In addition to these, the different tracks worked with additional document collections. For example, the Multimedia track in 2005 made use of an XML version of the Lonely Planet collection and the Book Search track in 2007 provided a collection of 42,000 digitized books marked up in XML.

URL to Code

http://inex.is.informatik.uni-duisburg.de/

Cross-references

- ▶ Content-and-Structure Query
- ▶ Content-Only Query
- ▶ Evaluation Metrics for Structured Text Retrieval
- ▶ Narrowed Extended XPath I
- ▶ Presenting Structured Text Retrieval Results
- ▶ Processing Overlaps
- ▶ Relevance
- ▶ Specificity
- ▶ XML

Recommended Reading

1. Crouch C. Relevance feedback at the INEX 2004 workshop. SIGIR Forum 39(1):41–42, June 2005.
2. Denoyer L. and Gallinari P. Report on the XML mining track at INEX 2005 and INEX 2006: categorization and clustering of XML documents. SIGIR Forum 41(1):79–90, June 2007.
3. Fuhr N., Gövert N., Kazai G., and Lalmas M. (eds.). In Proc. 1st Workshop of the INitiative for the Evaluation of XML Retrieval, 2002.
4. Fuhr N., Lalmas M., and Malik S. (eds.). Proc 2nd Workshop of the INitiative for the Evaluation of XML Retrieval, 2003.
5. Fuhr N., Lalmas M., Malik S., and Kazai G. (eds.). Advances in XML Information Retrieval and Evaluation. In Proc. 4th Int. Workshop of the Initiative for the Evaluation of XML Retrieval, 2005.
6. Fuhr N., Lalmas M., Malik S., and Szlávik Z. (eds.). Advances in XML Information Retrieval. In Proc. 3rd Int. Workshop of the Initiative for the Evaluation of XML Retrieval, 2004.
7. Fuhr N., Lalmas M., and Trotman A. (eds.). Comparative Evaluation of XML Information Retrieval Systems, In Proc. 5th Int. Workshop of the Initiative for the Evaluation of XML Retrieval, 2006.
8. Fuhr N., Lalmas M., Trotman A., and Kamps J. (eds.). Focused access to XML documents. In Proc. 6th Int. Workshop of the Initiative for the Evaluation of XML Retrieval, 2007.
9. Geva S. and Sahama T. The NLP task at INEX 2004. SIGIR Forum 39(1):50–53. June 2005.
10. Gövert N., Fuhr N., Lalmas M., and Kazai G. Evaluating the effectiveness of content-oriented XML retrieval methods. Inform. Retri., 9(6):699–722, 2006.
11. Pharo N. and Trotman A. The use case track at INEX 2006. SIGIR Forum 41(1):64–66, June 2007.
12. Tombros A., Malik S., and Larsen B. Report on the INEX 2004 interactive track. SIGIR Forum 39(1):43–49, June 2005.
13. Trotman A. and Lalmas M. The interpretation of CAS. In Proc. 4th Int. Workshop of the Initiative for the Evaluation of XML Retrieval, 2006, pp. 58–71.
14. van Zwol R., Kazai G., and Lalmas M. The Multimedia Track at INEX 2005: Overview, Advances in XML Information Retrieval and Evaluation. In Proc. 4th Int. Workshop of the Initiative for the Evaluation of XML Retrieval, 2005.
15. Voorhees E.M. and Harman D.K. TREC: Experiment and Evaluation in Information Retrieval (Digital Libraries and Electronic), MIT, Cambridge, MA.

Initiator

Kaladhar Voruganti
Network Appliance, Sunnyvale, CA, USA

Synonym

SCSI initiator

Definition

In SCSI protocol, the client which requests data is known as the initiator. The initiator typically resides on the host server that is accessing the storage. Initiators can also reside in storage virtualization boxes which can act as both initiators and targets.

Key Points

The SCSI initiator can reside in software or in a hardware adapter card. SCSI protocol provides commands

that allow initiators to reserve LUNs and prevent them from being accessed by other initiators. SCSI initiators communicate with SCSI targets using SCSI block level protocol. The block level protocol data is transported via other transport protocols such as parallel SCSI, Fiber Channel, Serial SCSI or Infiniband.

Cross-references

► Storage Protocols
► Target

In-Memory DBMS

► Main Memory DBMS

In-Network Aggregation

► Query Optimization in Sensor Networks

In-Network Query Processing

Samuel Madden
Massachussetts Institute of Technology, Cambridge, MA, USA

Synonyms

Tiny aggregation (TAG); TinyDB

Definition

In-Network query processing refers to the complete or partial evaluation of database queries at the edges of a network, rather than in a centralized database server. Though this phrase may also apply to the general process of distributed or parallel query evaluation, it is most commonly applied to environments like sensor networks, where the network edges consist of small, wireless devices with power and CPU constraints. Typically these devices produce the data to be processed via local sensors, so processing in the edges of the network can be beneficial because it can reduce bandwidth usage, which in turn can conserve energy.

Historical Background

Sensor networks are collections of small, inexpensive battery-powered, wirelessly networked devices equipped with sensors (microphones, temperature sensors, etc.) that offer the potential to monitor the world with unprecedented fidelity. To capture data from these networks, researchers have proposed several sensor network database systems, including TinyDB [11], Cougar [15], and SwissQM [12] have been proposed. These systems provide a high level SQL-like query language that allows users to specify what data they would like to capture from the network.

The canonical sensor network platform is the Berkeley Mote hardware running the TinyOS operating system [8]. Initial versions of the motes used Atmel 8-bit microprocessors and 40 kbit/s radios; newer generations, developed by companies like Crossbow Technologies (http://www.xbow.com) and Moteiv Technologies (http://www.moteiv.com) use Zigbee (802.15.4) radios running at 250 kbits/s and Atmel or Texas Instruments 8 or 16 bit microprocessors running at 4–8 MHz. Nodes typically are very memory constrained (with 4–10 KB of RAM and 48–128 KB of non-volatile flash-based program memory.) Most varieties can be interfaced to sensors that can capture a variety of readings, including light, temperature, humidity, vibration, acceleration, sounds, or images. The limited processing power and radio bandwidth of these devices constrains sample rates to at most a few kilosamples per second. The dominant power cost of operating these nodes is often cited as network transmissions [9,10]. Table 1 summarizes the major operations in a sensor network, and illustrates the reason why communication costs are considered dominant: the time spent sending messages is huge compared to the time spent computing or sampling. Therefore, the total energy consumption is dominated by message transmissions (in this example, communication cost is 50 times the combined cost to sample a sensor and evaluate a predicate).

Given these high per-message energy costs, one primary goal of sensor network database systems is to reduce message transmissions; in network processing is key to do this, as explained in the next section.

Foundations

Before describing the various in-network processing techniques that have been proposed, it is important

In-Network Query Processing. Table 1. Major activities in a data collection network, with time, power, and energy for each. Computation is assumed to be one sample, one message transmission, and one predicate evaluation per second (with no reception.) The sensor is assumed to be attached directly to the on-chip ADC. The sensor node is a MicaZ-class [4] mote with Atmel Atmega 128L microprocessor at 4 MHz and a TI/ChipCon CC 2420 250 kbps ZigBee radio running at 0 dBm. Predicate evaluation is assumed to take 400 cycles; sensor sampling is assumed to take 40 cycles

Activity	Time per action	Power	Energy
Send a message	3 ms	60 mW	180 μJ
Sample a sensor	10 μs	24 mW	0.24 μJ
Evaluate a predicate	100 μs	24 mW	2.4 μJ
Idle	899 ms	45 μW	40 μJ

to understand the basics of sensor network query languages and the usage model for the systems.

Network Formation and Communication

Typically, a user deploys one of these systems by placing a collection of static sensor nodes around an area to be monitored. These nodes contain a pre-compiled binary image of the sensor network database, but are not yet running any queries. When powered up, the nodes immediately begin to organize themselves into an ad-hoc network which typically takes the form a *spanning tree* rooted at a *root node* connected to a *basestation*. The basestation is usually a more powerful machine, such as a laptop PC, which issues queries to nodes and collects, processes, and visualizes query results.

Tree formation is accomplished by have the basestation periodically broadcast a beacon message. Nodes that hear this beacon re-broadcast it, indicating that they are one hop from the basestation. Nodes that hear those messages in turn re-broadcast them, indicating that they are two hops from the basestation, and so on. This process of (re)broadcasting beacons occurs continuously, such that (as long as the network is connected) all nodes will eventually hear a beacon message. When a node hears a beacon message, it chooses a node from which it heard the message to be its *parent*, sending messages through that parent when it needs to transmit data to the basestation. (In general, parent selection is quite complicated, as a node may hear beacons from several candidate parents. Early papers by Woo et al. [14] and DeCouto et al. [5] provide details.)

Users interact with the system by issuing queries at the basestation, which in turn broadcasts queries out into the network. Queries are typically disseminated via flooding down the routing tree. As nodes receive the query, they begin processing it. The basic programming model is data-parallel: each node runs the same query over data that it locally produces or receives from its neighbors. As nodes produce query results, they apply in-network processing to reduce query results, and then send those reduced result up the routing tree, towards the basestation, which receives the results and possibly merges them together to form a final query answer.

Query Language and Data Model

Most sensor network databases systems provide a SQL-like query interface. For example, the a TinySQL query (used in TinyDB) that requests the temperature from every node in a sensor network whose value is greater than 25°C once per second would look like:

```
SELECT nodeid, temperature
FROM sensors
WHERE temperature > 25°C
SAMPLE PERIOD 1s
```

This query is essentially standard SQL, with a few small additions. First, the `SAMPLE PERIOD` clause requests that a data reading be produced once every second. This means that each query produces a continuous stream of results rather than a single result set. Second, the `nodeid` attribute is a unique identifier assigned to each sensor node and available in every query. Third, the table `sensors` is *virtual table* of sensor readings. Here, *virtual* means that it conceptually contains one row for every sensor type (light, temperature, etc.) from every sensor node at every possible instant, but all of those rows and columns are not actually materialized. Instead, only the sensor readings needed to answer a particular query are actually generated.

Note also that although this table appears to be a single logical table its rows are actually produced by different, physically disjoint sensors. This is the key to in-network processing: when a sensor receives a query, it begins sampling its sensors at the appropriate

rate, applying predicates and aggregating data locally and forwarding on only those readings that will eventually be a part of the final query answer.

It should be clear that selection predicates over a single attribute of the sensors table can always be evaluated locally, inside of the network, before any network transmission is done. The remainded of this section discusses techniques that have been developed for in-network processing for aggregate and join queries.

In-Network Processing of Aggregates

Aggregate queries are particularly common in sensor networks, since users are often more interested in what is happening in general geographic regions rather than at a specific sensor. A common environmental monitoring query might ask for the temperature in a building grouped by room number:

```
SELECT roomNo, AVG(temp)
FROM sensors
GROUP BY roomNo
SAMPLE PERIOD 1 s
```

A naive implementation of sensor network aggregation would be to use a centralized, *server-based* approach where all sensor readings are sent to the base station, which then computes the aggregates. In the TAG system [10], however, the authors proposed computing aggregates in-network, and showed that this approach requires fewer message transmissions, is lower latency, and uses less power than the server-based approach.

Consider first the case of an aggregation query without group. Once the query has been disseminated into the network, the TAG data processing phase begins. In this phase, aggregate values are continually routed up from children to parents. The goal of the TAG algorithm is to produce a single aggregate value (or a single value per group) that combines the readings of all devices in the network once per sample period.

The basic insight of the TAG protocol is that it is possible to *partially aggregate* sensor values together at intermediate points inside the network. TAG accomplishes this by having each parent node collect readings from its children before doing its own transmission. Rather than simply forwarding all of the raw data, each parent node combines its data with data from its children to produce a compact *partial state record* (PSR). For example, suppose the user wants to compute the average temperature in the network. In TAG, the PSR representation for an average is a <*sum*, *count*> pair. Each node transmits exactly one PSR per sample period. Suppose a node receives a set P of n PSRs from its children, such that $P = p_1,...,p_n$. If the node has the local sensor value v; then it can compute its own partial state record as:

$$sum = v + \sum_{i=1}^{n} p_i \cdot sum$$

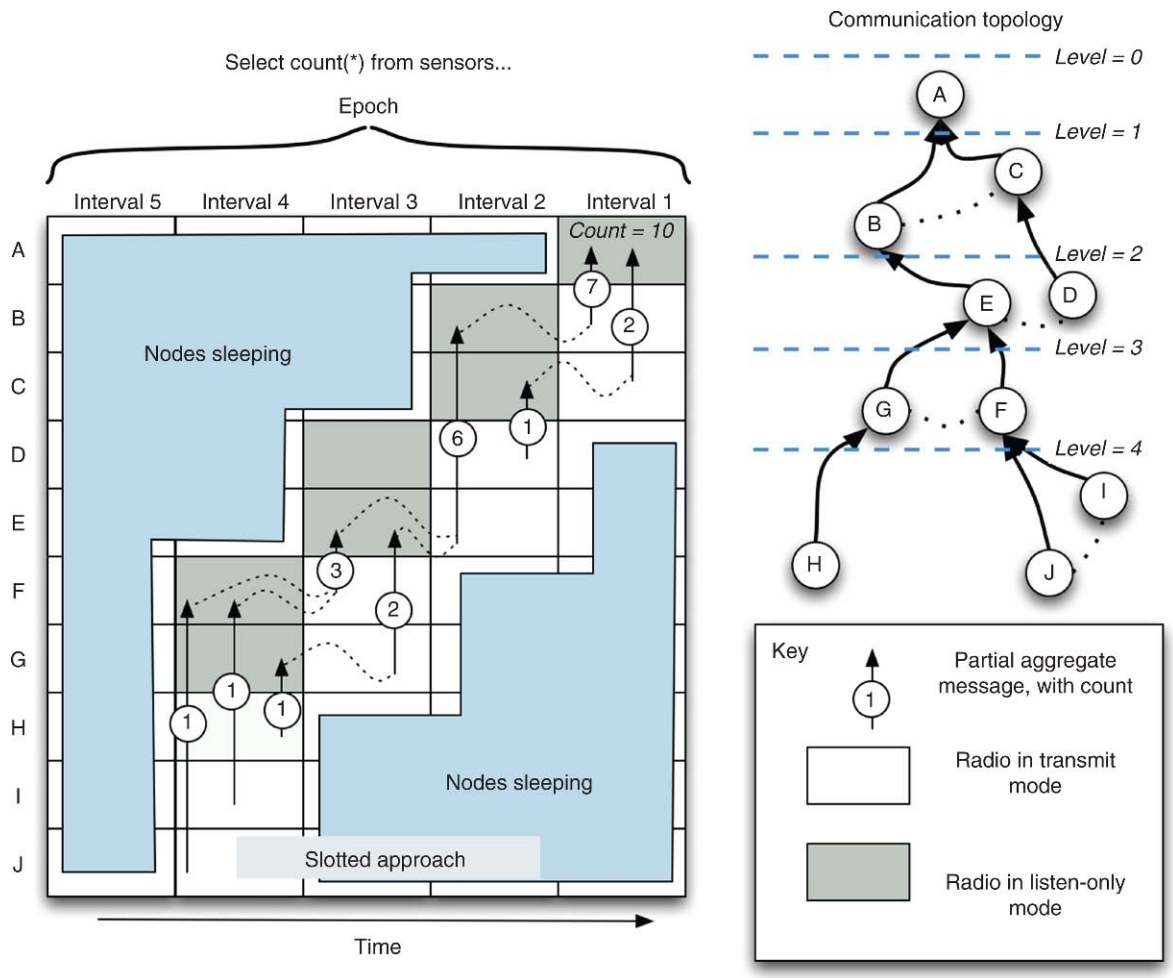
$$count = n + 1$$

By this definition, if a node has no children, it transmits the PSR <v, 1>. Finally, at the root of the network, the final average can be computed from the root's PSR as *PSR.sum/PSR.count*. To support the addition of new aggregation functions to the system, TAG includes a facility that allows programmers to define functions that initialize, merge, and compute the final value of partial state records for different aggregation functions.

Clearly, this technique reduces both the number of messages and total number of bytes that must be sent in networks of even modest size, as a parent with k nodes below it must transmit k readings without the TAG technique but only two values with the TAG approach.

For tag TAG-like protocols to work, nodes must nodes wait to receive readings from children. TAG does this by sub-dividing each sample period up into a series of time intervals corresponding to transmission slots, and assign nodes deeper in the routing tree to an earlier interval. This approach will work as long as sample periods are relatively long (on the order of half a second or so for Mica motes) and as long as it is possible to time synchronize nodes (using a technique like RBS [6]). The TinyDB system demonstrated a proof-of-concept implementation of this technique.

Figure 1 illustrates this in-network aggregation scheme for a simple `COUNT` query that reports the number of nodes in the network. In the figure, time advances from left to right, and different nodes in the communication topology are shown along the Y axis. Nodes transmit during the interval corresponding to their depth in the tree, so `H`, `I`, and `J` transmit first, during interval 4, because they are at level 4. Transmissions are indicated by arrows from sender to receiver, and the numbers in circles on the arrows represent `COUNT`s contained within each partial state record (which consists simply of a running count.) Readings from these three sensors are combined, by summing the running counts in the PSRs, at nodes `G` and `F`, both of



In-Network Query Processing. Figure 1. Partial state records flowing up the tree during and interval-based aggregation approach.

which transmit new partial state records during interval 3. Readings flow up the tree in this manner until they reach node A, which then computes the final count of 10. Notice that motes are idle for a significant portion of each sample period, during which time they can enter a low power sleeping state.

