Module 6

Hierarchical Clustering method: BIRCH. Density-Based Clustering –DBSCAN and OPTICS. Advanced Data Mining Techniques: Introduction, Web Mining- Web Content Mining, Web Structure Mining, Web Usage Mining. Text Mining. Graph mining:- Apriori based approach for mining frequent subgraphs. Social Network Analysis:- characteristics of social networks. Link mining:- Tasks and challenges

Hierarchical Clustering method: BIRCH

Hierarchical methods

A hierarchical clustering method works by grouping data objects into a tree of clusters. Hierarchical clustering methods can be further classified into agglomerative and divisive hierarchical clustering, depending on whether the hierarchical decomposition is formed in a bottom-up or top-down fashion. The quality of a pure hierarchical clustering method suffers from its inability to perform adjustment once a merge or split decision has been executed.

Agglomerative and divisive hierarchical clustering

In general, there are two types of hierarchical clustering methods:

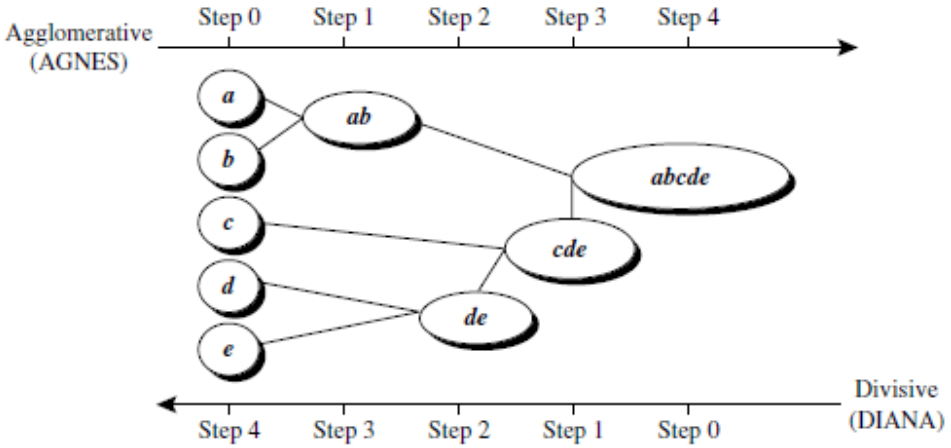
1. Agglomerative hierarchical clustering: This bottom-up strategy starts by placing each object in its own cluster and then merges these atomic clusters into larger and larger clusters, until all of the objects are in a single cluster or until certain termination conditions are satisfied. Most hierarchical clustering methods belong to this category. They differ only in their definition of inter-cluster similarity.

2. Divisive hierarchical clustering: This top-down strategy does the reverse of agglomerative hierarchical clustering by starting with all objects in one cluster. It subdivides the cluster into smaller and smaller pieces, until each object forms a cluster on its own or until it satisfies certain termination conditions, such as a desired number of clusters is obtained or the distance between the two closest clusters is above a certain threshold distance.

Figure below shows the application of AGNES (AGglomerative NESting), an agglomerative hierarchical clustering method, and DIANA (DIvisia ANAlysis), a divisive hierarchical clustering method, to a data set of five objects, {a, b, c, d, e}. Initially, AGNES places each object into a cluster of its own. The clusters are then merged step-by-step according to some criterion. For example, clusters C1 and C2 may be merged if an object in C1 and an object in C2 form the minimum Euclidean distance between any two objects from different clusters. This is a single-link approach in that each cluster is represented by all of the objects in the cluster, and the similarity between two clusters is measured by the similarity of the closest pair of data points belonging to different clusters.

The cluster merging process repeats until all of the objects are eventually merged to form one cluster. In DIANA, all of the objects are used to form one initial cluster. The cluster is split according to some principle, such as the maximumEuclidean distance

between the closest neighboring objects in the cluster. The cluster splitting process repeats until, eventually, each new cluster contains only a single object.



Agglomerative and divisive hierarchical clustering on data objects $\{a, b, c, d, e\}$.

In either agglomerative or divisive hierarchical clustering, one can specify the desired number of clusters as a termination condition.

Four widely used measures for distance between clusters are as follows, where $|p - p'|$ is the distance between two objects or points, p and p'; m_i is the mean for cluster, C_i; and n_i is the number of objects in C_i. They are also known as *linkage measures*.

$$\text{Minimum distance:} \quad dist_{min}(C_i, C_j) = \min_{p \in C_i, p' \in C_j} \{|p - p'|\} \tag{10.3}$$

$$\text{Maximum distance:} \quad dist_{max}(C_i, C_j) = \max_{p \in C_i, p' \in C_j} \{|p - p'|\} \tag{10.4}$$

$$\text{Mean distance:} \quad dist_{mean}(C_i, C_j) = |m_i - m_j| \tag{10.5}$$

$$\text{Average distance:} \quad dist_{avg}(C_i, C_j) = \frac{1}{n_i n_j} \sum_{p \in C_i, p' \in C_j} |p - p'| \tag{10.6}$$

“What are some of the difficulties with hierarchical clustering?" The hierarchical clustering method, though simple, often encounters difficulties regarding the selection of merge or split points. Such a decision is critical because once a group of objects is merged or split, the process at the next step will operate on the newly generated clusters. It will neither undo what was done previously, nor perform object swapping between clusters. Thus merge or split decisions, if not well chosen at some

step, may lead to low quality clusters. Moreover, the method does not scale well since the decision of merge or split needs to examine and evaluate a good number of objects or clusters.

One promising direction for improving the clustering quality of hierarchical methods is to integrate hierarchical clustering with other clustering techniques for multiple phase clustering. A few such methods are introduced in the following subsections. The first, called BIRCH, begins by partitioning objects hierarchically using tree structures, and then applies other clustering algorithms to refine the clusters.

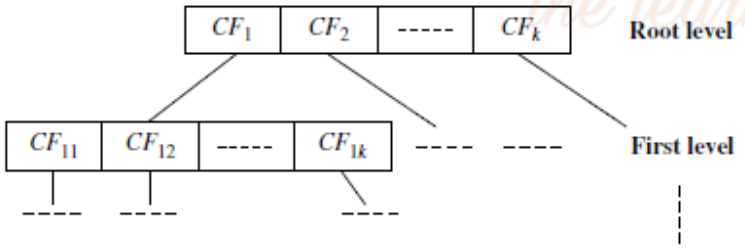
BIRCH: Balanced Iterative Reducing and Clustering using Hierarchies

BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) is an integrated hierarchical clustering method. It introduces two concepts, that of clustering feature and clustering feature tree (CF tree), which are used to summarize cluster representations. These structures help the clustering method achieve good speed and scalability in large databases. BIRCH is also effective for incremental and dynamic clustering of incoming objects.

Let's have a closer look at the above-mentioned structures. A clustering feature (CF) is a triplet summarizing information about subclusters of objects. Given N d-dimensional points or objects { Xi }in a subcluster, then the CF of the subcluster is defined as

CF = (N, $\overrightarrow{LS}$,SS) ,

where N is the number of points in the subcluster, $\overrightarrow{LS}$ is the linear sum on N points, i.e., $\sum_{i=1}^{n} x_i$ and SS is the square sum of data points, i.e., $\sum_{i=1}^{n} x_i^2$



CF-tree structure.

A clustering feature is essentially a summary of the statistics for the given subcluster: the zero-th, first, and second moments of the subcluster from a statistical point of view. It registers crucial measurements for computing clusters and utilizes storage efficiently since it summarizes the information about the subclusters of objects instead of storing all objects.

A CF tree is a height-balanced tree which stores the clustering features for a hierarchical clustering. An example is shown in Figure above. By definition, a non-leaf node in a tree has descendents or \children". The non-leaf nodes store sums of the CFs of their children, and thus, summarize clustering information about their

children. A CF tree has two parameters: branching factor, B, and threshold, T. The branching factor specifies the maximum number of children per non-leaf node. The threshold parameter speci_es the maximum diameter of subclusters stored at the leaf nodes of the tree. These two parameters inuence the size of the resulting tree.

“How does the BIRCH algorithm work?" It consists of two phases:

Phase 1: BIRCH scans the database to build an initial in-memoryCF tree, which can be viewed as a multilevel compression of the data that tries to preserve the inherent clustering structure of the data.

Phase 2: BIRCH applies a (selected) clustering algorithm to cluster the leaf nodes of the CF-tree.

For Phase 1, the CF tree is built dynamically as objects are inserted. Thus, the method is incremental. An object is inserted to the closest leaf entry (subcluster). If the diameter of the subcluster stored in the leaf node after insertion is larger than the threshold value, then the leaf node and possibly other nodes are split. After the insertion of the new object, information about it is passed towards the root of the tree. The size of the CF tree can be changed by modifying the threshold. If the size of the memory that is needed for storing the CF tree is larger than the size of the main memory, then a smaller threshold value can be specified and the CF tree is rebuilt. The rebuild process is performed by building a new tree from the leaf nodes of the old tree. Thus, the process of rebuilding the tree is done without the necessity of rereading all of the objects or points. This is similar to the insertion and node split in the construction of B+-trees. Therefore, for building the tree, data has to be read just once. Some heuristics and methods have been introduced to deal with outliers and improve the quality of CF trees by additional scans of the

data.

After the CF tree is built, any clustering algorithm, such as a typical partitioning algorithm, can be used with the CF tree in Phase 2.

BIRCH tries to produce the best clusters with the available resources. With a limited amount of main memory, an important consideration is to minimize the time required for I/O. BIRCH applies a multiphase clustering technique:

A single scan of data set yields a basic good clustering, and one or more additional scans can (optionally) be used to further improve the quality. The computation complexity of the algorithm is O(n), where n is the number of objects to be clustered.

“How effective is BIRCH?" Experiments have shown the linear scalability of the algorithm with respect to the number of objects, and good quality of clustering of the data. However, since each node in a CF-tree can hold only a limited number of entries due to its size, a CF-tree node does not always correspond to what a user may consider a natural cluster. Moreover, if the clusters are not spherical in shape, BIRCH does not perform well because it uses the notion of radius or diameter to control the boundary of a cluster.

Density-based methods

To discover clusters with arbitrary shape, density-based clustering methods have been developed. These typically regard clusters as dense regions of objects in the data space which are separated by regions of low density (representing noise).

DBSCAN: A density-based clustering method based on connected regions with sufficiently high density

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a density-based clustering algorithm. The algorithm grows regions with sufficiently high density into clusters, and discovers clusters of arbitrary shape in spatial databases with noise. It defines a cluster as a maximal set of density-connected points.

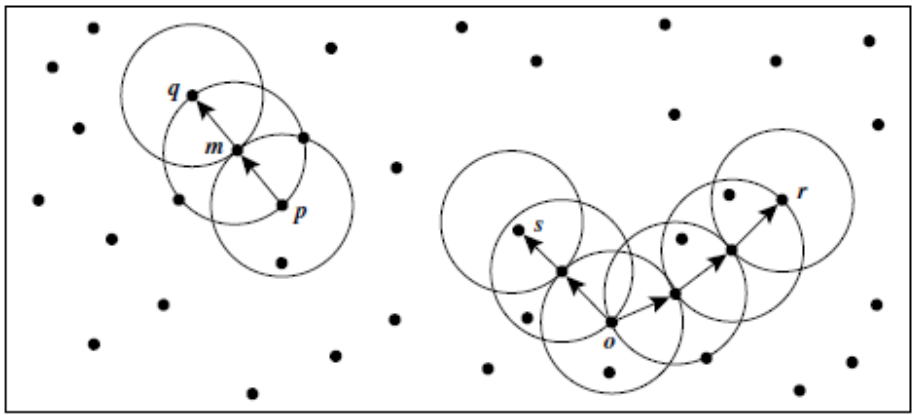
The basic ideas of density-based clustering involve a number of new definitions. We intuitively present these definitions, and then follow up with an example.