Supporting grouping in this setting can be done simply by tagging each partial state record with a group number (which can be derived by local evaluation of the grouping predicate at each sensor), and then treating each group as a separate aggregation operation that is independently merged and forwarded through the network.

For queries involving a HAVING clause, the TAG system proposes several optimizations for early in-network rejection of aggregates that do not satisfy the clause. The main observation is that in MIN/MAX queries it may be possible to determine that a particular group will definitely not satisfy the HAVING clause before the group reaches the root of the network.

Classes of Aggregates

One observation is that some aggregates will show more or less benefit from the TAG in-network processing techniques. In particular, if the partial state record is very compact (as in a COUNT or AVERAGE) query, there is a tremendous win to in-network aggregation. However, for aggregate functions that require access to all or most of the sensor readings before that final aggregate can be computed, the benefit is much less. For example, computing the exact median of a collection of readings requires all of the readings to be present. Hence, the TAG implementation of median offers no reduction in data transmission over a naive,

centralized approach. (Greenwald and Khanna [7] propose techniques for approximate and efficient computation of order statistics like medians in sensor networks.)

There are other important semantic properties of aggregates. One that is of particular interest relates to the sensitivity of the aggregation function to duplicates. Some functions, like MAX and MIN, are insensitive to duplicates: even if a particular reading is merged together with a partial state record many times, the final value of the aggregate will not be affected. Other aggregates, like SUM, COUNT, and AVERAGE are obviously sensitive to duplicates. The TAG system exploits this property by using a directed acyclic graph that terminates at the network root rather than a simple spanning tree when computing duplicate insensitive aggregates. This substantially improves the reliability of the algorithms (using little additional energy since radios generally operate in a broadcast mode that allows multiple receivers to hear a message at no additional cost). Several researchers [3,13] have proposed methods that accurately approximate the value of duplicate sensitive aggregates using a duplicate insensitive synopsis data structure, allowing the same DAG-based network topologies to be used with a broader range of aggregates.

In-Network Processing of Joins

There have been several join algorithms proposed for specific classes of join queries in sensor networks.

Bonfils and Bonnet [2] view joins as a way to correlate or compare data between several sensors in a network. They propose a setting in which two sensors each produce a data stream, and the user wishes to apply a temporal join operation over these streams that combines readings from approximately the same time together when a predicate is satisfied. Clearly, such an operation can be done at the root of the network. They observe, however, that it can also be done at the root of any subtree in the network that has both producer nodes as a subchild. If the join is data-reducing (e.g., it filters out some readings or produces tuples that are smaller than the combined size of the tuples from both producers) then this should result in an overall reduction in network bandwidth. Rather than trying to compute the best location for such operators centrally (using global network topology information), they propose a distributed algorithm where the join operator slowly moves towards the nodes, tending to move closer to the node that is producing a larger fraction of the join data, since that will result in the greatest overall reduction in network bandwidth.

Abadi and Madden [1] propose a method called REED for in-network execution of joins that involve a static table of data from outside of the sensor network with a stream of sensor data. Such situations arise, for example, where there is a table of thresholds that dictate what data should be sent out of the network at different times of the day. The observation here is that if the table is static and the join is cardinality reducing, then paying the cost of disseminating the table once will save energy in the long run. When there is sufficient memory on each of the nodes to store the complete table, such joins can be evaluated purely locally at each node. The REED system proposes several techniques that can be used to execute queries in-network when the static table exceeds the memory available on any one node. The most effective techniques involve sending just a portion of the table to each node (e.g., the thresholds for 8 A.M. to 8 P.M.) and then sending the raw data out of the network when the needed portion is unavailable (e.g., when it is between 8 P.M. and 8 A.M.)

Key Applications

In-network processing of queries can be used in any setting where declarative queries over sensor networks are needed. In network processing techniques make selection, join, and aggregation queries substantially cheaper to run – often saving an order of magnitude in totally energy cost to process a given query. These savings mean that sensor network deployments can last longer, or provide higher sample rates for the same longevity when compared to solutions that do not use in-network processing.

Cross-references

- ▶ Database Languages for Sensor Networks
- ▶ Distributed Query Processing
- ▶ Partial Pre-aggregation

Recommended Reading

1. Abadi D. and Madden S. Reed: Robust, Efficient Filtering and Event Detection in Sensor Networks. In Proc. 31st Int. Conf. on Very Large Data Bases, 2005, pp. 769–780.
2. Bonfils B. and Bonnet P. Adaptive and Decentralized Operator Placement for In-network Query Processing. In Proc. 2nd Int. Workshop Inf. Proc. in Sensor Networks, 2003, pp. 47–62.

3. Considine J., Li F., Kollios G., and Byers J. Approximate Aggregation Techniques for Sensor Databases. In Proc. 20th Int. Conf. on Data Engineering, 2004, pp. 449–460.
4. Crossbow, Inc. Micaz wireless sensor node data sheet. http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICAz_Datasheet.pdf.
5. De Couto D.S.J., Aguayo D., Bicket J., and Morris R. A High-Throughput Path Metric for Multi-hop Wireless Routing. In Proc. 9th Annual Int. Conf. on Mobile Computing and Networking, 2003, pp. 134–146.
6. Elson J. and Estrin D. Time Synchronization for Wireless Sensor Networks. In Proc. 15th Int. Parallel & Distributed Processing Symp., 2001, pp. 1965–1970.
7. Greenwald M.B. and Khanna S. Power-Conserving Computation of Order-Statistics Over Sensor Networks. In Proc. 23rd ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems, 2004, pp. 275–285.
8. Hill J., Szewczyk R., Woo A., Hollar S., Culler D., and Pister K. System Architecture Directions for Networked Sensors. In Proc. 9th Int. Conf. on Architectural Support for Programming Languages and Operating Systems, 2000, pp. 93–104.
9. Madden S. The Design and Evaluation of a Query Processing Architecture for Sensor Networks. PhD thesis, UC Berkeley, 2003.
10. Madden S., Franklin M.J., Hellerstein J.M., and Hong W. TAG: A Tiny AGgregation Service for Ad-Hoc Sensor Networks. In Proc. 5th USENIX Symp. on Operating System Design and Implementation, 2002.
11. Madden S., Hong W., Hellerstein J.M., and Franklin M. TinyDB web page. http://telegraph.cs.berkeley.edu/tinydb.
12. Müller R., Alonso G., and Kossmann D. SwissQM: Next Generation Data Processing in Sensor Networks. In Proc. 3rd Biennial Conf. on Innovative Data Systems Research, 2007, pp. 1–9.
13. Nath S. and Gibbons P.B. Synopsis Diffusion for Robust Aggregation in Sensor Networks. In Proc. 2nd Int. Conf. on Embedded Networked Sensor Systems, 2004, pp. 250–262.
14. Woo A., Tong T., and Culler D. Taming the Underlying Challenges of Reliable Multihop Routing in Sensor Networks. In Proc. 1st Int. Conf. on Embedded Networked Sensor Systems, 2003, pp. 14–27.
15. Yao Y. and Gehrke J. Query Processing in Sensor Networks. In Proc. 1st Biennial Conf. on Innovative Data Systems Research, 2003.

Instance Identification

► Record Matching

Instance-Completeness

► BP-Completeness

Instant

► Chronon

Instant Relation

► Event in Temporal Databases

Instruction Cache

► Processor Cache

Integrated DB&IR Semi-Structured Text Retrieval

Ralf Schenkel[1], Martin Theobald[2]
[1]Max-Planck Institute for Informatics, Saarbrücken, Germany
[2]Stanford University, Stanford, CA, USA

Synonyms

Using efficient database technology (DB) for effective information retrieval (IR) of semi-structured text

Definition

Integrated DB&IR semi-structured text retrieval combines IR-style scoring and ranking methods for effective search with indexing techniques and processing algorithms from the database world for efficient query evaluation.

Historical Background

Database research has traditionally focused on semi-structured documents that represent structured data with a well-defined schema and only little unstructured, textual content (aka. "data-centric" XML). Typical examples for such documents are invoices, purchase orders, or even complete bibliographies.

Early work in the field concentrated on "classical" data management problems for XML: storing XML data in relational or native XML systems, defining

query languages that integrate conditions on the structure and the content of results (like SQL for relational data), efficiently processing these queries on huge collections of documents, and auxiliary structures (like structural summaries and path indexes) to support processing. The focus of this work was on space and runtime efficiency.

Foundations

When semi-structured data formats, especially XML, became popular for storing and exchanging information throughout the 1990s, an abundance of different schemas for such data was developed independently. This created a challenge for existing database query languages like the Structured Query Language (SQL), focusing on exactly matching conditions on the content and structure of results. Now, similar structured and textual information was present in different, heterogeneous formats and schemas, which was often the case when information from different sources was integrated in a single application. In reaction to this, the strict, SQL-style querying paradigm prevalent at that time evolved towards a more relaxed IR-style vague search with partial and imprecise answers. This created three main scientific problems at the intersection of the DB and IR fields: (i) the definition of query languages to specify vague constraints on the structure and/or the content of results, (ii) the definition of relevance scores to rank results by their degree of matching with the query, and (iii) efficient algorithms and auxiliary structures to quickly compute the best results to a vague query, according to the relevance score. Solutions to these problems were developed mainly with data-centric documents in mind, and with a strong focus on query languages and algorithms (thus addressing problems i and iii).

One of the first systems to retrieve semi-structured data using a relaxed query language was the *Lore* [1] system developed at Stanford University. Its object-oriented, OQL-style, query language, coined *Lorel*, provided regular path expressions and tag wildcards to express structural vagueness, as well as keyword conditions over the content of subtrees matching the path condition. However, it was still more of a database query language as it did not yet foresee any ranking for the results.

Querying Semi-Structured Data with IR Support

A large body of proposals have been made for query languages over semi-structured data that support IR-style vague conditions on structure and content. The simplest of them merely aim to extend keyword search as known from text retrieval to semi-structured data by enhancing the keyword conditions with tag names of elements that should be matched (like in the query "`author:widom`" which restricts occurrences of the keyword "`widom`" to elements with tag name "`author`"). Matches to such a query are subtrees of the document that contain matches to all (for conjunctive evaluation) or at least one (for disjunctive evaluation) keyword. An important aspect here is the selection of subtrees of the right granularity, as large subtrees (such as the complete document) would often not be specific enough. The proposed solutions usually consider some variant of lowest common ancestor (LCA) search to identify suitable root nodes of the result trees, sometimes allowing for additional path conditions from elements containing the keywords towards the root element. Hardly any of the early proposed systems consider ranking of results; instead, they focus on efficient methods to retrieve all possible matches. Note that these techniques were primarily developed for data-centric XML (such as bibliographies) and cannot easily be applied for true full-text search over semi-structured documents.

Among the most prominent approaches for keyword-based ranked retrieval of XML data is XRank [7]. It generalizes traditional link analysis algorithms such as PageRank for authority ranking of linked XML collections and conceptually treats each XML element as an interlinked node in a large element graph. Then the *element rank* of an XML element corresponds to the PageRank value computed over a mixture of containment edges, obtained from the XML tree structure, and hyperlink edges, obtained from the inter-document link structure.

Full-fledged XML query languages with rich IR models for ranked retrieval were proposed by [6,13]. *XIRQL* [6], a pioneer in the area of ranked XML retrieval, presents a path algebra based on XQL, an early ancestor of W3C's XQuery, for processing and optimizing structured queries. It combines Boolean query operators with probabilistically derived weights for ranked result output, thus transferring the probabilistic IR paradigm to the XML case. It defines data-type-specific vague predicates for similarity search over differently typed XML elements such as person names or numbers, and it introduces a notion of *index objects* that serve as anchors from which the probabilistic

weights are derived (in the classic IR notion of a document). Using index objects follows the idea that only nodes of specific type and granularity in the document hierarchy should be presented as results to the end-user. Defining these index objects, however, may be strongly schema-dependent and assumes substantial knowledge about the general document structure and user intent, which typically requires their manual pre-selection from a – preferably compact – document type definition (DTD).

The *XXL* search engine [13] specifies a full-fledged, SQL-oriented query language for ranked XML IR with a high semantic expressiveness that made it stand way apart from the predominant XQL and XPath language standards at its time. For ranked result output, XXL leverages both a standard IR vector space model and an ontology-oriented similarity search for the dynamic relaxation of structure and term conditions. The principal structure of the query, however, is evaluated in a strictly Boolean manner.

More recently, various groups from both the DB and IR fields have started adding IR-style keyword conditions and full-text search to existing XML query languages. The NEXI (for Narrowed Extended XPath I) query language used in the INEX (INitiative for the Evaluation of XML Retrieval (INEX), see http://inex.is.informatik.uni-duisburg.de) benchmark series aims at a simplified and easy-to-comprehend subset of XPath, the W3C standard language for path matches within a document, with extended IR functionality. Here the `about` operator already anticipates the role of the `ftcontains` operator in the later W3C Full-Text extensions of XPath 2.0 and XQuery 1.0. TeXQuery [2], on the other hand, has been the foundation for the W3C's official Full-Text extension to XPath and XQuery, which extends these languages with the option to express actual full-text queries over XML documents. It provides many retrieval options known from text retrieval, such as phrases, weighting terms, and expanding terms using ontologies, but leaves details of the scoring model used to rank results up to the implementation.

Pioneering work in the area of vague structural matches was done by [3,11] (and refined later by [4]), who proposed relaxing queries with structural constraints to find matches in structurally similar, but not exactly matching documents. The *FlexPath* [4] algorithm integrates structure and keyword queries and regards the query structure as templates for the context of a full-text keyword search. The query structure (as well as the content conditions) can be dynamically relaxed for ranked result output according to predefined tree editing operations when matched against the structure of the XML input documents.

Query Processing for Semi-Structured Text Retrieval

Efficient evaluation and ranking of conditions on content and structure of semi-structured data has been a very fruitful and popular research area in recent years. The majority of the proposed algorithms for efficient query evaluation combine some form of precomputed auxiliary indexes (like inverted files) with top-k aware processing algorithms, most notably Fagin's family of threshold algorithms (TA), which provide threshold-based candidate pruning and early termination.

XRank uses inverted lists containing – for each term – the elements that contain the term, sorted in descending order of element rank, along with a threshold algorithm for pruning the search space. The *FlexPath* query processor uses separate index structures for storing and retrieving the structure- and content-related conditions of a path query. The *Whirlpool* system introduced by Marian et al. [10] provides a flexible architecture for processing top-k queries on XML documents *adaptively*. Whirlpool allows partial matches to the same query to follow different execution plans, and takes advantage of the top-k query model to make dynamic choices during query processing. The key features of Whirlpool are: (i) a partial match that is highly likely to end up in the top-k set is processed in a prioritized manner, and (ii) a partial match unlikely to be in the top-k set follows the cheapest plan that enables its early pruning. Whirlpool provides several adaptivity policies and also supports parallel evaluations.

TopX [12], the actual successor of XXL, focuses on a small, XPath-like subset of the XXL query language which allows for a radically different query processing architecture that outperforms XXL in terms of efficiency by a large margin. As a native top-k engine for XML, TopX also uses sorted index lists, but keeps a candidate queue in-memory and therefore is able to focus on sequential disk access and on minimizing random disk access through sophisticated index structures and judiciously scheduled index access decisions.

A large effort has been made on mapping XML to relational schemas with highly specialized index structures for efficient support of approximate query

processing, including support for IR-style retrieval functionality. *PF/Tijah* [8], which is now a part of *MonetDB/XQuery*, is an example for such an XQuery engine.

Support for XML-IR in Commercial Database Systems

Meanwhile, all commercially available databases with XML support, relational or native, provide some support for IR-style content search in combination to structural queries. As the full-text extensions of XPath and XQuery have not yet been finalized, systems typically come with their own extensions of their query language that are incompatible with – and sometimes less powerful than – the W3C proposals. Frequently, existing text search components are extended for XML support and provide the standard text search features (like phrase search, proximity conditions, stemming, etc.) for searching XML elements, usually with some scoring function to rank results.

Key Applications

The techniques presented before can be applied for efficiently retrieving information from large, possibly heterogeneous collections of semi-structured data. This includes more data-centric collections like bibliographies, collections of textual documents (like abstracts or full-text of publications or books), heterogeneous data exported from different sources, and eventually documents on the Web.

Cross-references

- ▶ XML Data Management:XML prototypes/systems
- ▶ XML Indexing
- ▶ Top-k XML Query Processing
- ▶ XQuery Full-Text

Recommended Reading

1. Abiteboul S., Quass D., McHugh J., Widom J., and Wiener J.L. The Lorel Query Language for Semistructured Data. Int. J. Digital Libraries, 1(1):68–88, 1997.
2. Amer-Yahia S., Botev C., and Shanmugasundaram J. TeXQuery: a full-text search extension to XQuery. In Proc. 12th Int. World Wide Web Conference, 2004, pp. 583–594.
3. Amer-Yahia S., Cho S., and Srivastava D. Tree Pattern Relaxation. In Advances in Database Technology, Proc. 8th Int. Conf. on Extending Database Technology, 2002, pp. 496–513.
4. Amer-Yahia S., Lakshmanan L.V.S., and Pandit S. FleXPath: Flexible Structure and Full-Text Querying for XML. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 2004, pp. 83–94.
5. Cohen S., Mamou J., Kanza Y., and Sagiv Y. XSEarch: A Semantic Search Engine for XML. In Proc. 29th Int. Conf. on Very Large Data Bases, 2003, pp. 45–56.
6. Fuhr N. and Großjohann K. XIRQL: A Query Language for Information Retrieval in XML Documents. In Proc. 24th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 2001, pp. 172–180.
7. Guo L., Shao F., Botev C., and Shanmugasundaram J. XRANK: Ranked Keyword Search over XML Documents. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 2003, pp. 16–27.
8. Hiemstra D., Rode H., Van Os R., and Flokstra J. PF/Tijah: text search in an XML database system. In Proc. 2nd International Workshop on Open Source Information Retrieval, 2006.
9. Hristidis V., Papakonstantinou Y., and Balmin A. Keyword Proximity Search on XML Graphs. In Proc. 19th Int. Conf. on Data Engineering, 2003, pp. 367–378.
10. Marian A., Amer-Yahia S., Koudas N., and Srivastava D. Adaptive Processing of Top-*k* Queries in XML. In Proc. 21st Int. Conf. on Data Engineering, 2005, pp. 162–173.
11. Schlieder T. and Meuss H. Querying and ranking XML documents. J. American. Soc. for Inf. Sci. & Tech. 53(6):489–503, 2002.
12. Theobald M., Bast H., Majumdar D., Schenkel R., and Weikum G. TopX: efficient and versatile top-*k* query processing for semistructured data. VLDB J., 17(1):81–115, 2008.
13. Theobald A. and Weikum G. Adding Relevance to XML. In Proc. 3rd Int. Workshop on the World Wide Web and Databases, 2000, pp. 105–124.
14. Xu Y. and Papakonstantinou Y. Efficient Keyword Search for Smallest LCAs in XML Databases, In Proc. ACM SIGMOD Int. Conf. on Management of Data, 2005, pp. 537–538.

Integration of Rules and Ontologies

Jan Małuszyński
Linköping University, Linköping, Sweden

Definition

The layered structure of the Semantic Web (see http://www.w3.org/2007/03/layerCake.png) adopted by the World Wide Web Consortium W3C includes, among others, the Ontology layer with the web ontology language OWL and the rule layer with the emerging Rule Interchange Format (RIF) http://www.w3.org/TR/rif-fld/ which allows rules to be translated between rule languages. The integration of rules and ontologies aims at developing techniques for interoperability between rules and ontologies in the Semantic Web. This is necessary for rule-based applications to access existing domain ontologies. In most of the proposals the integration is achieved by defining and implementing a

new language which is a common extension of a given rule language and a given ontology language, enhancing the expressive power of each of the components. Alternatively, the integration of rules and ontologies may be achieved by designing from scratch one language sufficiently expressive to define both rules and ontologies as well as their combinations.

Historical Background

In the initial phase of the Semantic Web research a significant effort was devoted to defining a language for ontology modeling. In 2004, it resulted in OWL, a family of three ontology languages OWL Lite, OWL DL, and OWL Full, based on Description Logics(DL). Each of them is a subset of the next one. The first two are syntactic variants of expressive description logics with semantics given by translation to formulae of first-order logic with equality. OWL DL (hence also its subsets) is supported by several reasoners. The original intention was to define OWL as a layer on top of RDF Schema which itself can be seen as ontology language. OWL Full, designed to achieve this goal, has a non-standard semantics and is difficult to implement. On the other hand, OWL DL includes a substantial subset of RDF Schema, which is extensively used in ontology definitions.

The importance of rules for web applications is reflected by the rule layer in the Semantic Web architecture. The rules formalisms considered for this layer offer modeling primitives not expressible in OWL. Their integration with OWL would thus enhance the expressive power of the latter.

In contrast to the ontology layer, no standard has been proposed yet for the rule layer. The rule languages proposed for the Semantic Web originate mainly from *logic programming* (see e.g., [18]). In contrast to OWL DL, they usually adopt the *closed world assumption*. This means that if a fact cannot be derived from a knowledge base it is concluded to be false. In logic programming, the closed world assumption is implemented by the *negation-as-failure* rule which returns $\neg p$ on failure to prove a fact p. This is an example of *non-monotonic reasoning*, not allowed in FOL.

Designing rules languages for the Semantic Web is among the objectives of the RuleML initiative (http://www.ruleml.org/). The W3C RIF Working Group is developing a core rule language as a basis for rule interchange (for more details see http://www.w3.org/2005/rules/wiki/RIF_Working_Group).

Foundations

Shortcomings of OWL

The following examples show why OWL is not sufficient for some applications and motivate its extensions. OWL Ontologies include classes (e.g., *Person*, *Woman*, *Man*) which are unary predicates and properties (e.g., *ParentOf*, *SisterOf*) which are binary predicates, but predicates of arity larger than two are not allowed. In OWL it is not possible to formalize the statement "an aunt of a person is a sister of a parent of that person." Also, the semantics of OWL does not allow to conclude that Mary is not a sister of John if the assertion *SisterOf*(*john*,*mary*) is not a logical consequence of the ontology. This kind of reasoning based on closed world assumption is useful in some applications. Most of the rule languages proposed for the Semantic Web do not share these shortcomings.

Rule Languages for Integration

The rule languages considered in integration proposals are usually extensions of Datalog. Generally rules have a form of "if" statements, where the predecessor, called the *body* of the rule, is a Boolean condition and the successor, called the *head*, specifies a conclusion to be drawn if the condition is satisfied.

In Datalog the condition of a rule is a conjunction of zero or more atomic formulae of the form $p(t_1,...,t_n)$ where p is an n-ary predicate symbol and $t_1,...,t_n$ are constant symbols or variables. Hence they are a restricted kind of FOL *terms* (see First order logic: syntax).

The head of a rule is an atomic formula (atom). For example, the rule

$$auntOf(X, Y) \leftarrow parentOf(Z, Y), sisterOf(X, Z)$$

states that X is an aunt of Y if Z is a parent of Y and X is this parent's sister. The semantics of Datalog associates with every set of rules (rulebase) its least Herbrand model (see e.g., [18]), where each ground (i.e., variable-free) atom is associated with a truth value *true* or *false*. The least Herbrand model is represented as the set of all atoms assigned to true. These are all the ground atoms which follow from the rules interpreted as implications in FOL. For example, the least Herbrand model of the rulebase consisting of the rule above and of the facts *parentOf*(*tom*, *john*), *sisterOf*(*mary*, *tom*) includes the formula *auntOf*(*mary*, *john*). On the other hand, *auntOf*(*mary*, *tom*) does not follow in this rulebase. Hence the closed world

assumption, used in Datalog, will result in the conclusion $\neg auntOf(mary, tom)$. Datalog rulebases constitute a subclass of logic programs. The latter use FOL terms, not necessarily restricted to constants and variables. Proposals for the integration of rules and ontologies are mostly based on the following extensions of Datalog (which apply also to logic programs):

- *Datalog with negation*, where the body may additionally include negation-as-failure (NAF) literals of the form *not a* where *a* is an atom. Intuitively a NAF literal *not a* is considered true if it does not follow from the program that *a* is true. For example, $happy(john)$ can be concluded from the rulebase:

  $$happy(X) \leftarrow healthy(X), \; not\ hungry(X)$$
  $$healthy(john) \leftarrow$$

  Two commonly accepted formalizations of this intuition are: the *well-founded semantics* and the *stable model semantics* (see the survey [2]). The well-founded semantics [22] associates with a rulebase a unique (three-valued) Herbrand model, where each ground atom is assigned one of three logical values *true*, *false* or *unknown*. The stable model semantics [9] (called also the *answer set semantics*) associates with each rulebase some (possibly zero) two-valued Herbrand models. For a large class of programs relevant in practice (so called *stratified programs*, see e.g., [2]) both semantics coincide.
- *Extended Datalog*. This extension (see e.g., *extended logic programs* in [2]) makes it possible to state explicitly negative knowledge. This is achieved by allowing negative literals of the form $\neg p$, where $\neg$ is called the *strong negation* connective, in the heads of rules as well as in the bodies. For example, the rule

  $$\neg healthy(X) \leftarrow hasFever(X)$$

  allows to draw an explicit negative conclusion. In addition, NAF literals are also allowed in the bodies.
- *Rulebases with priorities*. Datalog rulebases employing strong negation may be inconsistent, i.e., may allow to draw contradictory conclusions. For example, the rules

  $$fly(X) \leftarrow bird\,(X)$$
  $$bird(Y\,) \leftarrow penguin(Y\,)$$
  $$\neg fly(X) \leftarrow penguin(X)$$
  $$penguin(tweety) \leftarrow$$

  allow to conclude $fly(tweety)$ and $\neg fly(tweety)$. In Defeasible Logic [19] and in Courteous Logic Programs [10] a priority relation on rules can be specified for a rulebase. The contradictions in the derived conclusions are then resolved by means of the defined priorities.
- *Disjunctive Datalog* [5] (see also *disjunctive logic programs* in [2]) admits disjunction of atoms in the rule heads, and conjunction of atoms and NAF literals in the bodies, e.g.,

  $$male(X) \vee female(X) \leftarrow person(X).$$

  A commonly used semantics of Disjunctive Datalog rulebases is an extension of the answer set semantics.

The rule languages are supported by implementations which make it possible to query and/or to construct the models of rulebases.

Approaches to Integration

The integration of a given rule language with a given ontology language is usually achieved by defining a common extension of both, to be called the integrated language. Alternatively, one can adopt an existing knowledge representation language expressive enough to represent rules and ontologies. As OWL is a standard ontology language the ontology languages considered in integration proposals are usually its subsets. The approaches can be classified by the degree of the integration of rules and ontologies achieved in the integrated language.

- *Homogeneous Integration*. The integrated language makes no distinction between the rule predicates and the ontology predicates. It includes the original rule language and the original ontology language as sublanguages. The integration is to be *faithful* in the sense that the sublanguages should have the same semantics as the respective original languages. The homogeneous integration is difficult to achieve since ontology languages are usually based on FOL and rule languages have different kind of semantics. Examples of the homogeneous integration include:
  - *DLP (Description Logic Programs)* [12], which is a language obtained by intersection of a Description Logic with Datalog rules interpreted as FOL implications. DLP has a limited

expressive power, but a DLP ontology can be compiled into rules and easily integrated into a rulebase of a more expressive rule language. For example *Sweet Rules* http://sweetrules.projects.semwebcentral.org/ combine DLP and Datalog with strong negation and priorities. The technique of compiling ontologies to rules is also used in DR-Prolog [1] based on Defeasible Logic.

- *F-logic*, extending classical predicate calculus with the concepts of objects, classes, and types. It is expressive enough to represent ontologies, rules and their combinations [13].
- *SWRL (Semantic Web Rule Language)* http://www.w3.org/Submission/SWRL/, extending OWL DL with rules interpreted as FOL implications. Thus, SWRL is based on FOL and does not offer nonmonotonic features, such as negation-as-failure. A so-called DL-safe subset [17] of SWRL is supported by KAON2 system http://kaon2.semanticweb.org which also offers a support for a restricted subset of F-logic.
- *Hybrid MKNF Knowledge Bases* [16], taking a modal logic as a basis of faithful integration of Description Logic with Disjunctive Datalog under the answer set semantics. A variant of this approach considering nondisjunctive hybrid MKNF knowledge bases under well-founded semantics is presented in [14].
- *Quantified Equilibrium Logic*, considered in [3] as a unified framework which embraces classical logic as well as disjunctive logic programs, thus providing a foundation for the integration of rules and ontologies.

- *Heterogeneous Integration.* In this approach, the distinction between the rule predicates and the ontology predicates is preserved in the integrated language. The integration of rules and ontologies is achieved by allowing the ontology predicates in the rules of the integrated language. Assume, for example, that an ontology classifies courses as project courses and lecture courses.

$$Project \sqcup Lecture = Course$$

It also includes assertions like *Lecture*(*cs*05), *Project* (*cs*21) or *Course*(*cs*32) (e.g., for courses including lectures and projects). The assertions indicate offered courses. A person is considered a student if he/she is enrolled in an offered lecture or project. This can be expressed by the following rules, using the ontology predicates

$$student(X) \leftarrow enrolled(X,Y), Lecture(Y)$$
$$student(X) \leftarrow enrolled(X,Y), Project(Y)$$

In addition the rulebase includes enrollment facts, e.g., *enrolled*(*joe*, *cs*32). The extended language allows thus to define ontologies using the constructs of the ontology language and the rulebases with rules referring to the ontologies. An extended rulebase together with an ontology is called a *hybrid knowledge base*. In the heterogeneous approaches implementations are often based on *hybrid reasoning principle*, where a reasoner of the ontology language is interfaced with a reasoner of the rule language to reason in the integrated language.

Two kinds of heterogeneous approaches can be distinguished:

1. *Loose coupling*. In this approach the semantics of hybrid knowledge bases is based on a transformation which eliminates ontology queries from ground extended rules by querying the underlying ontology. A ground set of extended rules is thus reduced to a set of rules without ontology predicates in the following way. If the answer to a ground ontology query in a rule body is positive, the query is removed from the rule, otherwise the rule is removed from the set. The loose coupling approach applied to the example above does not allow to conclude that Joe is a student. This is because neither *Lecture*(*cs*32) nor *Project*(*cs*32) can be derived from the ontology. Examples of loose coupling include:
   - *dl-programs* [6], combining (disjunctive) Datalog with negation under the answer set semantics with OWL DL. So called DL-queries, querying the ontology, are allowed in rule bodies. They may also refer to a variant of the ontology, where the set of its assertions is modified by the DL-query. This enables bi-directional flow of information between rules and ontologies. A variant of the language based on the well-founded semantics is presented in [7].
   - *TRIPLE* [21], a rule language with the syntax inspired by F-logic. It admits queries to the ontology in rule bodies.
   - *SWI Prolog* http://www.swi-prolog.org/, a logic programming system with a Semantic Web library

which makes it possible to invoke RDF Schema and OWL reasoners from Prolog programs.

2. *Tight integration.* In this approach the semantics of hybrid knowledge bases is defined by combining the model-theoretic semantics of the original rule language with the FOL semantics of the ontology language. For example, tight integration of Datalog (without negation) with a Description Logic can be achieved within FOL by interpreting Datalog rules as FOL implications. In this semantics $student(joe)$ is a logical consequence of the example hybrid knowledge base. As $Course(cs32)$ is an assertion of the ontology, it follows by the axiom $Project \sqcup Lecture = Course$ that in any FOL model of the ontology $Project(cs32)$ or $Lecture(cs32)$ is true. As $enrolled(joe, cs32)$ is true in every model, so the premises of at least one of the implications

$$student(joe) \leftarrow enrolled(joe, cs32), Lecture(cs32)$$
$$student(joe) \leftarrow enrolled(joe, cs32), Project(cs32)$$

must be true in any model. Hence $student(joe)$ is concluded. Examples of tight integration include:
   - *CARIN* [15], a classical work on integrating Datalog with a family of Description Logics under the FOL semantics.
   - $\mathcal{DL} + log$ [20], integrating Disjunctive Datalog under the answer set semantics with OWL DL. For each FOL model of the ontology the rules of the knowledge base are reduced to rules of Disjunctive Datalog, with stable models defined by the answer set semantics.
   - *Hybrid Rules* [4], integrating logic programs under well-founded semantics with OWL. For each FOL model of the ontology the rules of the knowledge base are reduced to a logic program with the model defined by the well-founded semantics.

The theoretical foundations developed by studying integration of ontologies with variants of Datalog provide a basis for further extensions. This includes dealing with uncertain and inconsistent knowledge, and using integrated Datalog-based languages as condition languages for ECA-Rules.

Key Applications

The integration of rules and ontologies is a relatively new research topic, focused so far on developing tools and prototypes. Key applications include semantic data integration, ontology-based web search and semantic recommendation systems. Industrial applications of this kind are discussed in the video lecture [8]. The Ontobroker system referred therein is based on F-logic. Another field of potential key applications is e-business as discussed in the tutorial video [11], with focus on Sweet Rules.

URL to Code

The following systems integrating rules and ontologies can be downloaded:

- *KAON2* from http://kaon2.semanticweb.org,
- *Sweet Rules* from http://sweetrules.projects.semwebcentral.org/,
- *SWI Prolog* from http://www.swi-prolog.org/,
- *TRIPLE* from http://triple.semanticweb.org/.

A prototype implementation of *dl-programs* (NLP-DL) can be accessed at http://con.fusion.at/nlpdl/.