- The neighborhood within a radius ε of a given object is called the ε -neighborhood of the object.
- If the ε -neighborhood of an object contains at least a minimum number, MinP ts, of objects, then the object is called a core object.
- Given a set of objects, D, we say that an object p is directly density-reachable from object q if p is within the ε -neighborhood of q, and q is a core object.
- An object p is density-reachable from object q with respect to ε and MinP ts in a set of objects, D, if there is a chain of objects p1, . .. , pn, p1 = q and pn = p such that pi+1 is directly density reachable from pi with respect to ε and MinP ts, for 1 ≤ i ≤ n, pi € D.
- An object p is density-connected to object q with respect to ε and MinP ts in a set of objects, D, if there is an object o € D such that both p and q are density-reachable from o with respect to ε and MinP ts.

Density reachability is the transitive closure of direct density reachability, and this relationship is asymmetric. Only core objects are mutually density reachable. Density connectivity, however, is a symmetric relation.

Example 8.5 Consider Figure 8.9 for a given ε represented by the radius of the circles, and, say, let MinP ts = 3.

Based on the above definitions:



Density-reachability and density-connectivity in density-based clustering. *Source:* Based on Ester, Kriegel, Sander, and Xu [EKSX96].

- Of the labeled points, M, P, O, and R are core objects since each is in an **ε** -neighborhood containing at least three points.
- M is directly density-reachable from P, and Q is directly density-reachable from M.
- Based on the previous observation, Q is (indirectly) density-reachable from P. However, P is not density-reachable from Q. Similarly, R and S are density-reachable from O.
- O, R and S are all density-connected.

A density-based cluster is a set of density-connected objects that is maximal with respect to density-reachability. Every object not contained in any cluster is considered to be noise.

“How does DBSCAN find clusters?" DBSCAN checks the **ε** -neighborhood of each point in the database. If the **ε** -neighborhood of a point p contains more than MinP ts, a new cluster with p as a core object is created. DBSCAN then iteratively collects directly density-reachable objects from these core objects, which may involve the merge of a few density-reachable clusters. The process terminates when no new point can be added to any cluster

Algorithm: DBSCAN: a density-based clustering algorithm.

Input:

- D: a data set containing n objects,
- ϵ: the radius parameter, and
- *MinPts*: the neighborhood density threshold.

Output: A set of density-based clusters.

Method:

```
(1)  mark all objects as unvisited;
(2)  do
(3)      randomly select an unvisited object p;
(4)      mark p as visited;
(5)      if the ε-neighborhood of p has at least MinPts objects
(6)          create a new cluster C, and add p to C;
(7)          let N be the set of objects in the ε-neighborhood of p;
(8)          for each point p' in N
(9)              if p' is unvisited
(10)                 mark p' as visited;
(11)                 if the ε-neighborhood of p' has at least MinPts points,
                     add those points to N;
(12)             if p' is not yet a member of any cluster, add p' to C;
(13)         end for
(14)         output C;
(15)     else mark p as noise;
(16) until no object is unvisited;
```

DBSCAN algorithm.

In this process, for an object **p**0 in N that carries the label "unvisited," DBSCAN marks it as "visited" and checks its **ε** -neighborhood. If the **ε** -neighborhood of **p**0 has at leastMinPts objects, those objects in the **ε** -neighborhood of **p**0 are added to N. DBSCAN continues adding objects to C until C can no longer be expanded, that is, N is empty. At this time, cluster C is completed, and thus is output.

To find the next cluster, DBSCAN randomly selects an unvisited object from the remaining ones. The clustering process continues until all objects are visited. The pseudocode of the DBSCAN algorithm is given in Figure 10.15.

The computational complexity of DBSCAN is O(n log n), where n is the number of database objects. The algorithm is sensitive to the user-defined parameters. DBSCAN is further discussed in the following section, which compares it to an alternative density-based clustering method called OPTICS.

OPTICS: Ordering Points To Identify the Clustering Structure

Although DBSCAN (the density-based clustering algorithm described in Section 8.6.1) can cluster objects given input parameters such as **ε** and MinP ts, it still leaves the user with the responsibility of selecting parameter values that will lead to the discovery of acceptable clusters. Actually, this is a problem associated with many other clustering algorithms. Such parameter settings are usually empirically set and difficult to determine, especially for real-world, high-dimensional data sets. Most algorithms are very sensitive to such parameter values: slightly different settings may lead to very dfferent clusterings of the data. Moreover, high-dimensional real data sets often have very skewed distributions. There does not even exist a global parameter setting for which the result of a clustering algorithm may accurately describe the intrinsic clustering structure.

To help overcome this difficulty, a cluster ordering method called OPTICS (Ordering Points To Identify the Clustering Structure) was proposed. Rather than produce a data set clustering explicitly, OPTICS computes an augmented cluster ordering for automatic and interactive cluster analysis. This ordering represents the density-based clustering structure of the data. It contains information which is equivalent to density-based clustering obtained from a wide range of parameter settings.

By examining DBSCAN, one can easily see that for a constant MinPts -value, density-based clusters with respect to a higher density (i.e., a lower value for **ε**) are completely contained in density-connected sets obtained with respect to a lower density. Recall that the parameter **ε** is a distance - it is the neighborhood radius. Therefore, in order to produce a set or ordering of density-based clusters, we provide a set of distance parameter values. To construct the different clusterings simultaneously, the objects should be processed in a specific order. This order selects an object that is density-reachable with respect to the lowest **ε** value so that clusters with higher density (lower **ε**) will be finished first. Based on this idea, two values need to be stored for each object { core-distance and reachability-distance:

- The core-distance of an object p is the smallest ε' value that makes p a core object. If p is not a core object, the core-distance of p is undefined.
- The reachability-distance of an object p and another object o is the greater value of the core-distance of p and the Euclidean distance between p and q. If p is not a core object, the reachability-distance between p and q is undefined.

“How are these values used?" The OPTICS algorithm creates an ordering of the objects in a database, additionally storing the core-distance and a suitable reachability-distance for each object. Such information is sufficient for the extraction of all density-based clusterings with respect to any distance ε' that is smaller than the distance **ε** used in generating the order.

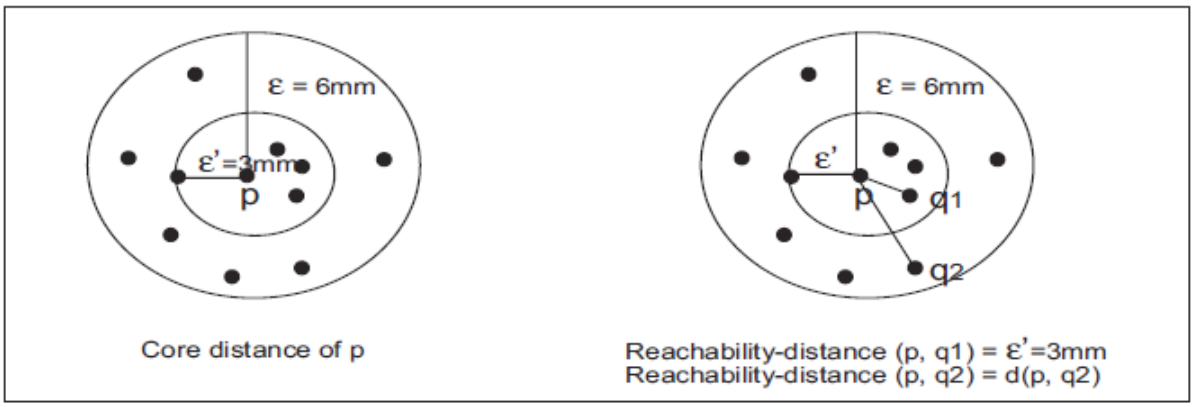


Figure 8.10: OPTICS terminology.

Figure 8.10 illustrates the concepts of core-distance and reachability-distance. Suppose that **ε** = 6mm

and MinP ts = 5. The core-distance of p is the distance, **ε**′, between p and the fourth closest data object.

The cluster ordering of a data set can be represented graphically, which helps in its understanding. For example, Figure above is the reachability plot for a simple 2-dimensional data set, which presents a general overview of how the data are structured and clustered. Methods have also been developed for viewing clustering structures of high- dimensional data.

Because of the structural equivalence of the OPTICS algorithm to DBSCAN, the OPTICS algorithm has the same run-time complexity as that of DBSCAN, i.e., O(n log n). Spatial indexing structures can be used to further enhance its performance.

Advanced Data Mining Techniques: Introduction, Web Mining- Web Content Mining, Web Structure Mining, Web Usage Mining. Text Mining. Graph mining:- Apriori based approach for mining frequent subgraphs. Social Network Analysis:- characteristics of social networks. Link mining:- Tasks and challenges

: Introduction, Web Mining- Web Content Mining, Web Structure Mining, Web Usage Mining.

INTRODUCTION

Determining the size of the World Wide Web is extremely difficult. It 1999 it was estimated to contain over 350 million pages with growth at the rate of about 1 million pages a day [CvdBD99]. Google recently announced that it indexes 3 billion Web documents [Goo01]. The Web can be viewed. as the the largest database available and presents a challenging task for effective design and access, Here we use the term database quite loosely because, of course, there is no real structure or schema to the Web. Thus, data mining applied to the Web has the potential to be quite beneficial. Web mining is mining of data related to the World Wide Web. This

may be the data actually present in Web pages or data related to Web activity. Web data can be classified into the following classes [SCDTOO]:

- Content of actual Web pages.
- Intrapage structure includes the HTML or XML code for the page.
- Interpage structure is the actual linkage structure between Web pages.
- Usage data that describe how Web pages are accessed by visitors.
- User profiles include demographic and registration information obtained about users. This could also include information found in cookies.

Web mining tasks can be divided into several classes. Figure 7.1 shows one taxonomy of Web mining activities [Za199]. Web content mining examines the content of Web pages as well as results of Web searching. The content includes text as well as graphics data. Web content mining is further divided into Web page content mining and search results mining. The first is traditional searching of Web pages via content, whilethe second is a further search of pages found from a previous search. Thus, some mining activities have beep built on top of traditional search engines, using their result as the data to be mined. With Web structure mining, information is obtained from the actual organization of pages on the Web.