Cross-references

▶ Datalog
▶ Description Logics
▶ ECA Rules
▶ First-Order Logic: Syntax
▶ First-Order Logic: Semantics
▶ Ontology
▶ OWL: Web Ontology Language
▶ Resource Description Framework (RDF) Schema (RDFS)
▶ Semantic Web
▶ W3C

Recommended Reading

1. Antoniou G. and Bikakis A. DR-Prolog: a system for defeasible reasoning with rules and ontologies on the semantic Web. IEEE Trans. Knowl. Data Eng., 19(2):233–245, 2007.
2. Baral C. and Gelfond M. Logic programming and knowledge representation. J. Logic Program., 19/20:73–148, 1994.
3. de Bruijn J., Pearce D., Polleres A., and Valverde A. Quantified equilibrium logic and hybrid rules. In Proc. 1st Int. Conf. on Web Reasoning and Rule Systems, 2007, pp. 58–72.
4. Drabent W. and Małuszyński J. Well-founded semantics for hybrid rules. In Proc. 1st Int. Conf. on Web Reasoning and Rule Systems, 2007, pp. 1–15.
5. Eiter T., Gottlob G., and Mannila H. Disjunctive datalog. ACM Trans. Database Syst., 22(3):364–418, 1997.
6. Eiter T., Lukasiewicz T., Schindlauer R., and Tompits H. Combining answer set programming with description logics for the

semantic web. In Proc. 9th Int. Conf. Principles of Knowledge Representation and Reasoning, 2004, pp. 141–151.
7. Eiter T., Lukasiewicz T., Schindlauer R., and Tompits H. Well-founded semantics for description logic programs in the semantic web. In Proc. 3rd Int. Workshop on Rules and Rule Markup Languages for the Semantic Web, 2004, pp. 81–97.
8. Erdmann M. Semantic web applications. 2007, first Asian Autumn School on Semantic Web, Tutorial video at: http://rease.semanticweb.org/ubp/.
9. Gelfond M. and Lifschitz V. The stable model semantics for logic programming. In Proc. 5th Int. Conf. Logic Programming, 1988, pp. 1070–1080.
10. Grosof B.N. Prioritized conflict handling for logic programs. In Proc. 14th Int. Conf. Logic Programming, 1997, pp. 197–211.
11. Grosof B. Semantic web rules with ontologies, and their e-Services applications. 2006, iSWC06 tutorial video at: http://videolectures.net/iswc06_grosof_swrot/.
12. Grosof B., Horrocks I., Volz R., and Decker S. Description logic programs: combining logic programs with description logic. In Proc. 12th Int. World Wide Web Conference, 2003, pp. 48–57.
13. Kifer M. Rules and ontologies in F-logic. In Reasoning Web, N. Eisinger and J. Małuszyński (eds.). LCNS, vol. 3564, 2005, pp. 22–34.
14. Knorr M., Alferes J.J., and Hitzler P. A well-founded semantics for hybrid MKNF knowledge bases. In Proc. 20th Int. Workshop on Description Logics, 2007, pp. 347–354.
15. Levy A. and Rousset M.C. CARIN: a representation language combining horn rules and description logics. Artif. Intell., 104 (1–2):165–209, 1998.
16. Motik B. and Rosati R. A faithful integration of description logics with logic programming. In Proc. 20th Int. Joint Conf. on AI, 2007, pp. 477–482.
17. Motik B., Sattler U., and Studer R. Query answering for OWL-DL with rules. J. Web Sem., 3(1):41–60, 2005.
18. Nilsson U. and Małuszyński J. Logic, Programming and Prolog, 2nd edn. Wiley, NY, 1995, now available free of charge at: http://www.ida.liu.se/ ulfni/lpp/.
19. Nute D. Defeasible logic. In Handbook of Logic in Artificial Intelligence and Logic Programming, Vol. 3. Oxford University Press, Oxford, 1994, pp. 353–395.
20. Rosati R. *DL*+log: tight integration of description logics and disjunctive datalog. In Proc. 10th Int. Conf. Principles of Knowledge Representation and Reasoning, 2006, pp. 68–78.
21. Sintek M. and Decker S. TRIPLE – a query, inference, and transformation language for the semantic web. In 2002, pp. 364–378
22. van Gelder A., Ross K.A., and Schlipf J.S. Unfounded sets and well-founded semantics for general logic programs. In Proc. 7th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems, 1988, pp. 221–230.

Intellectual Property

► Copyright Issues in Databases
► European Law in Databases

Intelligent Disks

► Active Storage

Intelligent Storage

► Intelligent Storage Systems

Intelligent Storage Systems

Kazuo Goda
The University of Tokyo, Tokyo, Japan

Synonyms

Intelligent Storage

Definition

The term Intelligent Storage System is a general term used to describe a storage system which has the capability of fully or partially realizing functions that used to be or are usually implemented on host computers.

Historical Background

The idea behind Intelligent Storage Systems may have its origin in the early researches on database machines. Similar ideas have continued to be studied in the academic communities to date, and they have recently been partially applied in commercial storage systems.

Foundations

The basic ideas of implementing full or partial application code on controller processors of disk drives may be traced back to the database machines which were actively studied in the 1970s and 1980s. The database machine was an approach of special hardware solutions. The early researchers focused on the development of filter processors, which could do selection operations closely to disk drives so as to obtain strong performance benefits. Several prototypes such as CASSM (University of Florida) [11] and RAP (Ohio University) [7] were implemented in the 1970s. Filter processors were at times coupled with a front-end database server. Such ideas called backend processors [3] were attempted in the industry. When it came to

the 1980s, with new algorithms such as hash-based joins, researchers proposed new database machines that pursued intensive parallel processing such as GRACE (The University of Tokyo) [6] and GAMMA (University of Wisconsin) [4]. Parallel machines that were specially designed for database processing were released and then resulted in commercial success. However, the solution of utilizing dedicated hardware lost its unique advantage by the 1990s, since powerful general-purpose machines became easily available. Major database vendors shifted to software-level solutions which used generic hardware instead.

Intelligent Storage Systems were again brought under the spotlight in the late 1990s. Storage network technologies such as Fibre Channel launched in the market. Then enterprise systems began to deploy the storage-centric system architecture, where storage systems could be designed and managed independently from host computers at the infrastructure level. Naturally, sophisticated new functions such as virtualization were being incorporated into storage processors and such solutions were widely accepted. Around the same time, Active Storage [8], Intelligent Disks [5] and Active Disks [1] were published in the academia in 1998. These were trying to exploit the capability of disk processors for data intensive applications such as ad-hoc query processing and image processing. The attempt of active storage looked similar to the database machine, but they carefully discussed software frameworks for running application code on disk processors.

After the twenty-first century began, storage networking has been practiced in many systems and storage resources are being consolidated more. A variety of new functions are being implemented in commercial storage systems. Functions that used to be run on host computers such as volume copy, remote replication and snapshot generation are usually executed in storage processors. Although only limited types of low-level applications are currently implemented in storage systems, the application domain is gradually being widened by the active research and development.

The motivation behind database machines and active storage was mainly in significant performance improvement. By processing data more closely to disk platters, they tried to efficiently exploit the limited bandwidth between main memory and storage systems. This "storage wall" is seen even in recent enterprise systems and thus such solutions are still beneficial. At the same time, in the light of the complexity of recent enterprise systems, Intelligent Storage Systems may have another substantial benefit. That is, storage-level implementation could improve the function-level isolation between components. This would be very helpful for system administrators to design and manage the complicated system.

Discussion on interface standards is a crucial point for realizing Intelligent Storage Systems. OSD (Object Storage Device) [2] has evolved out of the NASD (Network-Attached Secure Disk) project which started in Carnegie Mellon University in 1995. In contrast to traditional storage devices, where the storage space is represented as an array of fixed-size blocks, OSD works as a container of objects, their attributes and their metadata. Specifically, OSD can be seen as a storage device in which lower layers of file systems are implemented. The interface protocol of OSD, designed as an extension of SCSI, has been standardized as ANSI T10 SCSI OSD. Several NAS products and distributed files systems have already supported OSDs as backend storage devices. SNIA, a leading industry association of storage networks, has also promoted standardization. SMI-S (Storage Management Initiative-Specification) [9] is the standard protocol for storage management, which improves the interoperability between different storage devices, switches and management applications of different manufacturers. SNIA has developed and maintained SMI-S and has provided vendors with certification programs. XAM (eXtensible Access Method) [10] is another task operated by SNIA. XAM, a new suite of APIs, would provide an abstraction layer between storage devices which store fixed contents and management applications which access those contents.

Key Applications

Recent commercial storage systems have deployed storage-side implementation of simple functions such as data conversion between main frames and open systems, third-party copy, remote replication and snapshot generation. These were so far implemented only in top-end storage systems, but they are also being implemented in mid-range products and sometimes even in entry-level products.

Cross-references

- ▶ Active Storage
- ▶ Database Machine
- ▶ Network-Attached Secure Device

► Storage Network Architecture
► Storage Management

Recommended Reading

1. Acharya A., Uysal M., and Saltz J.H. Active disks: programming model, algorithms and evaluation. In Proc. 8th Int. Conf. on Architectural Support for Programming Languages and Operating Systems, 1998, pp. 81–91.
2. ANSI. Information Technology - SCSI Object-Based Storage Device Commands (OSD). Standard ANSI/INCITS 400–2004. 2004.
3. Canaday R.H., Harrison R.D., Ivie E.L., Ryder J.L., and Wehr L.A. A back-end computer for data base management. Commun. ACM, 17(10):575–582, 1974.
4. DeWitt D.J., Gerber R.H., Graefe G., Heytens M.L., Kumar K.B., and Muralikrishna M. GAMMA – a high performance dataflow database machine. In Proc. 12th Int. Conf. on Very Large Data Bases, 1986, pp. 228–237.
5. Keeton K., Patterson D.A., and Hellerstein J.M. A case for intelligent disks (IDISKs). ACM SIGMOD Rec., 27(3):42–52, 1998.
6. Kitsuregawa M., Tanaka Hi., Moto-Oka T. Application of hash to data base machine and its architecture. New Generation Comput, 1(1):63–74, 1983.
7. Ozkarahan E.A., Schuster S.A., and Smith K.C. RAP - An associative processor for database management. In Proc. National Computer Conf., 1975, pp. 379–387.
8. Riedel E., Gibson G.A., and Faloutsos C. Active storage for large-scale data mining and multimedia. In Proc. 24th Int. Conf. on Very Large Data Bases, 1998, pp. 62–73.
9. SNIA Storage Management Initiative. Storage Management Technical Specification, Overview Version 1.2.0, Revision 6. 2007.
10. SNIA XAM Initiative. XAM Initiative Overview, 2007.
11. Su S.Y.W. and Lipovski G.J. CASSM: a cellular system for very large data bases. In Proc. 1st Int. Conf. on Very Data Bases, 1975, pp. 456–472.

Interaction Design

► Human-Computer Interaction

Interactive Capture

► Visual Interfaces for Geographic Data

Interactive Information Exploration

► Information Navigation

Interactive Layout

► Visual Interfaces for Geographic Data

Interactive Visual Exploration of Multidimensional Data

► Visual On-line Analytical Processing (OLAP)

Interface

Patrick Eugster
Purdue University, West Lafayette, IN, USA

Definition

An interface describes the functionalities exported by an entity such as a software module. These functionalities typically consist in named operations with signatures describing potential arguments and return values, but interfaces may also include the definitions of data types, constants, exceptions, or even describe semantics.

Interfaces shield the internals of corresponding software modules from the outside, providing several benefits. Interfaces provide abstraction, in the sense that the internals of modules may evolve while other modules can still rely on the same functionalities (encapsulation). Safety and security are promoted by interfaces as these define single access points to respective software modules.

In object-oriented programming, software modules often coincide with classes, which also describe data types implicitly. In the case of such a class, an interface is thus roughly made up of the methods exported by the class. Mostly, interfaces are then defined implicitly by the classes as such sets of exported methods, but languages such as Java provide programmers the possibility of declaring interfaces as first class citizens which are then explicitly implemented by classes.

When objects are physically distributed, they can offer specific remote interfaces facilitating invocations potentially coming from remote hosts. Remote interfaces span typically only a subset of all the methods exported by objects.

A set of interfaces which describe an entire software component or framework is more commonly referred to as an application programming interface (API).

Historical Background

The concept of interface has partly evolved out of the *header files* used with the C programming language. In contrast to several more recent programming languages providing notions of interfaces integrated with the language semantics, those header files served mainly as preprocessing feature.

Interfaces were greatly popularized by programming languages based on *modules*, such as Pascal, Modula-2, or Ada. A first step towards a more rigorous underpinning of modules and measures for a good decomposition of software into such modules was provided by Parnas [8]. Parnas' work primarily aimed at achieving a clean separation of duties in software components and an effective decomposition into modules. As a dual of this problem, the design process for interfaces to modules has also been strongly guided by Parnas' seminal work, coining the term "well-defined interfaces." The main driving force behind the quest for clear-cut interfaces are safety/security concerns aiming at providing well-defined entry points to given modules in the form of interfaces.

The term *interface* itself thus emerged from the software engineering community. Ever since, programming languages have been strongly influenced by this notion, in particular because there is a strong overlapping between the notions of interface and *type* (*data type*). This culminates in object-oriented programming, where data types and modules are largely unified through classes representing the units for both data abstraction and behavior, and thus a type defines the interface to its instances, even if only implicitly (cf. [2]).

More recently, modeling languages such as the Unified Modeling Language (UML) have also introduced interfaces aside classes as purely descriptive means of capturing functionalities of objects.

Foundations

Interfaces describe the functionalities exported by an entity such as a software module. These functionalities typically consist in named operations with signatures describing potential arguments and return values, but interfaces may also include the definitions of data types, constants, exceptions, or even more information.

Interfaces and Classes

Most object-oriented programming languages are class-based, meaning that classes present the main unit of decomposition and as such can be viewed as modules. Interfaces of such classes encompass the functionalities exported by these classes – typically a set of methods which are accessible to other classes. In many integrated development environments (IDEs), there is the possibility to automatically "extract" such an interface view from a class, by only summarizing the names and signatures of its exported features.

Interfaces can be in that sense seen as abstract types, as opposed to classes which may *implicitly* describe interfaces, but primarily describe the implementation of those. Current programming languages such as Java provide interfaces as a first class construct aside classes. Declaring an `interface` leads to defining an abstract type. In this case, an object can provide several interfaces if its class implements several instances; the *conceptual* interface of the instance(s) is then made up of the union of these interfaces.

When collapsed with types, interfaces reap the benefits of the Liskov substitution principle [6], which states rules under which a type `T1` can be considered to be compliant with another type `T2`. Instances of `T1` can then be used whenever entities of type `T2` are expected. Transposed to interfaces this means in short that the interface of the latter type subsumes the interface of the former type: any module that builds on the former interface can be presented with an instance of the latter interface.

Protection

Conceptual interfaces are sometimes decomposed according to potential beholders of references to the corresponding entities. More precisely, different objects, or more generally different modules, can have access to the same entities through individual interfaces. By supporting different visibility rules (e.g., `private`, `public`, `protected`), possibly at the method level, programming languages provide the ability to restrict interfaces (or parts of such interfaces) to families of classes for example based on containment criteria (e.g., packages). In C++ for instance, a class `C1` can be explicitly declared to be a `friend` of another class `C2`, providing instances of `C1` access to certain methods of class `C2` which are otherwise unreachable. Introducing varied levels of protection yields different (accessible) interfaces for different parties.

For languages with weaker inherent support for interfaces and protection levels, design patterns can be used to enforce constraints. The *read-only interface* pattern for instance is used to explicitly define an interface through which entities can be manipulated without modifications to their state.

Remote Interfaces

In remote method invocations, *remote interfaces* are used to explicitly offer methods for invocation from remote hosts. Remote interfaces usually span only a subset of the methods provided by a class. The remaining methods however are limited to invocations within the same address space. In that sense, remote interfaces introduce a specific level of protection.

Remote interfaces commonly enjoy descriptions in a dedicated specification language, when interoperability is desired. Such specification languages allow the description of remote interfaces in an interoperable format, and subsequent generation of language-specific versions with corresponding compilers based on well-defined mappings to target languages. Examples are the *interface definition language* (IDL) used in the Common Object Request Broker Architecture (CORBA), or more recently the *web service description language* (WSDL). An earlier incarnation is the *external data representation* (XDR) introduced with the remote procedure call (RPC) protocol. Compilation of such interfaces commonly goes hand in hand with the generation of proxes or stubs which take care of transforming method invocation arguments and return values to and from an interoperable format such as XDR.

Contents

Besides methods, interfaces can also include constants, exceptions, and sometimes even fields, though exporting fields is sometimes viewed as going against the principle of encapsulation underlying object-oriented programming. In the Eiffel programming language, methods are unified with field (accesses), implicitly yielding access methods for fields. Eiffel is also the most popular language with further *semantic* descriptions in the interfaces, consisting in *contracts* in the form of pre- and postconditions for methods, as well as invariants at the class level. These were introduced as part of the original definition of *abstract data type* [6], and have given rise to a design paradigm called *design by contract* (DbC) [7].

In the case of Eiffel, its IDE also allows the programmer to view the interface of a given class, in this case including contracts. Contracts have a descriptive flavor in the sense that they augment interfaces and thus represent a specification feature, but also have an implementation flavor since they may refer to fields, and can be monitored at runtime as in Eiffel. Spec# follows the Eiffel approach by providing first class support, while the Java Modeling Language (JML) promotes annotations, which can be viewed as optional semantic interface descriptions, to which programmers only must adhere when making use of specific tools (e.g., for compilation).

Besides contracts, other types of information have been considered for augmenting interfaces. *Typestates* [9] capture abstract states associated with instances of a given type and their relationship with exported methods, thus adding state information to interfaces. *Interface automata* [4] focus directly on the *order* in which functionalities exposed by a given interface can be triggered. Such sequences are sometimes also colloquially referred to as *protocols*. *Resource interfaces* [3] are another example of augmented interfaces, describing namely the physical resources necessary for a given module to be able to function properly.

Components

In software engineering, in particular in the *component* view, interfaces may similarly contain far more information than types. A component usually refers to a subsystem of its own consisting in several modules, whose individual interfaces make up the conceptual interface. The interface of a component can for instance also define what kind of interface the component itself relies upon (while this is usually somewhat embedded in object-oriented code).

When several interfaces are bundled either in the context of a component, or a framework, the term *application programming interface* (API) is commonly employed. The different levels of technical protection for functionalities offered at the programming language level are in the case of APIs often complemented by simply limiting the disclosure of APIs to selected corporate players only.