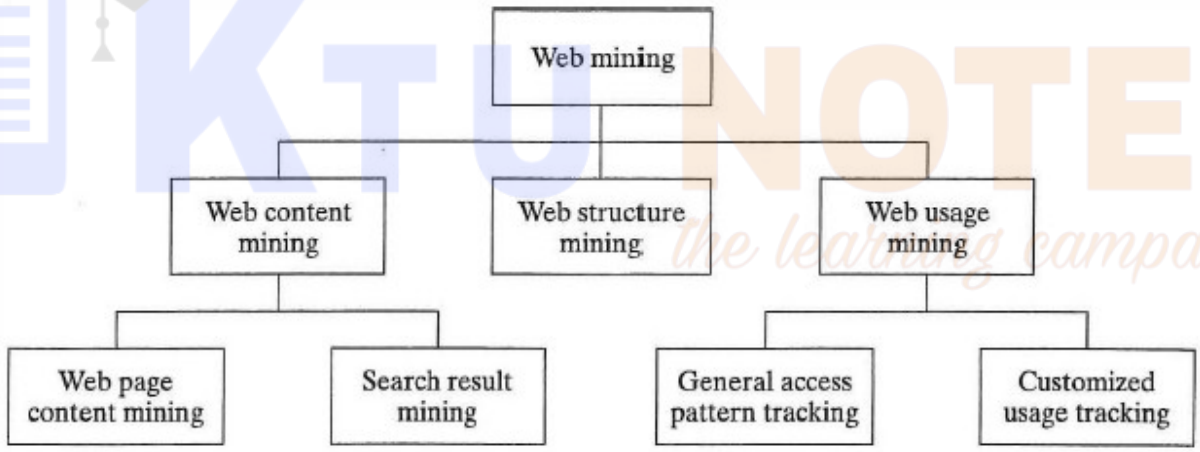


FIGURE 7.1: Web mining taxonomy (modified from [Zaï99]).

Content mining is similar to the work performed by basic IR techniques, but it usually goes farther than simply employing keyword searching. For example, clustering may be applied to Web pages to identify similar pages. The intrapage structure includes links within the page as well as the code (HTML, XML) for the page. Web usage mining looks at logs of Web access. General access pattern tracking is a type of usage mining that looks at a history of Web pages visited. This usage may be general or may be targeted to specific usage or users. Besides identifying what the traffic patterns look like, usage mining also involves the mining of these sequential patterns.

For example, patterns can be clustered based on their similarity. This in turn can be used to cluster users into groups based on similar access behavior.

WEB CONTENT MINING

Web content mining can be thought of as extending the work performed by basic searchengines. There are many different techniques that can be used to search the Internet.Only a few of these techniques are discussed here. Most search engines are keywordbased.Web content mining goes beyond this basic IR technology. It can improve ontraditional search engines through such techniques as concept hierarchies and synonyms,user profiles, and analyzing the links between pages. Traditional search engines must have crawlers to search the Web and gather information, indexing techniques to store the information, and query processing support to provide fast and accurate information to users. Data mining techniques can be used to help search engines provide the efficiency, effectiveness, and scalability needed.

One taxonomy of Web mining divided Web content mining into agent-based and database approaches [CMS97] . Agent-based approaches have software systems (agents)that perform the content mining. In the simplest case, search engines belong to this class,as do intelligent search agents, information filtering, and personalized Web agents. Intelligent search agents go beyond the simple search engines and use other techniques besides keyword searching to accomplish a search. For example, they may use user profiles or knowledge concerning specified domains. Information filtering utilizes IR techniques, knowledge of the link structures, and other approaches to retrieve and categorize documents. Personalized Web agents use information about user preferences to direct their search. The database approaches view the Web data as belonging to a database. There have been approaches that view the Web as a multilevel database, and there have been many query languages that target the Web.

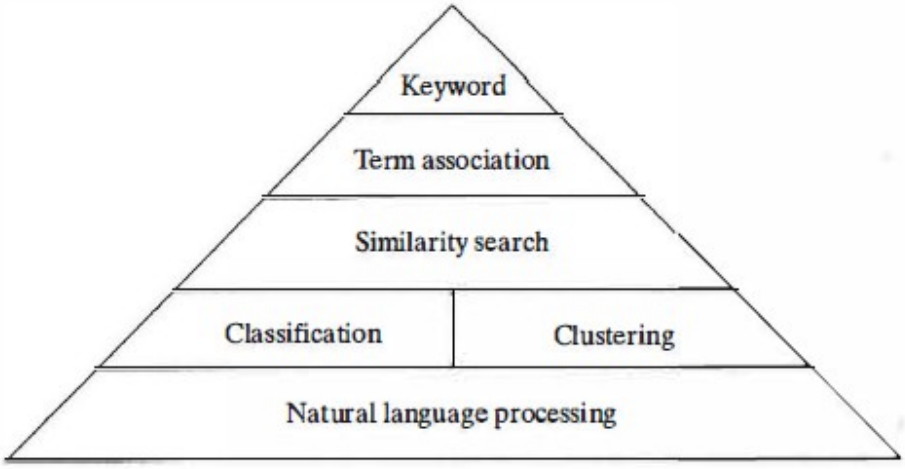


FIGURE 7.2: Text mining hierarchy (modified version of [Zaï99, Figure 2.1]).

Basic content mining is a type of text mining. As seen in Figure 7.2, a modified version of [Za199, Figure 2.1], text mining functions can be viewed in a hierarchy with the simplest functions at the top and the more complex functions at the bottom. Much research is currently under way that investigates the use of natural language processing techniques in text mining to uncover hidden semantics, such as question and answer systems. Moretraditional mining operations involve keyword searching, similarity measures, clustering, and classification.

Many Web content mining activities have centered around techniques to summarize the information found. In the simplest case, inverted file indices are created on keywords. Simple search engines retrieve relevant documents usually using a keywordbased retrieval technique similar to those found in traditional IR systems. While these do not perform data mining activities, their functionality could be extended to include more mining-type activities.

One problem associated with retrieval of data from Web documents is that they are not structured as in traditional databases. There is no schema or division into attributes.

Traditionally, Web pages are defined using hypertext markup language (HTML). Web pages created using HTML are only semistructured, thus making querying more difficult than with well-formed databases containing schemas and attributes with defined domains. HTML ultimately will be replaced by extensible markup language (XML), which will provide structured documents and facilitate easier mining.

Crawlers

A robot (or spider or crawler) is a program that traverses the hypertext structure in the Web. The page (or set of pages) that the crawler starts with are referred to as the seed URLs. By starting at one page, all links from it are recorded and saved in a queue. These new pages are in turn searched and their links are saved. As these robots search the Web, they may collect information about each

page, such as extract keywords and store in indices for users of the associated search engine. A crawler may visit a certain number of pages and then stop, build an index, and replace the existing index. This type of crawler is referred to as a periodic crawler because it is activated periodically.

Crawlers are used to facilitate the creation of indices used by search engines. They allow the indices to be kept relatively up-to-date with little human intervention. Recent research has examined how to use an incremental crawler. Traditional crawlers usually replace the entire index or a section thereof. An incremental crawler selectively searches the Web and only updates the index incrementally as opposed to replacing it. Because of the tremendous size of the Web, it has also been proposed that a focused crawler be used.

A focused crawler visits pages related to topics of interest. This concept is illustrated in Figure 7.3. Figure 7.3(a) illustrates what happens with regular crawling, while Figure 7.3(b) illustrates focused crawling. The shaded boxes represent pages that are visited. With focused crawling, if it is determined that a page is not relevant or its links should not be followed, then the entire set of possible pages underneath it are pruned and not visited. With thousands of focused crawlers, more of the Web can be covered than with traditional crawlers. This facilitates better scalability as the Web grows. The focused crawler architecture consists of three primary components [CvdBD99]:

• A major piece of the architecture is a hypertext classifier that associates a relevance score for each document with respect to the crawl topic. In addition, the classifier determines a resource rating that estimates how beneficial it would be for the crawler to follow the links out of that page.

• A distiller determines which pages contain links to many relevant pages. These are called hub pages. These are thus highly important pages to be visited. These hub pages may not contain relevant information, but they would be quite important to facilitate continuing the search.

• The crawler performs the actual crawling on the Web. The pages it visits are determined via a priority-based structure governed by the priority associated with pages by the classifier and the distiller.

A performance objective for the focused crawler is a high precision rate or harvest rate. To use the focused crawler, the user first identifies some sample

documents that are of interest. While the user

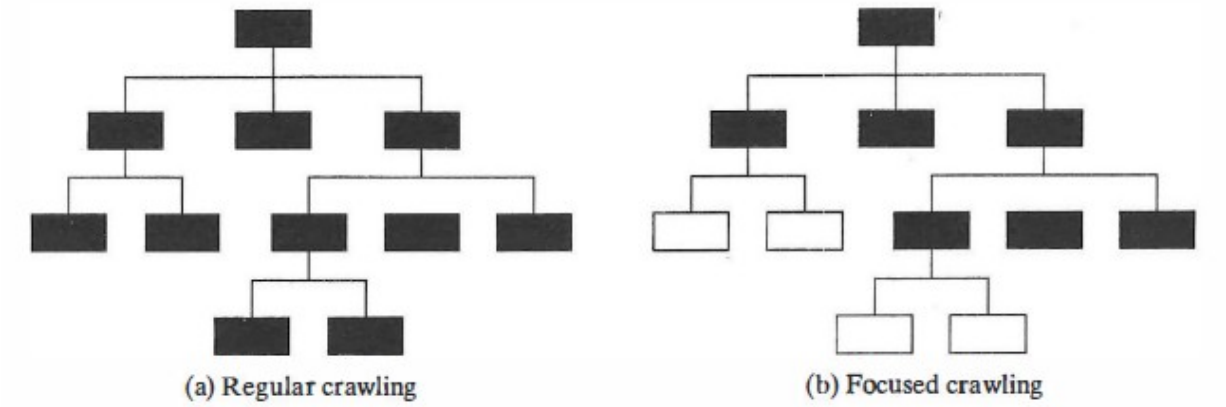


FIGURE 7.3: Focused crawling.

browses on the Web, he identifies the documents that are of interest. These are then classified based on a hierarchical classification tree, and nodes in the tree are marked as good, thus indicating that this node in the tree has associated with it document(s) that are of interest. These documents are then used as the seed documents to begin the focused crawling. During the crawling phase, as relevant documents are found it is determined whether it is worthwhile to follow the links out of these documents. Each document is classified into a leaf node of the taxonomy tree.

One proposed approach, hard focus, follows links if there is an ancestor of this node that has been marked as good. Another technique, soft focus, identifies the probability that a page, d, is relevant as

$$R(d) = \sum_{good(c)} P(c \mid d) \qquad (7.1)$$

Here c is a node in the tree (thus a page) and good(c) is the indication that it has been labeled to be of interest. The priority of visiting a page not yet visited is the maximum of the relevance of pages that have been visited and point to it.

The hierarchical classification approach uses a hierarchical taxonomy and a naive Bayes classifier. A hierarchical classifier allows the classification to include information contained in the document as well as other documents near it (in the linkage structure).

The objective is to classify a document d to the leaf node c in the hierarchy with the highest posterior probability P(c I d) . Based on statistics of a training set, each node c in the taxonomy has a probability. The probability that a document can be generated by the root topic, node q , obviously is 1. Following the argument found in [CDAR98], suppose q , . . . , Ck = c be the path from the root node to the leaf c. We

thus know

$$P(c_i \mid d) = P(c_{i-1} \mid d)\,P(c_i \mid c_{i-1}, d) \tag{7.2}$$

Using Bayes rule, we have

$$P(c_i \mid c_{i-1}, d) = \frac{P(c_i \mid c_{i-1})P(d \mid c_i)}{\sum_{s \text{ is a sibling of } c_i} P(d \mid s)} \tag{7.3}$$

P (d I c_i) can be found using the Bernoulli model, in which a document is seen as a bag of words with no order [CvdBD99] .

More recent work on focused crawling has proposed the use of context graphs. The context focused crawler (CFC) performs crawling in two steps. In the first phase, context graphs and classifiers are constructed using a set of seed documents as a training set. In the second phase, crawling is performed using the classifiers to guide it. In addition, the context graphs are updated as the crawl takes place. This is a major difference from the focused crawler, where the classifier is static after the learning phase. The CFC approach is designed to overcome problems associated with previous crawlers:

• There may be some pages that are not relevant but that have links to relevant pages. The links out of these documents should be followed.

• Relevant pages may actually have links into an existing relevant page, but no links into them from relevant pages. However, crawling can really only follow the links out of a page. It would be nice to identify pages that point to the current page. A type of backward crawling to determine these pages would be beneficial.

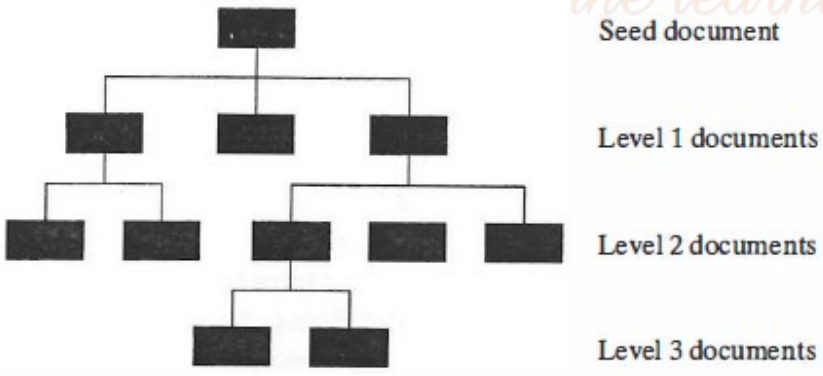


FIGURE 7.4: Context graph.

The CFC approach uses a context graph, which is a rooted graph in which the root represents a seed document and nodes at each level represent pages that have links to a node at the next higher level. Figure 7.4 contains three levels. The number of levels in a context graph is dictated by the user. A node in the graph with a path of length n to the seed document node represents a document that has links indirectly

to the seed document through a path of length n. The number of back links followed is designated as input to the algorithm. Here n is called the depth of the context graph. The context graphs created for all seed documents are merged to create a merged context graph. The context graph is used to gather information about topics that are related to the topic being explored.

Backward crawling finds pages that are not pointed to by relevant documents but are themselves relevant. These types of pages may be new and may not yet have been discovered and linked to from other pages. Although backward links do not really exist in the Web, a backward crawl can be performed relatively easily because most search engines already maintain information about the back links. This type of information is similar to that often used by commercial citation servers, which find documents that cite a given document. The value of the Science Citation Index in performing traditional literature searches is well known. The use of backlinks on the Web can provide similar benefits.

CFC performs classification using a term frequency-inverse document frequency (TF-IDF) technique. The vocabulary used is formed from the documents in the seed set and is shown in the merged context graph. Each document is represented by a TF-IDF vector representation and is assigned to a particular level in the merged context graph.

Harvest System

The Harvest system is based on the use of caching, indexing, and crawling. Harvest is actually a set of tools that facilitate gathering of information from diverse sources. The Harvest design is centered around the use of gatherers and brokers. A gatherer obtains information for indexing from an Internet service provider, while a broker provides the index and query interface. The relationship between brokers and gatherers can vary. Brokers may interface directly with gatherers or may go through other brokers to get to the gatherers. Indices in Harvest are topic-specific, as are brokers. This is used to avoid the scalability problems found without this approach.

Harvest gatherers use the Essence system to assist in collecting data. Although not designed explicitly for use on the Web, Essence has been shown to be a valid technique for retrieving Web documents [HS93]. Essence classifies documents by creating a semantic index. Semantic indexing generates different types of information for different types of files and then creates indices on this information. This process may first classify files based on type and then summarize the files typically based on keywords. Essence uses the file extensions to help classify file types.

Virtual Web View

One proposed approach to handling the large amounts of somewhat unstructured data on the Web is to create a multiple layered database (MLDB) on top of the data in the Web (or a portion thereof). This database is massive and distributed. Each layer of this database is more generalized than the layer beneath it. Unlike the lowest level (the Web), the upper levels are structured and can be accessed (and mined) by an SQL-like query language. The MLDB provides an abstracted and condensed view of a portion of the Web. A view of the MLDB, which is called a Virtual Web View (VWV), can be constructed.

The indexing approach used by MLDB does not require the use of spiders. The technique used is to have the Web servers (masters, administrators) themselves send their indices (or changes to indices) to the site(s) where indexing is being performed. This process is triggered when changes to the sites are made. Each layer of the index is smaller than that beneath it and to which it points. To assist in the creation of the first layer of the MLDB, both extraction and translation tools are proposed. Translation tools are used to convert Web documents to XML, while extraction tools extract the desired information from the Web pages and insert it into the first layer of the MLDB . Web documents that use XML and follow a standard format would not need any tools to create the layers. It is proposed that any translation functions be performed directly by the local administrators .

The layer- 1 data can be viewed as a massive distributed database. The higher levels of the database become less distributed and more summarized as they move up the hierarchy. Generalization tools are proposed, and concept hierarchies are used to assist in the generalization process for constructing the higher levels of the MLDB . These hierarchies can be created using the WordNet Semantic Network. WordNet is a database of the English language. Nouns, adjectives, verbs, and adverbs are listed, divided into groups of synonyms, and linked together using both lexical and semantic relationships.

A Web data mining query language, WebML is proposed to provide data mining operations on the MLDB . WebML is an extension of DMQL. Documents are accessed using data mining operations and lists of keywords. A major feature of WebML are four primitive operations based on the use of concept hierarchies for the keywords [ZaY99] :

1. COVERS : One concept covers another if it is higher (ancestor) in the hierarchy. This coverage is extended to include synonyms as well.

2. COVERED BY: This is the reverse of COVERS in that it reverses to descendents.

3. LIKE: The concept is a synonym.

4. CLOSE TO: One concept is close to another if it is a sibling in the hierarchy. Again, this is extended to include synonyms.

The following example illustrates WebML. The query finds all documents at the level of "www .engr.smu.edu" that have a keyword that covers the keyword cat:

SELECT *

FROM document in ' ' www . engr . srnu . edu ' '

WHERE ONE OF keywords COVERS ' ' cat ' '

WebML allows queries to be stated such that the WHERE clause indicates selection based on the links found in the page, keywords for the page, and information about the domain Where the document is found. Because WebML is an extension of DMQL, data mining functions such as classification, summarization, association rules, clustering, and prediction are included.

Personalization

Another example of Web content mining is in the area of personalization. With personalization, Web access or the contents of a Web page are modified to better fit the desires of the user. This may involve actually creating Web pages that are unique per user or using the desires of a user to determine what Web documents to retrieve.

With personalization, advertisements to be sent to a potential customer are chosen based on specific knowledge concerning that customer. Unlike targeting, personalization may be performed on the target Web page. The goal here is to entice a current customer to purchase something he or she may not have thought about purchasing. Perhaps the simplest example of personalization is the use of a visitor' s name when he or she VISits a page. Personalization is almost the opposite of targeting. With targeting, businesses display advertisements at other sites visited by their users. With personalization, when a particular person visits a Web site, the advertising can be designed specifically for that person. MSNBC, for example, allows personalization by asking the user to enter his or her zip code and favorite stock symbols [msnOO] . Personalization includes such techniques as use of cookies, use of databases, and more complex data mining and machine learning strategies [BDH+95] . Example 7 . 1 illustrates a more complex use of personalization. Personalization may be perfprmed in many ways-some are not data mining. For example, a Web site may require that a visitor log on and provide information.

This not only facilitates storage of personalization information (by ID), but also avoids a common problem of user identification with any type of Web mining. Mining activities related to personalization require examining Web log data to uncover patterns of access behavior by use. This may actually fall into the category of Web usage mining.

EXAMPLE 7.1

Wynette Holder often does online shopping through XYZ.com. Every time she visits their site, she must first log on using an ID. This ID is used to track what she purchases as well as what pages she visits. Mining of the sales and Web usage data is performed by XYZ to develop a very detailed user profile for Wynette. This profile in tum is us□d to personalize the advertising they display. For example, Wynette loves chocolate. This is evidenced by the volume of chocolate she has purchased (and eaten) during the past year. When Wynette logs in, she goes directly to pages containing the clothes she is interested in buying. While looking at the pages, XYZ shows a banner ad about some special sale on Swiss milk chocolate. Wynette cannot resist. She immediately follows the link to this page and adds the chocolate to her shopping cart. She then returns to the page with the clothes she wants.

Personalization can be viewed as a type of clustering, classification, or even prediction. Through classification, the desires of a user are determined based on those for the class. With clustering, the desires are determined based on those users to which he or she is determined to be similar. Finally, prediction is used to predict what the user really wants to see. There are three basic types of Web page: personalization [MCSOO] :

• Manual techniques perform personalization through user registration preferences or via the use of rules that are used to classify individuals based on profiles or demographics.

• Collaborative filtering accomplishes personalization by recommending information (pages) that have previously been given high ratings from similar users.

• Content-based filtering retrieves pages based on similarity between them and user profiles.

One of the earliest uses of personalization was with My Yahoo ! [MPROO]. With My Yahoo ! a user himself personalizes what the screen looks like [YahOO]. He can provide preferences in such areas as weather, news, stock quotes, movies, and sports.

Once the preferences are set up, each time the user logs in, his page is displayed. The personalization is accomplished by the user explicitly indicating what he wishes to see.

Some observations about the use of personalization with My Yahoo! are [MPROO]:

• A few users will create very sophisticated pages by utilizing the customization provided.

• Most users do not seem to understand what personalization means and use only use the default page.

• Any personalization system should be able to support both types of users.

This personalization is not automatic, but more sophisticated approaches to personalization actually use data mining techniques to determine the user preferences. An automated personalization technique predicts future needs based on past needs or the needs of similar users.

News Dude uses the interestingness of a document to determine if a user is interested in it [BP99]. Here interestingness is based on the similarity between the document and that of what the user wishes. Similarity is measured by the co-occurrence of words in the documents and a user profile created for the user. The target application for News Dude is news stories. News Dude actually prunes out stories that are too close to stories the user has already seen. These are determined to be redundant articles. News Dude uses a two-level scheme to determine interestingness. One level is based on recent articles the user has read, while the second level is a more long-term profile of general interests.

Thus, a short-term profile is created that summarizes recent articles read, and a longterm profile is created to summarize the general interests. A document is found to be interesting if it is sufficiently close to either. It was shown that the use of this two-level approach works better than either profile by itself [MPROO].

Another approach to automatic personalization is that used by Firefly. Firefly is based on the concept that humans often base decisions on what they hear from others. If someone likes a TV show, a friend of that person may also like the

program. User profiles are created by users indicating their preferences. Prediction of a user's desires are then made based on what similar users like. This can be viewed as a type of clustering. This approach to Web mining is referred to as collaborative filtering. The initial application of Firefly has been to predict music that a user would like. Note that there is no examination of the actual content of Web documents, simply a prediction based on what similar users like. (One might argue whether this is really a content based Web mining approach.)