Key Applications

There are various reasons for further studying the domain of interfaces:

- Complexity. With the increasing complexity of computer systems, the problem of "efficiently" decomposing/composing software is given more and more attention. Abstraction and encapsulation are key desired properties. In software engineering terms, one attempts to achieve decoupling and increase modularity (For a precise definition of modularity see [8]). The art of decomposing software is also partly captured by the study of *software architectures*. There is a clear dependency between the decomposition of software into modules and the discipline of designing interfaces between modules/components to respect modularity and achieve low coupling.
- Reuse and maintainability. These are also desired features of any software package. With good design and modularity reflected in rich interfaces, code namely also becomes easier to reuse and maintain.
- Safety. Safety has become a major driving force for programming language and software engineering research. Ensuring that interaction between (sub-) modules takes place in a safe manner helps avoiding runtime errors, and is supported by clear-cut and precise interfaces. In the predominant settings with static typing, the goal is to ensure that once successfully compiled, a module contains no calls violating interfaces.
- Security. Access rights and confinement can be expressed in interfaces as mentioned above. This becomes particularly visible in distributed settings, i.e., for achieving clean interfaces between entities running on distinct physical hosts.
- Interoperability. Not all software is written in the same programming language. With interfaces being described in "neutral" generic languages interoperability can be achieved. Examples are clearly provided by *interface definition languages* (IDLs) for second class distributed object packages such as DCE, CORBA, or DCOM. The idea of interoperability has more recently flown into **service-oriented architectures** (SOA) in general and **Web Services** in particular.

Future Directions

Motivated by the above, there is quite some incentive to further investigate interfaces. Various aspects offer themselves for additional efforts:

- *Evolution.* Low coupling and modularity definitely support evolution of components, but as mentioned above, only in terms of the internals of components. Evolution of components sometimes also necessitates evolution of their interfaces, which can however break any existing code relying on these. Evolution of interfaces is thus an important area to investigate. In programming language terms, this leads to the problem of extending and changing types, which can be partly addressed by introducing versioning, and by assisting the programmer in the adaptation of code after changes to interfaces it relies upon.
- *Information.* Interfaces describe contracts between modules, i.e., functionalities provided by a module and sometimes also requirements for providing these functionalities. So far, interfaces are mostly described through functional aspects of the modules which in programming language terms boils down to syntactic and typing information. Design by contract (DbC) augments these interfaces with semantics, but further information might be of relevance, which is not even directly materialized in the implementation of modules, as illustrated by resource interfaces [3].
- *Discovery.* Interfaces and extended interfaces can also be obtained by *mining* existing modules, e.g., [1]. This is particularly appealing when dealing with legacy code, which might have been described in a programming language with weak inherent support for interface descriptions. Discovery of interfaces has also gained more interest recently in the realm of *aspect-oriented programming* (AOP), whenever aspects advise code *obliviously*, i.e., without knowledge or consent from the main code. In the AOP philosophy aspects attempt to achieve a separation of duty by implementing *crosscutting concerns* in aspects alike modules, separately from the base code. Without clean interfaces for the interaction between base code and aspects however, reasoning about an entire software becomes hard (cf. [5]).

  Discovery can also be understood in the sense of discovering services at runtime. Especially SOAs provide strong support for finding services based on their functionalities, often described as some abstract interfaces. UDDI is an example of a lookup service that supports queries based on a specific notion of interface.

Cross-references

- ▶ Discovery
- ▶ Request Broker
- ▶ Service Oriented Architecture
- ▶ Web Services

Recommended Reading

1. Beyer D., Henzinger T.A., and Singh V. Algorithms for interface synthesis. In Proc. 19th Int. Conf. on Computer Aided Verification, 2007, pp. 4–19.
2. Canning P.S., Cook W.R., Hill W.L., and Olthoff W.G. Interfaces for strongly-typed object-oriented programming. ACM SIGPLAN Not., 24(10):457–467, 1989.
3. Chakrabarti A., de Alfaro L., Henzinger T.A., and Stoelinga M. Resource interfaces. In Proc. 3rd Int. Conf. on Embedded Software, 2003, pp. 117–133.
4. de Alfaro L. and Henzinger T.A. Interface automata. In Proc. 9th ACM SIGSOFT Int. Symp. on Foundations of Software Eng., 2001, pp. 109–120.
5. Griswold W.G., Sullivan K.J., Song W., Shonle M., Tewari N., Cai Y., and Rajan H. Modular software design with crosscutting interfaces. IEEE Softw., 23(1):51–60, 2006.
6. Liskov B.H. and Wing J.M. A behavioral notion of subtyping. ACM Trans. Program. Lang. Syst., 16(6):1811–1841, November 1994.
7. Meyer B. Applying design by contract. IEEE Comput., 25 (10):40–51, October 1992.
8. Parnas D.L. On the criteria to be used in decomposing systems into modules. Commun. ACM, 15(12):1053–1058, 1972.
9. Strom R.E. and Yemini S. Typestate: a programming language concept for enhancing software reliability. IEEE Trans. Softw. Eng., 12(1):157–171, 1986.

Interface Engines in Healthcare

Dan Russler
Oracle Corporation, Redwood Shores, CA, USA

Synonyms

Transformation engines; Mapping engines; Messaging engines; Service buses

Definition

A computer application that supports the transformation of the syntactic and semantic structures in communication content during transmission from a sending system to receiving system(s), ensuring reliable delivery of the communication and minimizing information loss and semantic shift during the communication.

Historical Background

Before the invention of application programming interfaces (API) and CORBA Interface Definition Language (IDL) files, "interface" was the term used to describe the electronic communication of information between two computers [1]. Today, the term "interface" also refers to communications between layers of software and even between software objects within a software layer. A modern example of an interface is a Web Service Definition Language (WSDL) file in XML format (www.w3.org).

Within early interfaces, the syntax of the communication content, i.e., linear arrangement of characters in the communication exported by the sending system, often could NOT be imported directly by the receiving system. Consequently, transformation procedures were employed to rearrange the characters in the communication into a linear structure that COULD be imported by the receiving system. In the same manner, if terms used by the sending systems could not be imported into the receiving system, a substitution of terms that could be imported by the receiving system was also applied to the communication content.

In order to reduce the burden of computing these transformations on the slower computers of the past, the execution of these transformation procedures was transferred to a separate computer that was placed in-between the sending and receiving computers. As time passed, the ability of these "interface systems" to support better and easier transformation authoring and high "transaction" or messaging volumes, these systems became known in the 1990s as "interface engines," a new kind of software application in themselves [4,5].

As the language of the industry evolved, the communications between computers became known as "messages"; the transformation of the syntactic and semantic content became known as "mapping" from sender to receiver; and techniques for ensuring reliable receipt of communications between a sending and receiving computer became known as "reliable delivery protocols" or "reliable messaging protocols."

However, despite the increasing sophistication of the transformation tools, the industry soon discovered that there was a limited ability to ensure that all the information sent by the sending system could be imported by the receiving system [7]. Information loss and shift in meaning of the communication content was observed when evaluating the information content of the sending and

receiving systems after the communication occurred. In many ways, the result was similar to the garbling of sentences that occurs in the children's games that test the ability of children sitting in a ring to sequentially whisper a sentence into the ear of the next child. The child who initiates the sentence rarely gets the same sentence whispered back by the last child.

As a consequence, organizations that developed standards in support of many other industries began to support standardized messaging structures for the computer industry, including message standards in the healthcare industry. These standards included both the arrangement of "fields" and special characters in the messages and the sets of terms used to populate these fields. In healthcare, the messages most widely used internationally by the year 2000 were authored by a specialized healthcare standards organization, the Health Level 7 (HL7) standards development organization (www.hl7.org), which focused on ISO Level 7 transaction protocols [1].

These messaging standards reduced the cost of each messaging "interface" between computers by as much as tenfold from the 1980s to the turn of the century. However, the implementation cost of these messaging interfaces (HL7 version 2.x messages) continued to retail at over $20,000 per interface (in addition to the cost of the interface engine), and many hospitals required over one hundred messaging interfaces.

By the early 1990s, planners in HL7 began exploring model-based development methods and new message authoring techniques that would both improve the quality and reduce the cost of communications between computer systems in healthcare. The improvement in quality of communication refers to the preservation of information content and semantic meaning of the communication or what is known as "semantic interoperability." The reduction in cost comes from decreasing the number of choices allowed to developers who "interpret" the standards into actual application code. The result of this planning effort was the publication of the HL7 Reference Information Model (RIM) [6] and RIM-derived messages, electronic documents [2], and web services.

Parallel efforts in other industries as well as the growth of Internet communications between computer systems have caused rapid evolution in both the kinds of communications the healthcare industry wishes to utilize and the techniques for electronically communicating between computer systems.

As the result of many initiatives across industries, there have been great strides in communication methods that no longer utilize the traditional, preconfigured point-to-point HL7 interface engines. The concept of a "bus" was borrowed from the internal computer bus that supports the physical "plug & play" communication of multiple physical components (such as hard drives) within a computer. This concept of a "bus" was abstracted to a "service bus" located within a data center that allows dynamic selection of multiple web services in a service-oriented architecture. As a result, interface communication in a data center no longer needs to rely on a "point-to-point" pre-configured solution. Rather, in a service bus, a "service directory" may be used to dynamically select the method of communication and the endpoint(s) of communication desired. Increasingly, these new kinds of "messaging engines" or "enterprise service buses" are being utilized in healthcare data centers. Finally, "healthcare service bus" is a term used to describe a "virtual service bus" or a system of "federated enterprise service buses" where dynamic web services are supported between individual enterprises and enable healthcare communications across the wider community (www.openhealthtools.org).

Although many people believe that legacy computer systems and traditional, preconfigured point-to-point messaging methods will continue to be used for many years in healthcare, the consequence of new communication techniques used by many other industries is that communication methods in healthcare will also evolve. The role of the traditional data-center-based, HL7 2.x interface engine will gradually be reduced in favor of web-service-based communications that support transformation and routing across healthcare communities.

Foundations

The scientific study of messaging concepts begins with narrow, reductionist models, e.g., the physics of electrons and gravitational bonds communicating between atoms and within molecules. These concepts are studied within incrementally more complex systems in chemistry labs, organic chemistry labs, genetic and hormonal communications, electronic systems and neural communications, human communication, and finally, in computer systems that support human communication.

The study of messaging was enhanced by the development of communication models, many of which include the concept of state machines. Communication

may be defined as "transferring awareness of a change of state in the model of the sending system to the receiving system(s)." This communication may be as simple as informing the receiver of the increase in the state of ionic attraction that occurs when an atom has gained or lost an electron or as complex as informing a second organization that the state of a hospital now includes a patient ready for placement in a nursing home.

The concept of a "state machine" was introduced in the mathematical modeling and electronics literature. A "state machine" was a systems model developed in the 1950s and 1960s to describe state transitions in sequential circuits [3]. Generally, finite state machines describe a model wherein a "state" describes the static "snapshot" or configuration of elements within a modeled system. The visual image generated by the use of the term "machine" is that of a machine moving while an observer takes sequential snapshots, each of which illustrates the machine in different configurations or states. A "process" is the change in configuration or transformation of elements in the state machine, i.e., "state transition." "Event-driven state machines" highlight the trigger events used in event driven programming and messaging models; one visualizes an operator pushing buttons on the machine. And communication can then be described as the transferring of awareness between two or more systems, specifically, the awareness of the state transition of the sending system. One visualizes sending a picture of the new state of the machine, or perhaps a picture of the new state and a picture of the former state, as soon as the operator pushes the button on the machine.

A trigger event in healthcare is often a clinical observation result on a patient, a clinical order, or other clinical event. Observation results by clinicians characterize the clinical state of a patient during a specific time period. Changes in the clinical state of the patient are tracked by obtaining sequential observation results. When a transition occurs in the clinical state of a patient, the change may trigger a message alerting a physician. A new clinical order by the physician may trigger a message to the lab requesting a new lab test.

In the same manner, changes in the clinical state of the patient may be recorded in an electronic medical record system. The resulting state transition in the electronic medical record system may trigger a message to another electronic system. As illustrated, state transitions in electronic medical record systems are closely related to the clinical state changes in the patient and the awareness of care providers about the clinical state of the patient.

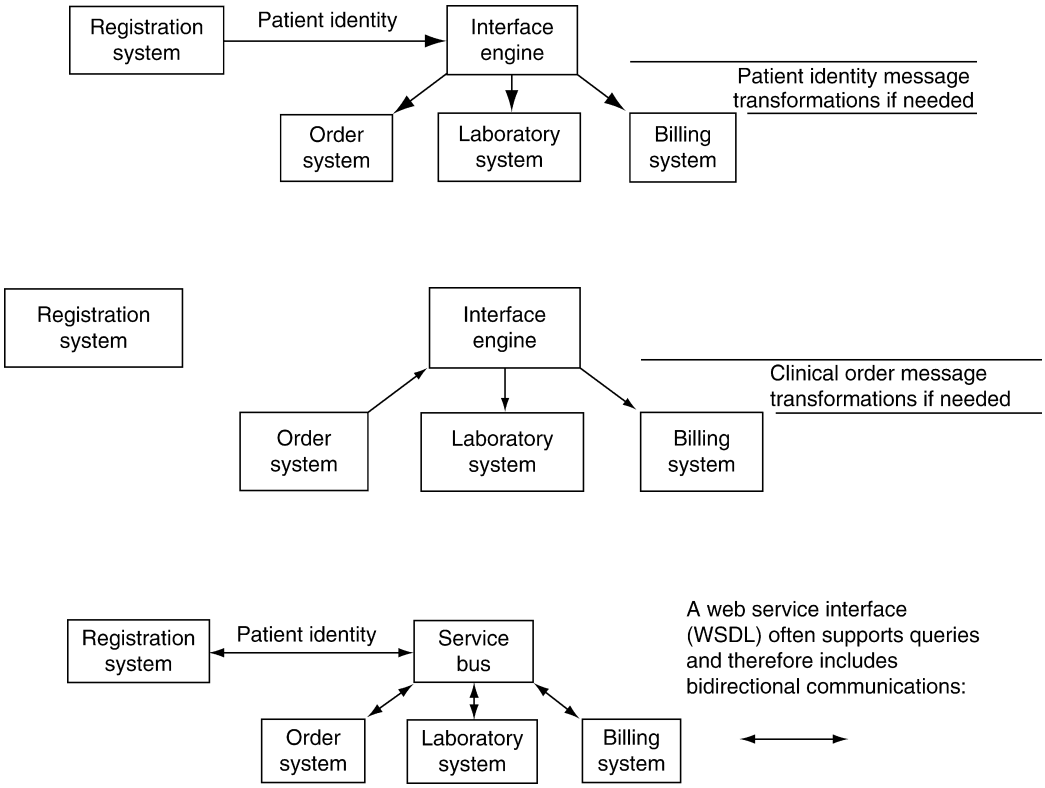
Finally, the study of the science of state changes, the communication of state changes, and the relationship of state changes to electronic communications in healthcare has been applied to more sophisticated techniques for dynamically orchestrating these communications into optimized process flows. Web service orchestrators have evolved as components of data integration engines such as service buses in healthcare. And optimized patient care processes are increasingly implemented with web service orchestrated electronic health record activities (www.infoway-inforoute.ca).

Key Applications

The most common first step in the installation of Hospital Information Systems (HIS) is to establish an identity system for enrolling patients as they enter the hospital, commonly referred to as a "registration system." When the registration system records a new patient identity or updates an older patient's demographic information, data about the patient is communicated via messages to other systems, such as order management systems, laboratory systems, and billing systems. This data includes a patient identifier and at least the first name, last name, date-of-birth, gender, and address. Traditionally, point-to-point interfaces are pre-configured within an interface engine from the registration system to the other systems, allowing broadcast of the identity of the new patient or updated demographic information to the other systems. If needed by any of the receiving systems, transformations may be applied to the syntax and semantics of the message that is outbound from the interface engine to each specific receiving system:

Later, if a clinical order is generated within the Order Management System for a patient, a subset of the fields in the Patient Identity message will be included in the clinical order and sent to the laboratory system and billing system via the interface engine. Again, transformations of the message may occur if needed by the receiving systems:

What is illustrated in the two figures above is that in a traditional interface engine scenario, the Registration System is never queried for additional Patient Identity information. Therefore, Patient Identity information is redundantly stored within the other systems. However, as populations managed by systems grow,



the inefficiency of redundantly storing Patient Identity information grows as well. As a consequence, larger healthcare systems, such as regional or community-sized systems, are evolving towards a "just in time" communication of Patient Identity. A "service bus" replaces the Interface Engine, and the Registration System is dynamically queried as needed for additional Patient Identity information:

Cross-references

- ▶ Clinical Event
- ▶ Clinical Observation
- ▶ Clinical Order

Recommended Reading

1. Collen M. A History of Medical Informatics in the United States, 1950 to 1990. American Medical Informatics Association, Bethesda, MD, 1995.
2. Dolin R. et al. HL7 clinical document architecture, release 2. J. Am. Med. Inform. Assoc., 13(1):30–39, 2006.
3. Gill A. Introduction to the Theory of Finite-state Machines. McGraw-Hill, New York, 1962.
4. Lenz R. et al. A practical approach to process support in health information systems. J. Am. Med. Inform. Assoc., 9(6):571–585, 2002.
5. McDonald C. The barriers to electronic medical record systems and how to overcome them. J. Am. Med. Inform. Assoc., 4(3):213–221, 1997.
6. Russler D. et al. Influences of the unified service action model on the HL7 reference information model. In Proc. Symp. on Computer Applications in Medical Care, 1999, pp. 930–934.
7. White T. et al. Extending the LOINC conceptual schema to support standardized assessment instruments. J. Am. Med. Inform. Assoc., 9(6):586–599, 2002.