Another collaborative approach is called Web Watcher. Web Watcher prioritizes links found on a page based on a user profile and the results of other users with similar profiles who have visited this page [JFM97]. A user is required to indicate the intent of the browsing session. This profile is then matched to the links the user follows.

WEB STRUCTUR E MINING

Web structure mining can be viewed as creating a model of the Web organization or a portion thereof. This can be used to classify Web pages or to create similarity measures between documents. We have already seen some structure mining ideas presented in the content mining section. These approaches used structure to improve on the effectiveness of search engines and crawlers.

7.3.1 PageRank

The PageRank technique was designed to both increase the effectiveness of search engines and improve their efficiency [PBMW98]. PageRank is used to measure the importance of a page and to prioritize pages returned from a traditional search engine using keyword searching. The effectiveness of this measure has been demonstrated by the success of Google [GooOO] . (The name Google comes from the word googol, which is 10^{100}.) The PageRank value for a page is calculated based on the number of pages that point to it. This is actually a measure based on the number of backlinks to a page. A backlink is a link pointing to a page rather than pointing out from a page. The measure is not simply a count of the number of backlinks because a weighting is used to provide more importance to backlinks coming from important pages. Given a page p, we use Bp to be the set of pages that point to p, and Fp to be the set of links out of p. The PageRank of a page p is defined as [PBMW98]

$$PR(p) = c \sum_{q \in B_p} \frac{PR(q)}{N_q} \qquad (7.4)$$

(7.4)

Here N_q = I F_q |. The constant c is a value between 0 and 1 and is used for normalization. A problem, called rank sink, that exists with this PageRank calculation is that when a cyclic reference occurs (page A points to page B and page B points to page A), the PR value for these pages increases. This problem is solved by adding an additional term to the formula:

$$\mathrm{PR}'(p) = c \sum_{q \in B_p} \frac{\mathrm{PR}(q)}{N_q} + cE(v) \qquad (7.5)$$

where c is maximized. Here E (v) is a vector that adds an artificial link. This simulates a random surfer who periodically decides to stop following links and jumps to a new page. E(v) adds links of small probabilities between every pair of nodes. The PageRank technique is different from other approaches that look at links. It does not count all links the same. The values are normalized by the number of links in the page.

Clever

One recent system developed at IBM, Clever, is aimed at finding both authoritative pages and hubs [CDK+99] . The authors define an authority as the "best source" for the requested information [CDK+99]. In addition, a hub is a page that contains links to authoritative pages. The Clever system identifies authoritative pages and hub pages by creating weights. A search can be viewed as having a goal of finding the best hubs and authorities.

Because of the distributed and unsupervised development of sites, a user has no way of knowing whether the information contained within a Web page is accurate. Currently, there is nothing to prevent someone from producing a page that contains not only errors, but also blatant lies. In addition, some pages might be of a higher quality than others. These pages are often referred to as being the most authoritative. Note that this is different from relevant. A page may be extremely relevant, but if it contains factual errors, users certainly do not want to retrieve it. The issue of authority usually does not surface in traditional IR. Hyperlink-induced topic search (HITS) finds hubs and authoritative pages [Kle99a].

The HITS technique contains two components:

• Based on a given set of keywords (found in a query), a set of relevant pages (perhaps in the thousands) is found.

• Hub and authority measures are associated with these pages. Pages with the highest values are returned.

The HITS algorithm is outlined in Algorithm 7 .1. A search engine, S E, is used to find a small set, root set (R), of pages, P, which satisfy the given query, q. This set is then expanded into a larger set, base set (B), by adding pages linked either to or from R. This is used to induce a subgraph of the Web. This graph is the one that is actually examined to find the hubs and authoriti es. In the algorithm, we use the notation G(B, L) to indicate that the graph (subgraph) G is composed of vertices (pages in this case) B and directed edges or arcs (links) L. The weight used to find authorities, xp. and the weight used to find hubs, Yp . are then calculated on G. Because pages at the same site often point to each other, we shopld not really use the structure of the links between these pages to help find hubs and authorities. The algorithm therefore removes these links from the graph. Hubs should point to many good authori ties, and authorities should be pointed to by many hubs. This observation is the basis for the weight calculations shown in the algorithm. An

implementation of the weight calculations using an adjacency matrix is found in the literature [Kle99a]. The approach is basically to iteratively recalculate the weights until they converge. The weights are normalized so that the sum of the squares of each is 1. Normally, the number of hubs and authori ties found is each between 5 and 10.

ALGORITHM 7.1

Input :

W //WWW viewed as a directed graph

q //Query

s //Support

Output :

A //Set of authority pages

H //Set of hub pages

HITS algorithm

R = SE(W, q)

B = R U {pages linked to from R} U {pages that link to pages in R} ;

G(B, L) = Subgraph of W induced by B;

G(B, L^1) = Delete links in G within same site ;

$x_p = \sum_q$ where $(q,p) \in L^1$ y_q; // Find authority weights;
$y_p = \sum_q$ where $(p,q) \in L^1$ x_q; // Find hub weights;

A= {p I p has one of the highest x_p);

H = {p I p has one of the highest Y_p);

7.4 WEB USAGE MINING

Web usage mining performs mining on Web usage data, or Web logs. A Web log is a listing of page reference data. Sometimes it is referred to as clickstream data because each entry corresponds to a mouse click. These logs can be examined from either a client perspective or a server perspective. When evaluated fr om a server perspective, mining uncovers information about the sites where the service resides. It can be used to Improve the design of the sites. By evaluating a client's sequence of clicks, information about a user (or group of users) is detected. This could be used to perform prefet ching and caching of pages. Example 7.2 fr om [XDO la] illustrates Web usage mining.

EXAMPLE 7.2

The webmaster at ABC Corp. learns that a high percentage of users have the following pattern of reference to pages: (A, B,A,C). This means that a user accesses page A, then page B, then back to page A, and finally to page C. Based on this observati on, he determines that a link is needed directly to page C fr om page B. He then adds this link.

Web usage mining can be used for many different purposes. By looking at the sequence of pages a user accesses, a profile about that user could be developed, thus aiding in personalization. With site mining, the overall quality and effectiveness of the pages at the site can be evaluated. One taxonomy of Web usage mining applications has included [SCDTOO] :

• Personalization for a user can be achieved by keeping track of previously accessed pages. These pages can be used to identify the typical browsing behavior of a user and subsequently to predict desired pages.

• By determining fr equent access behavior for users, needed links can be identified to improve the overall performance of fu ture accesses.

• Information concerning frequently accessed pages can be used for caching.

• In addition to modifications to the linkage structure, identifying common access behaviors can be used to improve the actual design of Web pages and to make other modifications to the site. For example, suppose that visitors to an e-commerce site can be identified as customers or noncustomers. The behavior of customers can be compared with that for those who do not purchase anything. This can be used to identify changes to the overall design. It may be determined that many visitors never get past a particular page. That target page can be improved in an attempt to tum these visitors into customers.

• Web usage patterns can be used to gather business intelligence to improve sales and advertisement.

• Gathering statistics concerning how users actually access Web pages may or may not be viewed as part of mining.

Web usage mining actually consists of three separate types of activities [SCDTOO] :

• Preprocessing activities center around reformatting the Web log data before processing.

• Pattern discovery activities form the maj or portion of the mining activities because these activities look to find hidden patterns within the log data.

• Pattern analysis is the process of looking at and interpreting the results of the discovery activities.

There are many issues associated with using the Web log for mining purposes:

• Identification of the exact user is not possible from the log alone.

• With a Web client cache, the exact sequence of pages a user actually visits · difficult to uncover from the server site. Pages that are rereferenced may be found in the cache.

• There are many security, privacy, and legal issues yet to be solved. For example, is the set of pages a person visits actually private information? Should a Web browse actually divulge information to other companies about the habits of its users? After all, this information could be valuable to potential advertisers.

7.4. 1 Preprocessing

The Web usage log probably is not in a format that is usable by mining applications. As With any data to be used in a mining application, the data may need to be reformatted and cleansed. There are, in addition, some issues specifically related to the use of Web logs. Steps that are part of the preprocessing phase include cleansing, user identification session identification, path completion, and formatting [CMS99] .

DEFINITION 7.1. Let P be a set of literals, called pages or clicks, and U be a set of users. A log is a set of triples { (u_1 , p_1,... , t_1) , . . . , (u_n , P_n , t_n) } where u_i € U p_i € P, and ti is a timestamp. Standard log data consist of the following: source site, destination site, and timestamp, as shown in Definition 7 . 1 . The source and destination sites could be listed as a URL or an IP address. The definition assumes that the source site is identified by a user ID and the destination site is identified by a page ID. Additional data such as Web browser information also may be included. Before processing the log, the data may be changed in several ways. For security or privacy reasons, the page addresses may be changed into unique (but nonidentifying) page identifications (such as alphabetic characters). This conversion also will save storage space. In addition, the data may be cleansed by removing any irrelevant information. As an example, the log entries with figures (gif, jpg, etc.) can be removed.

Data from the log may be grouped together to provide more information. All pages visited from one source could be grouped by a server to better understand the patterns of page references from each user (source site). Similarly, patterns from groups of sites may be discovered. References to the same site may be identified and examined to better understand who visits this page.

A common technique is for a server site to divide the log records into sessions. As shown in Definition 7.2 from [XDOl a], a session is a set of page references from one source site during one logical period. Historically, a session would be identified by a user logging into a computer, performing work, and then logging off. The login and logoff represent the logical start and end of the session. With Web log data, this is harder to determine. Several approaches can be used to identify these logical periods:

• Combine all records from the same source site that occur within a time period.

• Add records to a session if they are from the same source site and the time between two consecutive tiinestamps is less than a certain threshold value.

NCR uses an approach based on the second concept. Any inactive period of 30 minutes or more ends a session [SPOO]. Empirical results have shown that 25.5 minutes is appropriate [CP95].

DEFINITION 7.2. Let L be a log. A **session** S is an ordered list of pages accessed by a user, i.e., $S = \langle\langle p_1, t_1\rangle, \langle p_2, t_2\rangle, \ldots, \langle p_n, t_n\rangle\rangle$, where there is a user $u_i \in U$ such that $\{\langle u_i, p_1, t_1\rangle, \langle u_i, p_2, t_2\rangle, \ldots, \langle u_i, p_n, t_n\rangle\} \subseteq L$. Here $t_i \leq t_j$ iff $i \leq j$. Since only the ordering of the accesses is our main interest, the access time is often omitted. Thus, we write a session S as $\langle p_1, p_2, \ldots, p_n\rangle$.

Associated with each session i s a unique identifier, which i s called a session ID. The length of a session S is the number of pages in it, whilch is denoted as len(S). Let database D be a set of such sessions, and the total length of D be len(D) = $\sum_{s \in D} len(S)$.

There are many problems associated with the preprocessing activities, and most of these problems center around the correct identification of the actual user. User identification is complicated by the use of proxy servers, client side caching, and corporate firewalls. Tracking who is actually visiting a site (and where they come from) is difficult.

Even though a visit to a Web page will include a source URL or IP address that indicates the source of the request, this may not always be accurate in determining the source location of the visitor. Users who access the Internet through an Internet service provider (ISP) will all have the source location of that provider. It is not unique to the individual. In addition, the same user may use different ISPs. Also, there will be many users accessing the Web at the same time from one machine. Cookies can be used to assist in identifying a single user regardless of machine used to access the Web. A cookie is a file that is used to maintain client-server information between accesses that the client makes to the server. The cookie file is stored at the client side and sent to the server with each access.