Internet Transactions

▶ Web Transactions

Interoperability in Data Warehouses

Riccardo Torlone
University of Rome, Rome, Italy

Synonyms

Data warehouse integration

Definition

The term refers to the ability of combining the content of two or more heterogeneous data warehouses, for the purpose of cross-analysis. This need emerges in a variety of practical situations. For instance, when different designers of a large company develop their data marts independently, or when different organizations involved in the same project need to integrate their data warehouses.

Data Warehouse interoperability is a special case of the general problem of database integration, but it can be tackled in a more systematic way because data warehouses are structured in a rather uniform way, along the widely accepted concepts of dimension and fact. As it happens in the general case, different degrees of interoperability can be pursued by adopting standards and/or by applying reconciliation techniques, likely specific for this context.

The problem is becoming increasingly relevant with the spreading of federated architectures. Nevertheless, it has been the focus of a few systematic works and numerous open problems remain to be solved.

Historical Background

In spite of its relevance, the problem of data warehouse integration has received little attention so far. Conversely, the general problem of databases integration has been studied in the literature extensively and several aspects, both at scheme and instance level, have been deeply investigated, such as the automatic matching of terms and the resolution of structural conflicts (see [8,12] for surveys on these topics).

In the specific context of data warehouses, Kimball [7] has identified the problem for the first time: he has investigated the integration of heterogeneous dimensions in a scenario of data warehouse design and has introduced the informal notions of dimension *conformity*. Intuitively, two dimensions are conformed if their share some information in a consistent way. This is an important requirement in *drill-across queries*, which are basically joins of different facts over common dimensions. The notion of conformity has been formalized and extended by Cabibbo and Torlone in the context of data mart integration [4] under the name of dimension *compatibility*: they have demonstrated that this property gives the ability to perform correct drill-across queries over heterogeneous data marts.

An issue related to the integration of data warehouses, which has been studied in the context of statistical databases, is the *derivability* of summary data. This notion has been defined by Sato [14] as the problem of deciding whether a summary data (which is, in a statistical database, the counterpart of a fact table) can be inferred from another summary data aggregated in a different way. The concept has been extended by Malvestuto [9], by considering the case in which the source is composed by several heterogeneous data sets: he proposes an algebraic approach to this problem and provides some necessary and sufficient conditions of derivability. Unfortunately, statistical databases have some similarity with multidimensional databases, but also some important diversities: this makes the application of these approaches to data warehouses not easy.

Some related work has been done on the problem of integrating a data warehouse with external data stored in XML [6] and in object-oriented [11] format, but just a few works have been devoted to the specific problem of the interoperability between heterogeneous data warehouses. They will be discussed in the following section.

While current commercial tools do not provide a complete support for data warehouse interoperability, they offer facilities that can be very useful in this framework, such as metadata import/export (using XML) and standardized ways to represent data (using a multidimensional model).

Foundations

Since data warehouse interoperability can be considered a special case of the problem of database integration, general data reconciliation techniques can be often used. For instance, methods for the automatic matching of terms or for the resolution of structural conflicts. In addition, it is possible to take advantage on the fact that, in this context, the data sources always have a multidimensional structure. Therefore, the problem can be addressed by focusing on the reconciliation of heterogeneous dimensions and facts. Following this observation, the section discusses: standards that can be adopted to support data warehouse interoperability, conflicts that can arise in this context, and methodologies that can be used to perform the integration.

Standards

An important support for interoperability can be provided by the adoption of standards. Initially, two industry standards have been proposed by multi-vendor organizations for data warehouses: the Open Information Model (OIM) developed by the Meta Data

Coalition (MDC), and the Common Warehouse Metamodel (CWM) developed by the Object Management Group (OMG) [16]. Later, MDC and OMG joined their efforts and proposed a new version of the CWM as the standard metadata model. The Common Warehouse Metamodel is a platform-independent specification for exchanging multidimensional data between different platforms and tools. It is based on the standards UML, XMI, and MOF, and provides a set of generic, external representations of metadata, called *metamodels*, that provide a comprehensive framework for data exchange. These metamodels can be used to describe the various components of the data warehouse architecture: data sources, ETL processes, multidimensional cubes, relational tables, and so on. However, it has been observed that their expressivity is not sufficient to capture all the complex semantics of conceptual multidimensional models, so they hardly can be used for effective integration of different data warehouses [13].

Conflicts

In the integration of different multidimensional data sources, a number of conflicts can arise, both at the schema and at the instance level.

- *Dimension conflicts*:
  - Schema: conflicts can arise on entity names (e.g., different names for the same dimensions and/or different names for similar levels of two dimensions) and on dimension hierarchies (similar dimensions organized over different levels of aggregation and/or inconsistencies on the roll-up relationships between levels).
  - Instance: conflicts can arise on member names (different names for the same members of different dimensions) and on the members of dimensions (similar dimensions populated by different members).
- *Fact conflicts*:
  - Schema: still, conflicts can arise on names (different names for the same measures) and on dimensions that differ in number and/or in the levels of aggregation.
  - Instance: conflicts can arise on measures (inconsistent values for the same measures and/or differences in scales).

As mentioned in the previous section, Cabibbo and Torlone [4] have identified a fundamental property that should be enforced while solving conflicts between heterogeneous data warehouses: dimension and fact *compatibility*. Two different dimensions d_1 and d_2 are compatible when their common information is consistent, that is, when aggregations computed over d_1 and d_2 and aggregations computed over the dimension obtained by merging d_1 and d_2 produce the same results. Having compatible dimensions and facts is important because it gives the ability to look consistently at data across data marts and to combine and correlate such data by means of drill across queries. Building on this notion, they have also identified a number of desirable properties that a *matching* between dimensions (that is, a correspondence between their levels) should satisfy: (i) the *coherence* of the hierarchies on levels, (ii) the *soundness* of the levels in correspondence, according to the members associated with them, and (iii) the *consistency* of the roll-up functions that relate members of different levels within the matched dimensions.

Integration Techniques

Two heterogeneous data warehouses can be combined if they share one or more dimensions and can be actually integrated if their facts can be joined, in a consistent way, over such common dimensions. It follows that a general methodology for achieving interoperability in data warehouses includes the following steps:

1. Identification of the facts that can be integrated and the dimensions of these facts that can be combined to perform the integration
2. Resolution of conflicts between common dimensions
3. Resolution of conflicts between facts to be integrated
4. Reconciliation and integration of dimensions and facts according to the desired level of interoperability

While this process can be supported by general reconciliation techniques based, for instance, on domain ontologies, it is possible to rely on specific techniques that take into account the rather standard structure of dimensions and facts. As usual, the level of interoperability can range from a scenario of loosely coupled integration, in which there is just the need to identify the common information between sources while preserving their autonomy, to a scenario of tightly coupled integration, in which the goal is rather merging the sources. In the former approach, queries are performed over a virtual view defined on the original

sources, in the latter, queries are performed against a materialized view built from the sources.

Banek et al. [1] have addressed the problem of matching schema structures specific to data warehouses, the initial step of the above methodology. Their approach consists of two basic tasks. First, similarity matches between multidimensional structures are identified by comparing their names, data types and substructures (e.g., matches cannot violate the partial order in hierarchies). Then, heuristic rules, based on graph similarity, are used to choose the actual mappings, among the possible matches.

A methodology for the resolution of conflicts that guides the designers through the combination of independent data cubes has been proposed by Berger and Schrefl [2]. They also propose a specific language called SQL-MDi (SQL for multi-dimensional integration), supporting the methodology. In their approach, the goal is the generation of a tightly coupled architecture that combine heterogeneous multidimensional data sources into a materialized warehouse.

Cabibbo and Torlone [5] have proposed two practical approaches to the integration of autonomous data warehouses that try to enforce matchings satisfying the properties discussed in the previous section and refer to the scenarios of loosely and tightly coupled integration, respectively. As a preliminary tool, they introduce a powerful technique, the *chase of dimensions*, that can be used in both approaches to test for consistency and combine the content of the dimensions to integrate. This technique operates over a tableau populated by the members of the dimensions to be integrated, and makes use of the roll-up functions defined over such dimensions. Two integration algorithms are then proposed. The first algorithm provides the operations, expressed in an abstract algebra, that applied to the original dimensions, allow the specification of correct drill-across joins between the heterogeneous sources. The second algorithm generates new dimensions and facts, obtained by merging the original data sources, that constitute the reconciled data warehouse.

From a practical point of view, a general federated architecture supporting the interoperability of distributed and autonomous data warehouses has been proposed by Mangisengi et al. [10]. Tseng and Chen [15] have proposed a framework in which, after a resolution of conflicts, autonomous data cubes are first transformed into XML documents, then conflicts are solved by means of XQuery operations, and finally the access to integrated data is achieved through queries posed over an XML global view.

Key Applications

A common practice for building a data warehouse is to implement a series of data marts, each of which provides a dimensional view of a single business process [7]. These data marts should be based on common dimensions but what happens in practice is that, very often, different departments of the same company develop their data marts independently. It turns out that methods and tools for data warehouse reconciliation are very useful in such common situation.

Indeed, the need for combining autonomous data warehouse arises in other common scenarios. For instance, when different companies merge or get involved in a federated project or when there is the need to combine a proprietary data warehouse with data available elsewhere, for instance, in external and likely heterogeneous information sources, or in multidimensional data wrapped from the Web.

Furthermore, methods supporting data warehouse interoperability can be useful when there is the need to migrate a data mart from one implementation platform to another.

Future Directions

The area of data warehouse interoperability is largely unexplored and there is still a compelling need of systematic studies and effective tools. From a conceptual point of view, the problem needs a deeper investigation that takes into account, for instance, cases in which the structure of the data warehouses to be combined is non standard (e.g., for the presence of non-strict hierarchies or many-to-many relationships between facts and dimensions). In particular, the presence of irregular hierarchies makes the problem of dimension compatibility much harder since it requires complex tests at instance level. From a practical point of view, there is still a lack of effective tools specifically supporting the integration of autonomous and heterogeneous data warehouses.

Experimental Results

A preliminary tool supporting the interoperability of data warehouses has been recently proposed [3].

Cross-references

► Common Warehouse Metamodel
► Data Mart

► Data Warehouse
► Data Warehousing Systems: Foundations and Architectures
► Data Integration
► Multidimensional Modeling

Recommended Reading

1. Banek M., Vrdoljak B., Min Tjoa A., and Skocir Z. Automating the schema matching process for heterogeneous data warehouses. In Proc. 9th Int. Conf. Data Warehousing and Knowledge Discovery, 2007, pp. 45–54.
2. Berger S. and Schrefl M. Analysing multi-dimensional data across autonomous data warehouses. In Proc. 8th Int. Conf. Data Warehousing and Knowledge Discovery, 2006, pp. 120–133.
3. Cabibbo L., Panella I., and Torlone R. DaWaII: a tool for the integration of autonomous data marts. In Proc. 22nd Int. Conf. on Data Engineering, Demo session, 2006.
4. Cabibbo L. and Torlone R. On the Integration of Autonomous Data Marts. In Proc. 16th Int. Conf. on Scientific and Statistical Database Management, 2004, pp. 223–234.
5. Cabibbo L. and Torlone R. Integrating heterogeneous multidimensional databases. In Proc. 17th Int. Conf. on Scientific and Statistical Database Management, 2005, pp. 205–214.
6. Jensen M.R., Møller T.M., and Pedersen T.B. Specifying OLAP Cubes on XML Data. J. Intell. Inf. Syst., 17(2–3):255–280, 2001.
7. Kimball R. and Ross M. The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling, Wiley, 2nd edn., 2002.
8. Lenzerini M. Data integration: a theoretical perspective. In Proc. 21st ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems, 2002, pp. 233–246.
9. Malvestuto F.M. The Classification Problem with Semantically Heterogeneous Data. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1988, pp. 157–176.
10. Mangisengi O., Huber J., Hawel C., and Eßmayr W. A framework for supporting interoperability of data warehouse islands using XML. In Proc. 3rd Int. Conf. Data Warehousing and Knowledge Discovery, 2001, pp. 328–338.
11. Pedersen T.B., Shoshani A., Gu J., and Jensen C.S. Extending OLAP querying to external object databases. In Proc. Int. Conf. on Information and Knowledge Management, 2000, pp. 405–413.
12. Rahm E. and Bernstein P.A. A survey of approaches to automatic schema matching. VLDB J., 10(4):334–350, 2001.
13. Rizzi S., Abelló A., Lechtenbörger J., and Trujillo J. Research in data warehouse modeling and design: dead or alive? In Proc. ACM 9th Int. Workshop on Data Warehousing and OLAP, 2006, pp. 3–10.
14. Sato H. Handling Summary Information in a Database: Derivability. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1981, pp. 98–107.
15. Tseng F.S.C. and Chen C.W. Integrating heterogeneous data warehouses using XML technologies J. Inf. Sci., 31(3):209–229, 2005.
16. Vetterli T., Vaduva A., and Staudt M. Metadata standards for data warehousing: open information model vs. common warehouse metamodel. ACM SIGMOD Rec., 29(3):68–75, 2000.

Interoperation of NLP-based Systems with Clinical Databases

Yves A. Lussier, Matthew G. Crowson
University of Chicago, Chicago, IL, USA

Synonyms

Semantic web

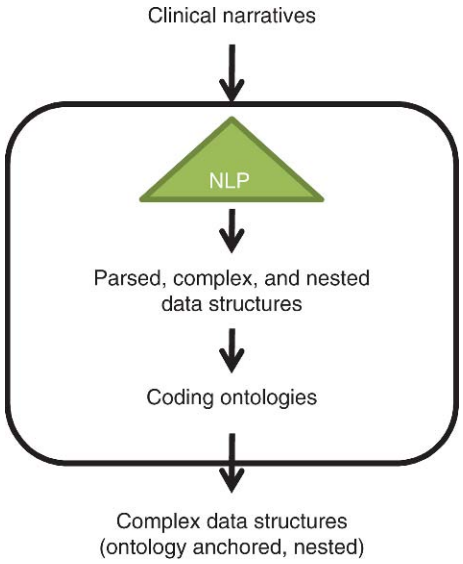
Definition

Natural language processing (NLP) is the automation of processes to interpret and understand meaning in human communications. In the life sciences, NLP assists in wide-scale storage and retrieval of specific "bundles" of clinical data embedded in patient charts which are commonly "free text". Both expert-system and statistical based NLPs have been in use in biomedicine for over three decades and some have shown an expert-like level of accuracy [1,3,6]. With the advent of electronic medical records, the sheer amount of data necessitates automated means for proper analysis to aid in patient care and research purposes.

Key Points

NLP commonly relies on indexing/tokenization, which is a process of breaking down text strings into data bundles. These bundles then need to be understood, which can be accomplished by mapping to clinical ontology. These clinical ontologies provide a means of disambiguating and organizing the mapped concepts to permit more efficient computation. See Fig. 1 below.

Once the tokenization process occurs, the data can be stored in a variety of methods. Fig. 2a demonstrates how a relational database requires an "unbundling process" to fit into its simplified, tabular storage structure. Although this is an efficient process for both storage and retrieval, data loss occurs as simplifying assumptions are made. As a consequence, during retrieval, queries can only be directed at the level of complexity that is stored. In contrast, Fig. 2b, post-tokenization, the data is not forced into a fixed structure, but rather is stored whole, i.e., in XML databases. XML format databases and ontology-anchoring [5] are important components of modern high-performance NLP systems. The retention of data complexity permits more nuanced and complex queries as the tokenized data can be retrieved in its entirety. This more rigorous model is more computationally intensive. However,



Interoperation of NLP-based Systems with Clinical Databases. Figure 1. NLP system.

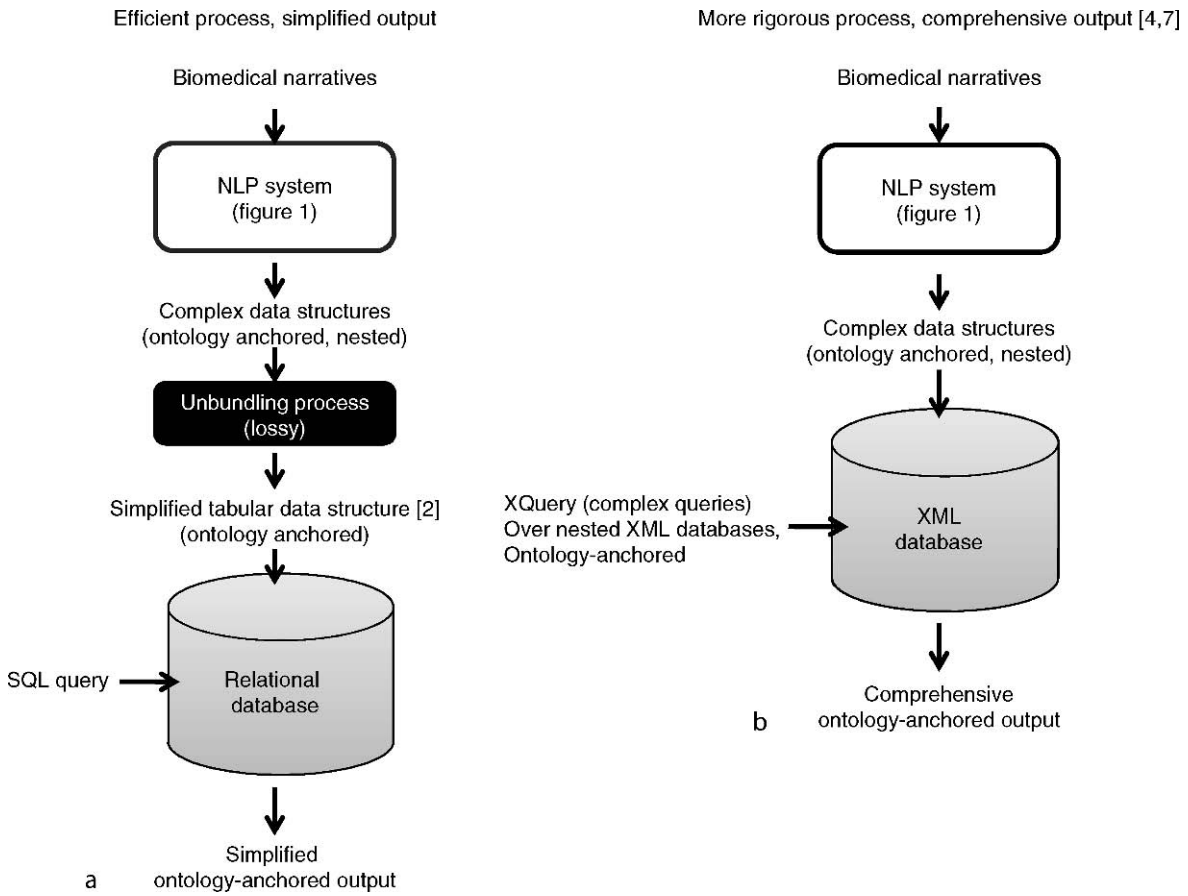
recent advancements in processing power have made these arguments moot. The main upside of this model over the "lossy relational database model" is that it facilitates better storage and utilization of high-throughput generated biomedical data as researchers cannot always anticipate the clinical question posed *a priori*.

Cross-references

- ▶ Clinical Data Acquisition, Storage and Management
- ▶ Clinical Ontologies
- ▶ Electronic Health Record
- ▶ Ontologies and life Science Data Management
- ▶ Storage Management
- ▶ XML Storage

Recommended Reading

1. Chapman WW. Dowling JN. Wagner MM. Classification of emergency department chief complaints into 7 syndromes: a retrospective analysis of 527,228 patients. Ann. Emerg. Med., 46(5):445–455, 2005.



Interoperation of NLP-based Systems with Clinical Databases. Figure 2. Two models of NLP-directed output.

2. Chen ES., Hripcsak G., and Friedman C. Disseminating natural language processed clinical narratives. AMIA Annu Symp Proc., 2006:126–30.
3. Collier N., Nazarenko A., Baud R., and Ruch P. Recent advances in natural language processing for biomedical applications. Int. J. Med. Inform, 75(6):413–417, 2006.
4. Friedman C., Hripcsak G., Shagina L., and Liu H. Representing information in patient reports using natural language processing and the extensible markup language. J. Am. Med. Inform. Assoc., 6(1):76–87, 1999.
5. Friedman C., Shagina L., Lussier Y., and Hripcsak G. Auomated encoding of clinical documents based on natural language processing. J. Am. Med. Inform. Assoc., 11(5):392–402, 2004.
6. Hripcsak G., Friedman C., Alderson PO., DuMouchel W., Johnson SB., and Clayton PD. Unlocking clinical data from narrative reports: a study of natural language processing. Ann. Intern. Med., 122(9):681–688, 1995.
7. Johnson SB., Campbell DA., Krauthammer M., Tulipano PK., Medonca EA., Friedman C., and Hripcsak G. A native XML database design for clinical document research. AMIA Annu Symp Proc., 2003:883.

executed in parallel. For instance, a select operator can be executed in parallel with the subsequent join operator. The advantage of such execution is that the intermediate select result is not materialized, thus saving memory and disk accesses. Pipeline parallelism puts constraints on the way the data at the consuming operator node is stored, i.e., it should fit in main memory. Independent parallelism is easier since it applies when there is no dependency between the operators that are executed in parallel. For instance, the two select operators on two different relations can be executed in parallel. This form of parallelism is very attractive because there is no interference between the nodes.

Cross-references
► Operator-Level Parallelism
► Parallel Data Placement
► Parallel Query Execution Algorithms

Inter-Operator Parallelism

Esther Pacitti
INRIA and LINA, University of Nantes, Nantes, France

Synonyms
Pipelined and independent parallelism

Definition
Inter-operator parallelism enables different operators of the query to be executed in parallel, i.e., by different nodes. Given an operator tree for a query, there are two forms of inter-query parallelism: pipelined and independent parallelism. With pipelined parallelism, an operator that consumes data produced by another operator can proceed in parallel to that operator, as soon as it receives some data. With independent parallelism, two operators with no data dependency can proceed in parallel.

Key Points
Inter-operator parallelism is very attractive when the query is complex, i.e., has many operators on different relations. Thus, the more complex the query, the more opportunities there are for inter-operator parallelism. With pipeline parallelism, operators with a producer-consumer dependency can be

Inter-Query Parallelism

Nikos Hardavellas, Ippokratis Pandis
Carnegie Mellon University, Pittsburgh, PA, USA

Synonyms
User-level parallelism

Definition
Inter-query parallelism is a form of parallelism in the evaluation of database queries, in which several different queries execute concurrently on multiple processors to improve the overall throughput of the system.

Key Points
When multiple non-conflicting requests are submitted to a database management system, then the system can execute them in parallel to improve the overall throughput [1]. This form of parallelism is called inter-query parallelism. Inter-query parallelism is a consequence of the concurrency of user requests. It is orthogonal to intra-query parallelism, in which several processors cooperate for the faster execution of a single query [2]. Both forms of parallelism can co-exist in a database management system. Inter-query parallelism is common in on-line transaction processing (OLTP), where multiple concurrent users submit requests to the system. It is a challenge for the database management

system to achieve high performance and maintain the ACID properties in the presence of multiple concurrently executing requests or transactions.

Cross-references

- ▶ ACID properties
- ▶ Intra-Operator Parallelism
- ▶ Intra-Query Parallelism
- ▶ Operator-Level Parallelism

Recommended Reading

1. DeWitt D.J. and Gray J. Parallel database systems: the future of high-performance database computing. Commun. ACM, 35(6):85–98, 1992.
2. Graefe G. Encapsulation of Parallelism in the Volcano Query Processing System. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1990, pp. 102–111.

Interval-based Temporal Models

- ▶ Period-Stamped Temporal Models

Intra-operator Parallelism

Esther Pacitti
INRIA and LINA, University of Nantes, Nantes, France

Synonyms

Single instruction multiple data (SIMD) parallelism

Definition

Intra-operator parallelism enables an operator which accesses some data to be executed by multiple nodes, each working on a different partition of the data. With intra-operator parallelism, the same operator is applied to multiple partitions, thereby dividing the response time by the number of nodes. Intra-operator parallelism exploits the various forms of data placement and dynamic partitioning using specific algorithms for the different relational operators.

Key Points

Intra-operator parallelism is based on the decomposition of a relational operator into a set of independent operator instances, each processing a different relation partition. This decomposition is done using static or dynamic partitioning of the relations. Static partitioning corresponds to the initial data placement and is typically exploited by the select or scan operators. Dynamic partitioning, i.e., repartitioning a relation a different way, is useful for binary operators that are costly. One main repartitioning solution is hashing on some important attribute, e.g., join attribute. Intra-operator parallelism is easier for unary operators such as scan or select and more difficult for binary operators such as join, in which case, more complex parallel execution algorithms are necessary.

Cross-references

- ▶ Parallel Data Placement
- ▶ Parallel Query Execution Algorithms

Intra-Query Parallelism

Nikos Hardavellas, Ippokratis Pandis
Carnegie Mellon University, Pittsburgh, PA, USA

Synonyms

Query parallelism

Definition

Intra-query parallelism is a form of parallelism in the evaluation of database queries, in which a single query is decomposed into smaller tasks that execute concurrently on multiple processors.

Key Points

Intra-query parallelism is achieved when several processors cooperate in the execution of a single query to improve the query's response time. Intra-query parallelism is orthogonal to inter-query parallelism, in which multiple independent requests execute concurrently on several processors to improve the overall system throughput.

There exist two forms of intra-query parallelism: operator-level parallelism and intra-operator parallelism. Operator-level parallelism is obtained by executing concurrently several operators of the same query. For example, consider a simple query that consists of a scan operator and an aggregation. The scan operator uses a selection condition to filter tuples. The aggregation calculates some statistics over all qualifying tuples. Operator-level parallelism is obtained by executing

concurrently the scan operation as a single task and the aggregation as another, pipelining tuples from the scan to the aggregation as soon as they are produced. To realize this form of parallelism, all participating operators must be pipelineable. Non-pipelineable operators cannot participate because they need their entire input before executing, thereby halting the pipeline.

Intra-operator parallelism is obtained by executing concurrently multiple instances of an operator, with each instance working on a subset of the data. Intra-operator parallelism is based primarily on partitioning the input relation into non-overlapping data segments. In the example above, intra-operator parallelism is achieved by dividing the input relation into non-overlapping data segments and executing the scan and the aggregation of the segments in parallel, followed by a final merge of the results. The interested reader is referred to [1] and [2] as more comprehensive readings.

Cross-references

- ▶ Data Partitioning
- ▶ Inter-Query Parallelism
- ▶ Intra-Operator Parallelism
- ▶ Operator-Level Parallelism
- ▶ Stop-&-Go Operator

Recommended Reading

1. DeWitt D.J. and Gray J. Parallel database systems: the future of high-performance database computing. Commun. ACM, 35(6):85–98, 1992.
2. Graefe G. Encapsulation of parallelism in the volcano query processing system. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1990, pp. 102–111.

Intrinsic Time

- ▶ Valid Time

Intrusion Detection Technology

Tyrone Grandison, Evimaria Terzi
IBM Almaden Research Center, San Jose, CA, USA

Definition

Intrusion Detection (ID) is the process of monitoring events occurring in a system and signalling responsible parties when interesting (suspicious) activity occurs.

Intrusion Detection Systems (IDSs) consist of (i) an agent that collects the information on the stream of monitored events, (ii) an analysis engine that detects signs of intrusion, and (iii) a response module that generates responses based on the outcome from the analysis engine.

Historical Background

The concept of ID has existed for decades in the domains of personal home security, defense and early-warning systems. However, automated IDSs emerged in the public domain in 1980 [2] and sought to identify possible violations of the system's security policy by a user or a set of users.

One of the basic elements of an IDS is the audit log that captures the system activity. The initial IDSs exposed to the academic community stored operating system actions, i.e., addressed the operating system layer. Over time, other IDSs have emerged that store different artifacts, and try to identify intrusive behaviors at different layers of operation. The following layers of operation can be easily identified:

Operating System: The logs in this layer contain information from the kernel and other operating system components and help determine if an attacker is trying to compromise the OS.

Network: At the network layer, communication data is analyzed to determine if an attacker is trying to access one's network.

Application: Application level IDSs examine the operations executed in an application to ascertain if the application is being manipulated to extract behavior that is prohibited. Database-specific IDSs form an important group of application-level IDSs. Examples of such systems include Discovery [14] and RIPPER [10]. Due to the sensitive information stored in database systems, issues related to database-specific IDSs were among the first to be addressed [3,9,15].

The above categorization is historical and mostly depends on the type of log data the IDS uses in order to identify abnormal patterns. Irrespective of the operational layer, the very basic detection techniques used by different IDSs have some common basis, which are described in the next section.

Foundations

A high-level categorization of IDSs is given along with an abstract idea of how they work. For a more complete discussion on IDSs see [1,8,13].

Traditionally, there are two basic approaches to intrusion detection; *anomaly detection* and *misuse detection*. In anomaly detection the goal is to define and characterize legitimate behaviors of the users, and then detect anomalous behaviors by quantifying deviations from the former. However, identifying the distance between anomalous and legitimate behaviors is a rather difficult notion to quantify.

Anomaly detection can be *static* or *dynamic*. A static anomaly detection system is based on the assumption that there is a static portion of the system being monitored. Static portions of the system can be represented as a binary string or a set of binary strings (like files). If the static portion of the system ever deviates from its original form, either an error has occurred or an intruder has altered the static portion of the system.

Dynamic anomaly detectors are harder to build since building them requires a definition of behavior, which is often defined as a sequence (or partially ordered sequence) of distinct events. Differentiating between normal and anomalous activity in dynamic anomaly detection systems is much harder than the problem of distinguishing changes in static elements. Dynamic anomaly detection systems usually create a *base profile* to characterize normal, acceptable behavior. A profile usually consists of a set of observed measures of behavior for a selected set of dimensions. After initializing the base profile the dynamic anomaly detection systems are similar to the static ones; they monitor the behavior by comparing the current behavior with that implied by the base profile. Typically, there is a wide variation of acceptable behaviors and statistical methods are employed to measure deviation from the base profile. The main challenge in dynamic anomaly detection systems is that they must build accurate base profiles and then recognize behaviors that significantly deviate from the profile.

The main advantage of dynamic anomaly detection systems is that they do not require any configuration since they automatically learn the behavior of large number of subjects. Lacking prior knowledge of how an intrusion would manifest itself anomaly detection systems are capable of identifying novel intrusions or variations of known intrusions. However, building base profiles and defining measures of deviations from them is not an easy computational task. For that reason it has been an active area of research, in which several machine learning, time-series analysis and other data-analysis techniques have been employed [3,4–7,11,12].

Misuse detection is concerned with identifying intruders who are attempting to break into a system using some known technique. If a system security administrator was aware of all the known vulnerabilities then a misuse detection system would be able to identify their occurrences and eliminate them. A fairly precisely known kind of intrusion is known as *intrusion scenario*. A misuse detection system compares current system activity to a set of intrusion scenarios in an attempt to identify a scenario in progress.

The differentiating factor between the various misuse detection techniques is the model used for describing bad behaviors that constitute intrusions. *Rules* have been primarily used to model the system-administrator's knowledge about the system. Rule-based systems accumulate large numbers of rules which usually prove difficult to interpret and modify. In order to overcome these problems model-based rule organizations and state-transition representations were proposed. These modeling approaches are more intuitive particularly in misuse detection systems where users need to express and understand scenarios.

The main advantage of a misuse detection systems is that the system knows for a fact how normal behavior should manifest itself. This leads to a simple and efficient processing of the audit data. The obvious disadvantage of such systems is that the specification of the signatures to be detected is a time-consuming task that requires lots of domain knowledge. At the same time, misuse detection systems lack the ability to identify novel intrusion profiles.

Key Applications

A generic classification of the types of attacks that ID systems have traditionally tried to cope with are described below. The classification is mainly inspired by the one provided in [1].

- *External break ins*: When an unauthorized user tries to gain access to a computer system.
- *Masquerander (internal) attacks*: When an authorized user makes an attempt to assume the identity of another user. These attacks are called also internal because they are caused by already authorized users.
- *Penetration attack*: In this attack, a user attempts to directly violate the system's security policy.
- *Leakage*: Moving potentially sensitive data from the system.

- *Denial of Service*: Denying other users the use of system resources, by making these resources unavailable to other users.
- *Malicious use*: Miscellaneous attacks such as file deletion, viruses, resource hogging etc.

Future Directions

One of the major concerns associated with IDSs and their utility is their run-time efficiency. More often than not, IDSs consume too many system resources in order to be effective. Developing resource-aware IDSs systems raises some interesting challenges. One possible way of addressing this concern is via building *Mega Intrusion Detection Systems*. These would be systems that simultaneously monitor all operational layers. That is, the system administrator will not have to run a different ID software for operating system and application specific attacks, but just a single system that will simultaneously be able to detect intrusions in all the desired operational layers. Such systems are expected to be less resource demanding, however their development will certainly create several new design challenges.

This entry has mainly focused on IDSs and described them as mechanisms that guarantee other systems' security. However, IDSs are themselves systems and as such they have their own security risks. Therefore, they also require some protection to prevent an intruder from manipulating the intrusion detection system itself.

Recommended Reading

1. Axelsson S. Research in intrusion detection systems: a survey. In Technical Report 98-17 (revised in 1999). Chalmers University of Technology, 1999.
2. Bace R.G. Intrusion Detection. Macmillan Technical, New York, 2000.
3. Bertino E., Kamra A., Terzi E., and Vakali A. Intrusion detection in rbac-administered databases. In Proc. Asia-Pacific Comp. Syst. Arch. Conf., 2005, pp. 170–182.
4. Bertino E., Leggieri T., and Terzi E. Securing DBMS: characterizing and detecting query floods. In Proc. 7th Int. Conf. on Information Security, 2004, pp. 195–206.
5. Huang Y., Fan W., Lee W., and Yu P. Cross-feature analysis for detecting ad-hoc routing anomalies. In Proc. 23rd Int. Conf. on Distributed Computing Systems, 2003, pp. 478.
6. Kruegel C., Mutz D., Robertson W., and Valeur F. Bayesian event classification for intrusion detection. In Proc. Asia-Pacific Comp. Syst. Arch. Conf., 2003.
7. Lane T. and Brodley C.E. Temporal sequence learning and data reduction for anomaly detection. ACM Trans. Inf. Syst. Secur., 2(3):295–331, 1999.
8. Lee W. and Fan W. Mining system audit data: opportunities and challenges. ACM SIGMOD Rec., 30(4):35–44, 2001.
9. Lee V.C.S., Stankovic J.A., and Son S.H. Intrusion detection in real-time database systems via time signatures. In Proc. IEEE Real Time Technology and Applications Symposium, 2000, pp. 124–133.
10. Lee W., Stolfo S.J., and Mok K.W. A data mining framework for building intrusion detection models. In Proc. IEEE Symp. on Security and Privacy, 1999, pp. 120–132.
11. Lee W. and Xiang D. Information-theoretic measures for anomaly detection. In IEEE Symp. on Security and Privacy, 2001, pp. 130–143.
12. Ramadas M., Ostermann S., and Tjaden B.C. Detecting anomalous network traffic with self-organizing maps. In Proc. 6th Int. Symp. Recent Advances in Intrusion Detection, 2003, pp. 36–54.
13. Stolfo S.J., Lee W., Chan P.K., Fan W., and Eskin E. Data mining-based intrusion detectors: an overview of the columbia ids project. ACM SIGMOD Rec., 30(4):5–14, 2001.
14. Tener W.T. Discovery: an expert system in the commercial data security environment. In Proc. 4th IFIP TCII Int. Conf. on Security, 1986, pp. 261–268.
15. Wenhui S. and Tan D. A novel intrusion detection system model for securing web-based database systems. In Proc. 25th Annual Int. Computer Software Applications Conf., 2001, pp. 249–.