Identifying the actual sequence of pages accessed by a user is complicated by the use of client side caching. In this case, actual pages accessed will be missing from the server side log. Techniques can be used to complete the log by predicting missing pages.

Path completion is an attempt to add page accesses that do not exist in the log but that actually occurred. Some missing pages can be easily added. For example, if a user visits page A and then page C, but there is no link from A to C, then at least one page in this path is missing. Algorithms are used both to infer missing pages and to generate an approximate timestamp.

7.4.2 Data Structures

Several unique data structures have been proposed to keep track of patterns identified during the Web usage mining process. A basic data structure that is one possible alternative is called a trie. A trie is a rooted tree, where each path from the

root to a leaf represents a sequence. Tries are used to store strings for pattern-matching applications.

Each character in the string is stored on the edge to the node. Common prefixes of strings are shared. A problem in using tries for many long strings is the space required. This isv illustrated in Figure 7.5(a), which shows a standard trie for the three strings {ABOUT, CAT, CATEGORY}. Note that there are many nodes with a degree of one. This is a waste of space that is solved by compressing nodes together when they have degrees of one. Ftgure 7.5(b) shows a compressed version of this trie. Here a path

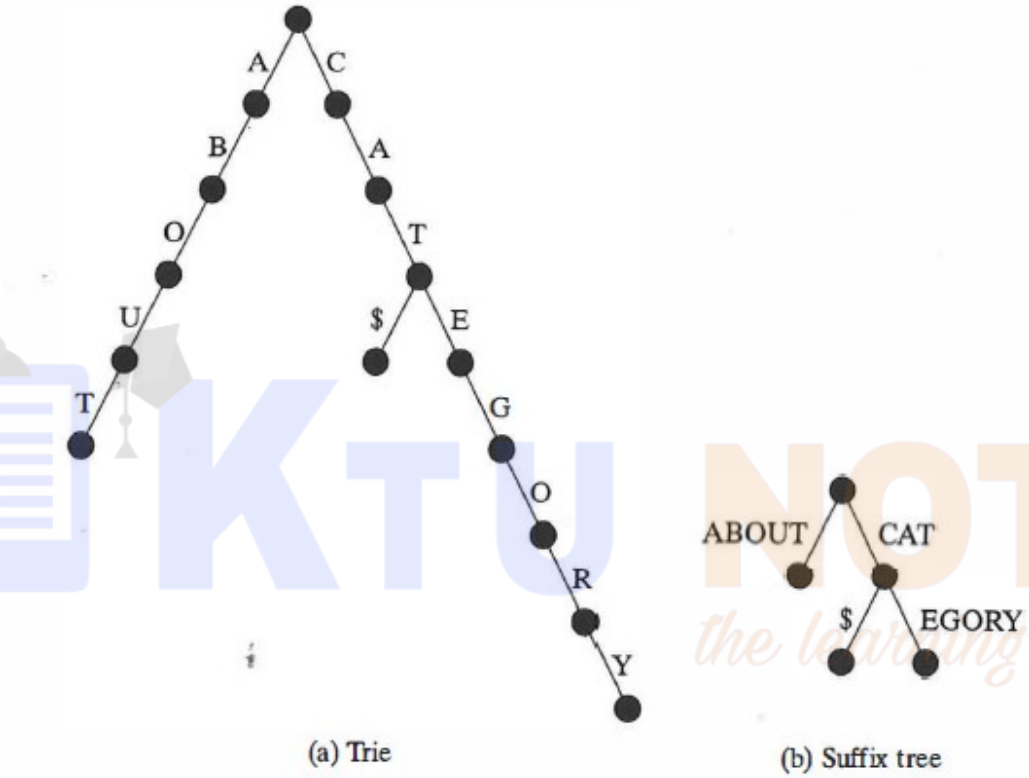


FIGURE 7.5: Sample tries.

consisting of nodes with single children is compressed to one edge. Note in both trees the extra edge labeled "$." This symbol (or any symbol that is not in the alphabet and is used to construct the strings) is added to ensure that a string that is actually a prefix of another (CAT is a prefix of CATEGORY) terminates in a leaf node.

The compressed trie is called a suffix tree. A suffix tree has the following characteristics:

- Each internal node except the root has at least two children.
- Each edge represents a nonempty subsequence.
- The subsequences represented by sibling edges begin with different symbols.

With the help of a suffix tree, it is efficient not only to find any subsequence in a sequence, but also to find the common subsequences among multiple sequences. A suffix tree can also be constructed from a sequence in time and space linear in the length of the sequence. When given one session of page references, many different patterns may be found. The exact number of patterns depends on the exact definition of the pattern to be found (discussed in subsection 7.4.3). Example 7.3 illustrates this idea.

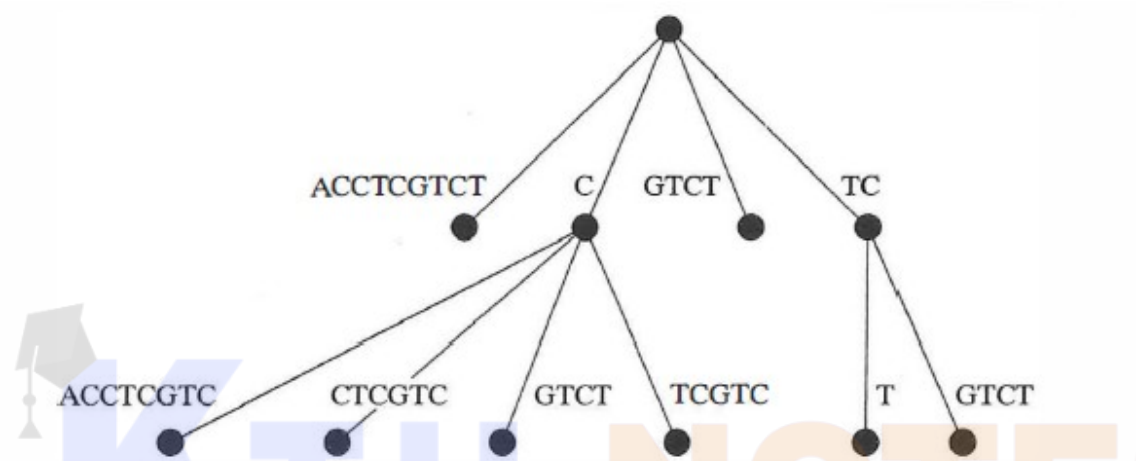


FIGURE 7.6: Sample suffix tree.

EXAMPLE 7.3

Suppose that one session has been identified to be (C, A, C, C, T , C , G, T, C, T). Many different patterns exist in this session. As a matter of fact, we could identify patterns starting at the first character, or the second, or any other. The suffix tree created for this session is shown in Figure 7.6. This tree does not contain the special "$" edges.

A slight variation on this suffix tree that is used to build a suffix tree for multiple sessions is called a generalized suffix tree (GST).

7 .4.3 Pattern Discovery

The most common data mining technique used on clickstream data is that of uncovering traversal patterns. A traversal pattern is a set of pages visited by a user in a session. Other types of patterns may be uncovered by Web usage mining. For example, association rules can look at pages accessed together in one session independent of ordering. Similar traversal patterns may be clustered together to provide a clustering of the users. This is different from clustering of pages, which tends to identify similar pages, not users.

Several different types of traversal patterns have been examined. These patterns differ in how the patterns are defined. The differences between the different types of patterns can be described by the following features:

• Duplicate page references (backward traversals and refreshes/reloads) may or may not be allowed.

• A pattern may be composed only of contiguous page references, or alternatively of any pages referenced in the same session.

• The pattern of references may or may not be maximal in the session. A frequent pattern is maximal if it has no subpattern that is also frequent.

Patterns found using different combinations of these three properties may be used to discover different features and thus may be used for different purposes. Knowledge of contiguous page references frequently made can be useful to predict future references

TABLE 7. 1 : Comparison of Different Types of Traversal Patterns (from [XDO la])

TABLE 7.1: Comparison of Different Types of Traversal Patterns (from [XD01a])

	Ordering	Duplicates	Consecutive	Maximal	Support
Association rules	N	N	N	N	$\frac{\text{freq}(X)}{\text{\# transactions}}$
Episodes	Y[1]	N	N	N	$\frac{\text{freq}(X)}{\text{\# time windows}}$
Sequential patterns	Y	N	N	Y	$\frac{\text{freq}(X)}{\text{\# customers}}$
Forward sequences	Y	N	Y	Y	$\frac{\text{freq}(X)}{\text{\# forward sequences}}$
Maximal frequent sequences	Y	Y	Y	Y	$\frac{\text{freq}(X)}{\text{\# clicks}}$

[1]Serial episodes are ordered, parallel episodes are not, and general episodes are partially ordered.

and thus for prefetching and caching purposes. Knowledge of backward traversals often followed can be used to improve the design of a set of Web pages by adding new links to shorten future traversals. The maximal property is used primarily to reduce the number of meaningful patterns discovered. The use of such

performance improvements as user side caching may actually alter the sequences visited by a user and impact any mining of the Web log data at the server side.

The different types of traversal patterns that have been studied and how they view these three features are shown in Table 7 . 1 (from [XDO la]). Example 7.4 illustrates a set of sessions to be used throughout this section. The sessions are listed in order, and all timestamps have been removed.

EXAMPLE 7.4

The XYZ Corporation maintains a set of five Web pages: {A, B, C, D , E}. The following sessions (listed in timestamp order) have been created: D = {S1 = {U1 , (A, B, C) } , s2 = { U2, (A , C) } , S3 = {U1 , (B, C, E) } , S4 = {U3 , (A , C, D , C, E)}}. Here we have added to each session the user ID. Suppose the support threshold is 30%.

Association Rules. Association rules can be used to find what pages are accessed together. Here we are · really finding large itemsets. A page is regarded as an item, and a session is regarded as a transaction with both duplicates and ordering ignored. The support is defined to be the number of occurrences of the itemset divided by the number of transactions or sessions. The application of the Apriori algorithm to the data in Example 7.2 is shown in Example 7.5.

EXAMPLE 7.5

Since there are four transactions and the support is 30%, an itemset must occur in at least two sessions. During the first scan, we find that L 1 = {{A}, {B}, {C}, {E}}, so C2 = { {A, B}, {A, C}, {A, E}, {B, C}, {B, E}, {C, E}}. Counting these is scan two, we find L2 = {{A, C}, {B, C}, {C, E}} and then generate C3 = {{A, B, C}, {A , C, E}, {B, C, E}}. Counting, we find that none of these are large. The large itemsets are then

L = {{A}, {B}, {C}, {E}, {A, C}, {B, C) , {C, E}}

Sequential Patterns. Although initially proposed for use with market basket data, sequential patterns have also been applied to Web access logs. A sequential pattern (as applied to Web usage mining) is defined as an ordered set of pages that satisfies a given support and is maximal (i.e., it has no subsequence that is also frequent). Support is defined not as the percentage of sessions with the pattern, but rather the percentage of customers who have the pattern. Since a user may have many sessions, it is possible that a sequential pattern could span sessions. It also need not be contiguously accessed pages. A k-sequence is a sequence of length k (i.e., is it has k pages in it).

Algorithm 7.2 outlines the steps needed to find sequential patterns. After the sort step to put the data in the correct order, the remaining steps are somewhat similar to those of the Apriori algorithm. The sort step creates the actual customer sequences, which are the complete reference sequences from one user (across transactions). During the first scan it finds all large 1-itemsets. Obviously, a frequent 1 -itemset is the same as a frequent 1-sequence. In subsequent scans, candidates are generated from the large itemsets of the previous scans and then are counted. In

counting the candidates, however, the modified definition of support must be used. In the algorithm we show that AprioriAll is used to perform this step.

ALGORITHM 7.2

Input :

D = {S1 , S2 , . . . , Sk} //Database of sessions

s / /Support

Output : Sequential patterns

Sequential patterns algorithm:

D = sort D on user- ID and time of first page reference in each session ;

find L_1 in D ;

L = AprioriAll(D, s , L_1) ;

f ind maximal reference sequences from L ;

Generating sequential patterns for Example 7.5 is shown in Example 7.6. Here Ci represents the candidate i-sequences and Li are the large i-sequences.

EXAMPLE 7.6

In this example, user U1 actually has two transactions (sessions). To find his sequential patterns, we must think of his sequence as the actual concatenation of those pages in S1 and S3 . Also, since support is measured not by transactions but by users, a sequence is large if it is contained in at least one customer' s sequence. After the sort step, we have that D = (S1 = {U1, (A, B, C)} , S3 = {U1 , (B, C, E) } , S2 = {U2, (A , C) } , S4 = { U3 , (A, C, D, C, E) }. We find L 1 {{A } , {B } , { C} , { D} , { E}} since each page is referenced by at least one customer. The fo llowing table outlines the steps taken by AprioriAll:

There are variations of this algorithm and several techniques used to improve the performance. The set of customer sequences is reformatted after L 1 is fo und. Each transaction is replaced with one that consists only of pages fr om L 1 . Candidates may be pruned before counting by removing any candidates that have subsequences that are not large .

Variations on AprioriAll are prop osed to avoid generating so many candidates. In effect, these improvements are used only to avoid generating sequences that are not maximal.

$$C_1 = \{\langle A\rangle, \langle B\rangle, \langle C\rangle, \langle D\rangle, \langle E\rangle\}$$
$$L_1 = \{\langle A\rangle, \langle B\rangle, \langle C\rangle, \langle D\rangle, \langle E\rangle\}$$

$$C_2 = \{\langle A, B\rangle, \langle A, C\rangle, \langle A, D\rangle, \langle A, E\rangle, \langle B, A\rangle, \langle B, C\rangle, \langle B, D\rangle, \langle B, E\rangle, \langle C, A\rangle, \langle C, B\rangle, \langle C, D\rangle, \langle C, E\rangle, \langle D, A\rangle, \langle D, B\rangle, \langle D, C\rangle, \langle D, E\rangle, \langle E, A\rangle, \langle E, B\rangle, \langle E, C\rangle, \langle E, D\rangle\}$$
$$L_2 = \{\langle A, B\rangle, \langle A, C\rangle, \langle A, D\rangle, \langle A, E\rangle, \langle B, C\rangle, \langle B, E\rangle, \langle C, B\rangle, \langle C, D\rangle, \langle C, E\rangle, \langle D, C\rangle, \langle D, E\rangle\}$$

$$C_3 = \{\langle A, B, C\rangle, \langle A, B, D\rangle, \langle A, B, E\rangle, \langle A, C, B\rangle, \langle A, C, D\rangle, \langle A, C, E\rangle, \langle A, D, B\rangle, \langle A, D, C\rangle, \langle A, D, E\rangle, \langle A, E, B\rangle, \langle A, E, C\rangle, \langle A, E, D\rangle, \langle B, C, E\rangle, \langle B, E, C\rangle, \langle C, B, D\rangle, \langle C, B, E\rangle, \langle C, D, B\rangle, \langle C, D, E\rangle, \langle C, E, B\rangle, \langle C, E, D\rangle, \langle D, C, B\rangle, \langle D, C, E\rangle, \langle D, E, C\rangle\}$$
$$L_3 = \{\langle A, B, C\rangle, \langle A, B, E\rangle, \langle A, C, B\rangle, \langle A, C, D\rangle, \langle A, C, E\rangle, \langle A, D, C\rangle, \langle A, D, E\rangle, \langle B, C, E\rangle, \langle C, B, E\rangle, \langle C, D, E\rangle, \langle D, C, E\rangle\}$$

$$C_4 = \{\langle A, B, C, E\rangle, \langle A, B, E, C\rangle, \langle A, C, B, D\rangle, \langle A, C, B, E\rangle, \langle A, C, D, B\rangle, \langle A, C, D, E\rangle, \langle A, C, E, B\rangle, \langle A, C, E, D\rangle, \langle A, D, C, E\rangle, \langle A, D, E, C\rangle\}$$
$$L_4 = \{\langle A, B, C, E\rangle, \langle A, C, B, E\rangle, \langle A, C, D, E\rangle, \langle A, D, C, E\rangle\}$$

$$C_5 = \emptyset$$

The WAP-tree (web access pattern) has been proposed to facilitate efficient counting. This tree is used to store the sequences and their counts. Once the tree is built, the original database of patterns is not needed. Each node in the tree is associated with an event (a page found at a particular time by a user). The node is labeled with the event and a count that is associated with the pattern prefix that ends at that event. Only individual frequent events are added to the tree.

Frequent Episodes. Episodes, which or iginally were prop osed for telecommunication alarm analysis, can also be applied to Web logs. All pages (corresponding to events) are ordered by their access time, and the users usually need not be identified (i.e., no sessions). By definition, an episode is a partially ordered set of pages [MTV95]. In addition, the individual page accesses must occur within a part icular time fr ame. A serial episode is an episode in which the events are totally ordered. Note that they need not be contiguous, however. A parallel episode is a set of events where there need not be any particular ordering. They still do need to satisfy the time constraint, however. Finally, a general episode is one where the events satisfy some partial order. Note that even though these seem similar to the idea of sequential patterns and association rules, the added constraint of a time window does make an episode different from either of these. The original definition has no concept of user, but, of course, the idea of an episode could be applied to events by one user or across users. In addition, episodes need not be maximal.

Example 7.7 illustrates the concept of episodes applied to the data in Example 7.4.

Here we keep the original ordering of the events.

EXAMPLE 7.7

The XYZ Corporation maintains a set of five Web pages $\{A, B, C, D, E\}$. Assume that the data in Example 7.2 have the following sequence (independent of user): $\langle A, B, C, A, C, B, C, A, C, D, C, E\rangle$. To find episodes, we must know the time window, so the following sequence shows each event with an integer timestamp: $\langle (A, 1), (B, 2), (C, 2), (A, 7), (C, 10), (B, 10), (C, 12), (A, 12), (C, 13), (D, 14), (C, 14), (E, 20)\rangle$. Suppose that we wish to find general episodes where the support threshold is 30% and the time window is 3. This means that only events that occur within a time window of 3 are valid. We assume that the time window is the difference between the last event and the first event in the episode. To illustrate episodes, Figure 7.7 illustrates the ordering of events as shown in a *DAG (directed acyclic graph)* where arcs are used to represent temporal ordering. The arcs are labeled with the time between successive events. Starting at the first event and looking at the maximum window size of 3, we see that we have two serial episodes: AC and AB. B and C occur as parallel episodes. Starting at the event looking at time 12, we have the following serial episodes: ACD, ACC, CCC, CCD, AC, CC, CC, CD. We have two parallel episodes: A and C, and C and D. There also is a general episode that can be seen as the subgraph from time 12 to time 14. When taking the frequency into account, an episode must occur a certain number of times in all windows.

Maximal Frequent Forward Sequences.

One approach to mining log traversal patterns is to remove any backward traversals [CPY98]. Each raw session is transformed into forward reference (i.e., removes the backward traversals and reload s/refreshes), from which the traversal patterns are then mined using improved level-wise algorithms. For example, for the session (A, B, A, C) in Example 7.2, the resulting forward sequences are (A, B) and (A, C). Looking at Example 7.8 and the sequence (A, B, C, A, C, B, C, A, C, D, E), we find the fo llowing maximal forward references:

(A, B, C) , (A, C), (A, C, D), (A , C , E)

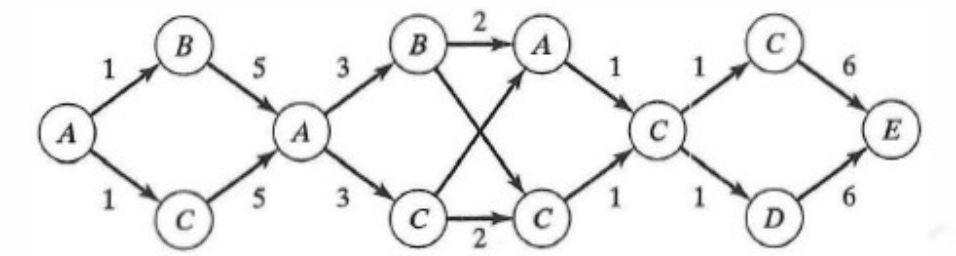


FIGURE 7.7: DAG for episodes.

As observed by the authors, the "real" access patterns made to get to the really used pages would not include these backward references. They assume that the backward reference is included only because of the structure of the pages, not because they really want to do this. The resulting set of forward references are called maximal forward references. They are called maximal because they are not subsequences of other forward references. The set of important reference patterns are those that occur with frequency above a desired threshold. In actuality, we are interested in finding consecutive subsequences within the forward references. A large reference sequence is a reference sequence (consecutive subsequence of a maximal forward reference) that appears more than a minimum number of times. The minimum number threshold is called the support.

Algorithm 7.3 outlines the steps needed to mine maximal reference sequences [CPY98] . After the maximal forward references are fo und, all subsequences of these references that occur greater than the required support are identified. Those large reference sequences that are not subsequences of other large reference sequences are found in step 3 and become the large reference sequences.

ALGORITHM 7.3

Input :

D = {s1 , s2 , ... , sk) //Database of sessions

s //Support

Output :

Maximal reference sequences

Maximal frequent forward sequences algorithm:

find maximal forward referenc es from D;

find large reference sequences from the maximal ones ;

find maximal reference sequences from the large ones ;

Maximal Frequent Sequences. The transformation used to remove backward references also loses potentially useful information; for example, from the two

forward sequences (A, B) and (A, C), we could not tell whether a direct link to page C from page B is needed, as shown in Example 7.2.

With maximal frequent sequences (MFS), all four properties in Table 7.1 are required. Since an MFS could potentially start with any page (click) in any session, the definition of support assumes that the number of clicks is in the denominator. Thus, the support of a sequence X is defined to be

$$\frac{freq(X)}{len(D)} = \frac{freq}{¿\, clicks}$$

A sequence X is frequent if its support is above a minimum threshold. An MFS must be maximal. Example 7.8 (from [XDOla]) shows the mining of MFS.

EXAMPLE 7.8

Given D = { (A, B, C, D, E, D, C, F),(A, A, B, C, D, E),(B, G, H, U, V),(G, H, W)}. The first session has backward traversals, and the second session has a reload/refresh on page A. Here len(D) = 22. Let the minimum support be Srnin = 0.09. This means that we are looking at finding sequences that occur at least two times. There are two maximal fre quent sequences: (A, B, C, D, E) and (G, H). Both sequences occur two times.