Inverse Document Frequency

Iadh Ounis
University of Glasgow, Glasgow, UK

Synonyms

IDF

Definition

The inverse document frequency (*IDF*) is a statistical weight used for measuring the importance of a term in a text document collection. The document frequency *DF* of a term is defined by the number of documents in which a term appears.

Key Points

Karen Sparck-Jones first proposed that terms with low document frequency are more valuable than terms with high document frequency during retrieval [2]. In other words, the underlying idea of *IDF* is that the more frequently the term appears in the collection, the less informative the term is.

In its simplest form, the *IDF* weight of a term is assigned as follows [3]:

$$IDF = \log_2 \frac{N}{DF} \quad (1)$$

where N is the number of documents in the collection, and DF is the document frequency of the term, i.e., the number of documents in which the term appears.

There have been different variations of the *IDF* weight in the literature. For example, Robertson and Walker proposed the following formula [1]:

$$IDF = \log \frac{N - DF + 0.5}{DF + 0.5} \quad (2)$$

where *0.5* is added to avoid having infinite values brought by zero *DF* values.

Cross-references

► Probabilistic Ranking Principle

Recommended Reading

1. Robertson S.E. and Walker S. On relevance weights with little relevance information. In Proc. 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp 16–24, 1997.
2. Sparck-Jones K. A statistical interpretation of term specificity and its application in retrieval. J. Doc., 28(1):11–20, 1972.
3. Sparck-Jones K. Index term weighting. Inform. Storage Retr., 9(11):619–633, 1973.

Inverse Element Frequency

► Term Statistics for Structured Text Retrieval

Inverted Files

Edleno Silva de Moura[1], Marco Antonio Cristo[2]
[1]Federal University of Amazonas, Manaus, Brazil
[2]FUCAPI, Manaus, Brazil

Synonyms

Inverted index; Full text inverted index; Postings file

Definition

An Inverted file is an index data structure that maps content to its location within a database file, in a document or in a set of documents. It is normally composed of: (i) a vocabulary that contains all the distinct words found in a text and (ii), for each word t of the vocabulary, a list that contains statistics about the occurrences of t in the text. Such list is known as the inverted list of t. The inverted file is the most popular data structure used in document retrieval systems to support full text search.

Historical Background

Efforts for indexing electronic texts are found in literature since the beginning of the computational systems. For example, descriptions of Electronic Information Search Systems that are able to index and search text can be found in the early 1950s [4].

In a seminal work, Gerard Salton wrote a book in 1968, containing the basis for the modern information retrieval systems [6], including a description of a model largely adopted up to now for indexing texts, known as Vector Space Model. The inverted file was adopted as the index structure for implementing the Vector Space Model. It has been widely applied and studied since then.

Foundations

Inverted files allow fast search for statistics related to the distinct words found in a text. They are projected for using words as the search unit, which restricts their use in applications where words are not clearly defined or in applications where the system does not use words as the search unit.

The statistics stored in an inverted file may vary according to the target application. Two alternatives usually found in literature are to record the position of all word occurrences in given text and to record general statistics about the word occurrences across text units. Indexing all the word occurrences is useful in applications where positional information should be taken into account, such as when it is necessary to allow search for phrases or proximity queries. Inverted files that store word statistics are usually deployed in systems that adopt information retrieval models, such as the Vector Space Model. In this particular case the text is divided into units of information, usually documents. For instance, in web search engines, these units are the pages crawled from the web, while the whole set of pages compose the indexed text.

Querying a word in an inverted index consists in first locating the word in the vocabulary and getting the position of its inverted list. This operation can be

performed in O(1) by using a hash algorithm. The inverted list of the word is then accessed in order to provide the search results. The vocabulary usually requires a sub-linear space when compared to the size of the inverted list, which makes it usually far smaller than these lists.

Word occurrences are stored in inverted files for applications where positional information should be taken into account, such as when it is desirable to provide support to phrase search or proximity queries. To search for a phrase or proximity pattern (where the words must appear consecutively or close to each other, respectively), each word is searched separately. Then, the resulting lists of occurrences are intersected considering the consecutiveness or closeness of the word positions in the text. The cost to perform such type of queries can be reduced by adopting an auxiliary data structure, known as next word index, which includes information about the next word in the positional inverted list entries. This alternative can significantly reduce the query processing times for phrase queries. Another choice could be to index pairs of consecutive words, but then the vocabulary would be much larger, which would make this option unfeasible.

Building an Inverted File

The texts indexed nowadays by search systems are usually too large for allowing the creation of inverted files completely in main memory. Disk-based algorithms for generating compressed inverted files have been extensively studied in the literature. An example is the *multiway merging* algorithm described in [7].

An implementation of this algorithm can be performed in three phases a, b, and c. In phase a, all documents are sequentially read from disk and parsed into index terms. This allows creating a perfect hashed vocabulary and also allows the application of filtering algorithms to remove undesired words from the index. For instance, search systems usually remove frequent words included in a pre-computed list, known as *stop words*. Heuristics for removing typos can also be adopted in this phase by analyzing the frequency of occurrences of each word in the text. In phase b, all documents are again sequentially read from disk and again parsed into index terms. Triplets composed of an index term number k_i, a document number d_j, and a frequency $f_{i,j}$ are then formed and inserted into a buffer B in main memory. Whenever this buffer fills (i.e., whenever a run is completed), the partial inverted lists are sorted in decreasing order of the frequencies $f_{i,j}$ (or another desired order according to the application), and stored in a temporary file F. In phase c, a disk-based multiway merge is done to combine the partial inverted lists into final lists. The details are shown in Fig. 1. Note that this example considers that the index stores the frequency of each word in each document. However, the algorithm can easily be adapted to create an index to store word occurrence positions or other type of statistics about the text, according to the target application.

The sequential algorithm shown above uses two passes for reading and parsing of the documents in the collection. This allows building a perfect hashed vocabulary which provides for direct access to any inverted

```
    (a1) read all documents sequentially, parse
         into index terms and create perfect hashed
         vocabulary
    (a2) create memory buffer B
a   (a3) R = 0 /* initialize number of runs */

    foreach dj do
b   begin

    (b1) read dj, parse into index terms
    (b2) foreach ki ∈ dj do B = B + [ki,dj,fi,j]
    (b3)  if 'buffer B is full' then
         begin
         (b3.1) do sort triplets on ki,fi,j
         (b3.2) F = write-disk (B)
         (b3.3) B = ϕ; R = R + 1
         end
    end
c   multiway-disk-merge (F,R)
```

Inverted Files. Figure 1. A disk-based algorithm for building inverted files.

list with no need to lookup at a vocabulary entry. Thus, once the perfect hash has been built, it is no longer necessary to keep the vocabulary in memory (all significant memory consumption is now represented by the buffer *B* which stores the inverted lists).

In cases where the text is too large, as it happens in search engines that try to index the whole web, distributed algorithms should be adopted for building inverted files. The current best alternative for building distributed inverted files is also the simplest solution. It partitions the text into small sub-collections, each of them fitting in a single machine. A local index is built for each sub-collection; when queries arrive, they are submitted to every sub-collection and evaluated against every local index. A final step merges the answers produced by each individual machine yielding a single ranking of results to the final users.

Compression

A technique to reduce the space requirements of inverted files is to compress the index. The key idea to reduce the size of inverted files is that the inverted list entries related to each word can be sorted in increasing order, and therefore the gaps between consecutive positions can be stored instead of the absolute values. Then, compression techniques for small integers can be used. As the gaps are smaller for longer lists, longer lists can be compressed better. Previous work has shown that inverted files can be reduced up to 10% of their original size without degrading the performance, and even the performance may improve because of reduced I/O [7].

Another alternative for reducing the space requirements of an inverted file and the query processing regarding the access to the index is to minimize the number of indexed entries by applying static pruning methods. Pruning methods try to avoid processing index entries without cause loss of quality in the final results produced by the search system. They can be classified as *dynamic* and *static*. Dynamic methods maintain the index completely stored on disk and use heuristics to avoid reading unnecessary information at query processing time. In this case, the amount of pruning performed varies according to the user queries, which represents an advantage, since the methods can be better adapted to each specific query. In contrast, static methods try to predict, at index construction time, the entries which will not be useful at query processing time. These entries are then removed from the index. For this reason, static methods can be seen as lossy compression methods. Static methods offer the advantage of both reducing the disk storage costs and time to process each query. A system that uses both static and dynamic methods can also be implemented to take advantage of the two types of pruning options [5].

Updating Operations

In applications where the indexed text changes over the time, with portions being removed, added or changed in the text, it is necessary to reflect such changes in the inverted file. The simplest approach is to rebuild the whole index, which may be acceptable if the index can be updated offline and the indexing time is small. However, if such conditions do not apply, more sophisticated strategies should be adopted. Several index maintenance strategies can be found in literature [8]. They can be divided into three categories, with the index rebuilding being the first obvious choice. The second category is the intermittent merge, where small indexes to register updates are stored in main memory, making the update inexpensive. In this case, the temporary main memory index and the disk index should be merged at query processing time. A real update should be periodically performed to avoid a memory overflow in the temporary index. The third category is the incremental update. It updates the main index term by term using a process similar to the mechanisms used for maintaining variable-length records in conventional database management systems.

Query Processing

The query processing over inverted files can be performed in two distinct forms, being based on a term order or on a document order basis [2]. In the term order basis, each inverted list is processed one at a time. The partial list of results obtained after processing each list is stored in memory, which means this method may require additional memory. The second form of processing queries is the document order processing, where whenever a document information is found in one of the lists, all information about this document is automatically read from the remaining inverted lists of terms present in the query. The document ordering method requires the inverted lists to be stored sorted by document number, or by occurrence

when the index store all term occurrences. Previous work indicate the term ordering method results in faster query evaluation. However, for small queries, which are common on many search applications, this difference becomes smaller. A combination of document order and term order may also be implemented, by processing the inverted lists in blocks.

A final comment about query processing is that it can be sped up by using cache strategies. At least three distinct cache layers have been proposed in literature. First, the system can adopt a cache of inverted lists to keep the most frequent portions of the lists in memory. Second, it can also be used a cache of results, which takes the final results provided to the users in a cache. Finally, a projection cache containing frequent intersections of lists can also be adopted. Previous work in literature conclude that these cache techniques can significantly increase the maximum capacity of systems for query processing [3].

Key Applications

Inverted files are by far the most applied indexing structures in text search systems. Such indexes are used, for instance, in the popular large scale web search engines.

Cross-references

- ▶ Compressed Inverted Files
- ▶ Information Retrieval Models
- ▶ Lossless Data Compression
- ▶ Text Retrieval
- ▶ Web Search and Crawling

Recommended Reading

1. Baeza-Yates R. and Ribeiro-Neto B. Modern Information Retrieval. Addison Wesley, Reading, MA, 1999.
2. Kaszkiel M. and Zobel J. Term-ordered query evaluation versus document-ordered query evaluation for large document databases. In Proc. 21st Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 1998, pp. 343–344.
3. 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.
4. Luhn H.P. A statistical approach to mechanized encoding and searching of literary information. IBM J. Res. and Dev., 309–317, October 1957.
5. de Moura E.S., dos Santos C.F., Fernandes D.R., Silva A.S., Calado P., and Nascimento M.A. Improving web search efficiency via a locality based static pruning method. In Proc. 14th Int. World Wide Web Conference, 2005, pp. 235–244.
6. Salton G. Automatic Information Organization and Retrieval. McGraw-Hill, New York, NY, 1968.
7. Witten I., Moffat A., and Bell T. Managing Gigabytes, 2nd edn. Morgan Kaufmann, Los Altos, CA, 1999.
8. Zobel J. and Moffat A. Inverted Files for Text Search Engines. ACM Comput. Surv., 38(2):1–56, July 2006.

Inverted Index

- ▶ Inverted Files
- ▶ Text Index Compression

Inverted Indexes

- ▶ Index Creation and File Structures

IP Storage

Kazuo Goda
The University of Tokyo, Tokyo, Japan

Definition

IP Storage is a storage device which has the capability of communicating over an IP network. The term IP Storage is usually identified with a storage device which provides block-level I/O services rather than file access services. Derived from the original meaning, the term IP Storage is sometimes used to refer to an IP SAN.

Key Points

The benefit of utilizing IP technology is its cost efficiency. Having been used for several decades, IP technology is highly mature. IP family protocols have been standardized and cheap hardware products have wide interoperability. The market has a number of excellent administration tools and educated administration personnel for IP technology. Designing and operating SANs on top of IP technology is much cheaper and easier than with Fiber Channel technology. Thus IP technology is sometimes considered to be replacement of Fiber Channel technology in entry-level SANs.

Internet SCSI (iSCSI), Internet Fiber Channel Protocol (iFCP) and Fiber Channel over Internet Protocol

(FCIP) are three major network protocols used for IP Storage. iSCSI transmits the SCSI protocol over IP networks by encapsulating SCSI data in TCP/IP packets. That is, the idea of iSCSI is to replace classical SCSI bus cables with IP networks. iSCSI is used mainly for transferring data between servers and storage devices and among storage devices. A storage device which communicates over the iSCSI protocol is often called an iSCSI target, whereas a server which accesses such an iSCSI device is called an iSCSI initiator. In contrast, iFCP and FCIP can encapsulate Fiber Channel frames in TCP/IP packets. These protocols are usually implemented in network devices such as Fiber Channel switches and routers, thus enabling bridging two or more Fiber Channel SANs by the use of IP networks. Although iSCSI and iFCP/FCIP are both IP storage techniques, they are used in real systems in different ways. That is, iSCSI is mainly used for constructing a local SAN at low cost, whereas iFCP and FCIP are utilized for integrating remote Fiber Channel SANs (often called SAN islands.)

Cross-references

► Storage Network Architectures

Recommended Reading

1. Clark T. IP SANS: A Guide to iSCSI, iFCP, and FCIP Protocols for Storage Area Networks. Addison-Wesley Professional, Reading, MA, 2001.
2. Troppens U., Erkens R., and Müller W. Storage Networks Explained. Wiley, New York, 2004.

ISAM File

► Index Sequential Access Method (ISAM)

iSCSI

► Storage Protocols

ISO 19136

► Geography Markup Language

Isolation

► ACID Properties
► Two-Phase Locking

Iteration

► Loop

Iterator

Evaggelia Pitoura
University of Ioannina, Ioannina, Greece

Synonyms

Iterator; Cursor; Synchronous pipelines

Definition

In general terms, a physical operator is an implementation of a relational operator. For a relational operator, there are many alternative physical operators that implement it, for instance, sort-merge and hash-join provide alternative algorithms for implementing join. The query execution engine provides generic implementations of all physical operators. Typically, each physical operator supports a uniform *iterator interface* that hides any internal implementation details and allows operators to be combined together. The iterator interface includes the functions: (i) *open()* that prepares an operator to produce data, (ii) *next()* that produces an output tuple, and (iii) *close()* that performs the final bookkeeping.

Key Points

During query processing, an input query is transformed to a plan to be executed by the query execution engine. An execution plan can be thought of as a dataflow graph where the nodes correspond to the physical operators and the edges represent the data flow among the physical operators. Generally speaking, a physical operator is an implementation of a relational operator, for instance, sort-merge and hash-join are physical operators providing alternative algorithms for the

implementation of join. The query execution engine provides generic implementations of all physical operators. A plan is executed by calling its physical operators in some (possibly interleaved) order. In most modern database systems, each physical operator supports a uniform *iterator interface* that allows an operator in the plan to get results from its input operators one tuple at a time, hiding the internal implementation details of each operator.

The iterator interface for an operator includes the functions open(), next(), and close(). The *open()* function initializes the state of the operator by allocating the appropriate input and output buffers and passing any related arguments. The code for the *next()* function calls the next() function recursively on each input operator until an output tuple is generated. The state of the operator keeps track of how much input has been consumed. Finally, when all output tuples have been produced, through repeated calls of the next() function, the *close()* function deallocates the state information and performs any other final bookkeeping.

By providing a common interface to all physical operators, any two physical operators can be plugged together. Furthermore, the iterator interface supports a pipeline model for the execution of the plan.

The iterator interface is also employed to encapsulate access methods such as the various kinds of indexes that the database system supports. In this case, open() can be used to pass the corresponding selection condition. Finally, parallelism and network communications can be encapsulated within special *exchange iterators.*

Cross-references

- ► Access Path
- ► Evaluation of Relational Operators
- ► Pipelining
- ► Query Plan

Recommended Reading

1. Graefe G. Query evaluation techniques for large databases. ACM Comput. Surv., 25(2): 73–170, 1993.
2. Hellerstein J.M., Stonebraker M., and Hamilton J. Architecture of a database system. Found. Trends Databases, 1(2):141–259, 2007.
3. Ramakrishnan R. and Gehrke J. Database Management Systems. McGraw-Hill, New York, 2003.