Algorithm 7.4 (from [XDO la]) shows the OAT(Online Adaptive Traversal Patterns) algorithm designed to find MFS. It utilizes a suffix tree to store patterns. One suffix tree is created for all sessions. Counts of patterns are maintained in the tree. A unique feature of OAT is its ability to adapt to the availabe amount of main memory. If the suffix tree is too big to fit into memory, it is compressed and the algorithm continues.

Details concerning the exact techniques used for compression can be found in the literature [XDO la].

ALGORITHM 7.4

Input : S1 , S2 , . .. , Sn : sessions

Smin : minimum support threshold

M: main memory size

Output :

Al l maximal frequent sequenc es (MFSs)

OAT algorithm:

ST =an emp ty suffix tree ;

//first scan

for i from 1 to n do

```
end for
//if insufficient main memory with inclusion of Si ;
//compress the suffix tree using frequent sequences;
if mem(ST U Si) > M , then
ST = OAT.compress(ST);
endif
//update the suffix tree with inclusion of si
ST = update(ST, Si) ;
if interrupted by the user, then
//do a depth -first traversal of ST and output the MFSs .
MFS_depth_first(ST.root) ;
endif
//second scan
if there are sequences not completely counted, then
count them in an additional scan ,
endif
output the MFSs in the suffix tree .
```

Pattern Analysis

Once patterns have been identified, they must be analyzed to detennine how that information can be used. Some of the generated patterns may be deleted and determined not to be of interest.

Recent work has proposed examining Web logs not only to identify fre quent types of traversal patterns, but also to identify patterns that are of interest because of their uniqueness or statistical properties [WUMO O]. Patterns fo und need not have contiguous page references. A Web mining query language, MINT, facilitates the statement of interesting properties. The idea of a sequence is expanded to the concept of what the authors call a g-sequence. A g-sequence is a vector that consists not only of the pages visited (events) but also of wildcards. For example, the g-sequence b * c stands for a sequence consisting of b, any number of pages, then c. With the use of wildcards, it is indicated that the events need not be contiguous. More complicated g-sequences can indicate specific constraints on the number of events that replace the wildcard. With MINT, selection of patterns that satisfy a g-sequence template are accomplished. The selection constraints may also include restrictions on support.

Some of the thrust of this work has been in comparing the differences between traversal patterns of the customers of an e-business site and those that are not customers [SPFOO] . Visitors to a site have been classified as short-time visitors, active investigators, and customers [BPW96] . Prepro cessing first filters out the visitors who are short-time. Using concept hierarchies, the contents of the Web pages are then abstracted to more general concepts. The log is then divided into those for customers and those for noncustomers . Each log is then examined to find patterns based on any desired requirements (such as freq uency). The patterns found across the two logs are then compared for similarity . Similarity is detennined using the fol lowing rule [SPFOO] :

• Two patterns are comparable if their g-sequences have at least the first n pages the same. Here n is supplied by the user.

In addition, only fragments of patterns that occur frequently are considered. The goal of this work is to increase the number of customers. Noncustomer patterns with no comparable customer patterns indicate that some changes to the link structure or Webpage designs may be in order. The project proposes rules and the use of a proxy server to dynamically change the link structures of pages.

Text Mining. Graph mining:- Apriori based approach for mining frequent subgraphs. Social Network Analysis:- characteristics of social networks. Link mining:- Tasks and challenges

Text Mining

Most previous studies of data mining have focused on structured data, such as relational,transactional, and data warehouse data. However, in reality, a substantial portion of the available information is stored in text databases (or document databases), which consist of large collections of documents from various sources, such as news articles, research papers, books, digital libraries, e-mail messages, andWeb pages. Text databases are rapidly growing due to the increasing amount of information available in electronic form, such as electronic publications, various kinds of electronic documents, e-mail, and theWorldWideWeb (which can also be viewed as a huge, interconnected, dynamic text database). Nowadays most of the information in government, industry, business, and other institutions are stored electronically, in the form of text databases. Data stored in most text databases are semistructured data in that they are neither completely unstructured nor completely structured. For example, a document may contain a few structured fields, such as title, authors, publication date, category, and so on, but also contain some largely unstructured text components, such as abstract and contents. There have been a great deal of studies on the modeling and implementation of semistructured data in recent database research. Moreover, information retrieval techniques, such as text indexing methods, have been developed to handle unstructured documents.

Traditional information retrieval techniques become inadequate for the increasingly vast amounts of text data. Typically, only a small fraction of the many available documents will be relevant to a given individual user.Without knowing what could be in the documents, it is difficult to formulate effective queries for analyzing and extracting useful information from the data. Users need tools to compare different documents, rank the importance and relevance of the documents, or find patterns

and trends across multiple documents. Thus, text mining has become an increasingly popular and essential theme

in data mining.

10.4.1 **Text Data Analysis and Information Retrieval**

“What is information retrieval?” Information retrieval (IR) is a field that has been developing in parallel with database systems for many years. Unlike the field of database systems,which has focused on query and transaction processing of structured data, information retrieval is concerned with the organization and retrieval of information from a large number of text-based documents. Since information retrieval and database systems each handle different kinds of data, some database systemproblems are usually not present in information retrieval systems, such as concurrency control, recovery, transaction management, and update. Also, some common information retrieval problems are usually not encountered in traditional database systems, such as unstructured documents, approximate search based on keywords, and the notion of relevance.

Due to the abundance of text information, information retrieval has found many applications. There exist many information retrieval systems, such as on-line library catalog systems, on-line document management systems, and the more recently developed Web search engines.

A typical information retrieval problem is to locate relevant documents in a document collection based on a user’s query, which is often some keywords describing an information need, although it could also be an example relevant document. In such a search problem, a user takes the initiative to “pull” the relevant information out from the collection; this is most appropriate when a user has some ad hoc (i.e., short-term) information need, such as finding information to buy a used car. When a user has a long-term information need (e.g., a researcher’s interests), a retrieval system may also take the initiative to “push” any newly arrived information item to a user if the item is judged as being relevant to the user’s information need. Such an information access process is called information filtering, and the corresponding systems are often called filtering systems or recommender systems. From a technical viewpoint, however, search and filtering share many common techniques. Below we briefly discuss the major techniques in information retrieval with a focus on search techniques.

Basic Measures for Text Retrieval: Precision and Recall

“Suppose that a text retrieval system has just retrieved a number of documents for me based on my input in the form of a query. How can we assess how accurate or correct the system was?” Let the set of documents relevant to a query be denoted as fRelevantg, and the set of documents retrieved be denoted as {Retrieved}. The set of documents that are both relevant and retrieved is denoted as

{Relevant} U {Retrieved}, as shown in the Venn diagram of Figure 10.6. There are two basic measures for assessing the quality of text retrieval:

Precision, recall, and F-score are the basic measures of a retrieved set of documents.

These three measures are not directly useful for comparing two ranked lists of documents because they are not sensitive to the internal ranking of the documents in a retrieved set.

In order to measure the quality of a ranked list of documents, it is common to compute an average of precisions at all the ranks where a new relevant document is returned. It is also common to plot a graph of precisions at many different levels of recall; a higher curve represents a better-quality information retrieval system. For more details about these measures, readers may consult an information retrieval textbook, such as [BYRN99].

- **Precision:** This is the percentage of retrieved documents that are in fact relevant to the query (i.e., "correct" responses). It is formally defined as

$$precision = \frac{|\{Relevant\} \cap \{Retrieved\}|}{|\{Retrieved\}|}.$$

- **Recall:** This is the percentage of documents that are relevant to the query and were, in fact, retrieved. It is formally defined as

$$recall = \frac{|\{Relevant\} \cap \{Retrieved\}|}{|\{Relevant\}|}.$$

An information retrieval system often needs to trade off recall for precision or vice versa. One commonly used trade-off is the **F-score**, which is defined as the harmonic mean of recall and precision:

$$F\_score = \frac{recall \times precision}{(recall + precision)/2}.$$

The harmonic mean discourages a system that sacrifices one measure for another too drastically.

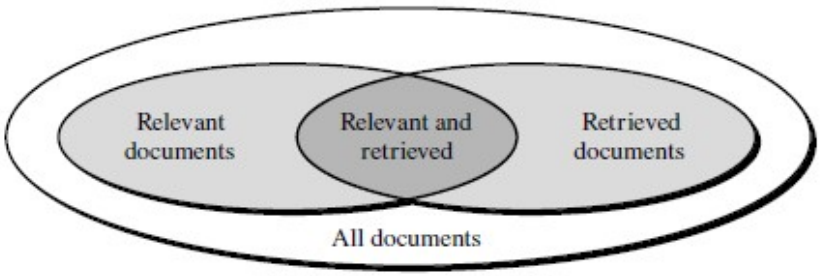


Figure 10.6 Relationship between the set of relevant documents and the set of retrieved documents.

Graphs become increasingly important in modeling complicated structures, such as circuits, images, chemical compounds, protein structures, biological networks, social networks, the Web, workflows, and XML documents. Many graph search algorithms have been developed in chemical informatics, computer vision, video indexing, and text retrieval. With the increasing demand on the analysis of large amounts of structured data, graph mining has become an active and important theme in data mining.

Among the various kinds of graph patterns, frequent substructures are the very basic patterns that can be discovered in a collection of graphs. They are useful for characterizing graph sets, discriminating different groups of graphs, classifying and clustering graphs, building graph indices, and facilitating similarity search in graph databases.

Recent studies have developed several graph mining methods and applied them to the discovery of interesting patterns in various applications. For example, there have been reports on the discovery of active chemical structures in HIV-screening datasets by contrasting the support of frequent graphs between different classes. There have been studies on the use of frequent structures as features to classify chemical compounds, on the frequent graph mining technique to study protein structural families, on the detection of considerably large frequent subpathways in metabolic networks, and on the use of frequent graph patterns for graph indexing and similarity search in graph databases.

Although graph mining may include mining frequent subgraph patterns, graph classification, clustering, and other analysis tasks, in this section we focus on mining frequent subgraphs.We look at various methods, their extensions, and applications.

9.1.1 **Methods for Mining Frequent Subgraphs**

Before presenting graph mining methods, it is necessary to first introduce some preliminary concepts relating to frequent graph mining.

We denote the vertex set of a graph g by V(g) and the edge set by E(g). A label function, L, maps a vertex or an edge to a label. A graph g is a subgraph of another graph g0 if there exists a subgraph isomorphism from g to g0. Given a labeled graph data set,

D = {G1,G2, ... ;Gn } , we define support(g) (or frequency(g)) as the percentage (or number) of graphs in D where g is a subgraph. A frequent graph is a graph whose support is no less than a minimum support threshold, min sup.

Example 9.1 Frequent subgraph. Figure 9.1 shows a sample set of chemical structures. Figure 9.2 depicts two of the frequent subgraphs in this data set, given a minimum support of 66.6%.

“How can we discover frequent substructures?” The discovery of frequent substructures usually consists of two steps. In the first step, we generate frequent substructure candidates. The frequency of each candidate is checked in the second step. Most studies on frequent substructure discovery focus on the optimization of the first step, because the second step involves a subgraph isomorphism test whose computational complexity is excessively high (i.e., NP-complete).

In this section, we look at various methods for frequent substructure mining. In general, there are two basic approaches to this problem: an Apriori-based approach and a pattern-growth approach.

Apriori-based Approach

Apriori-based frequent substructure mining algorithms share similar characteristics with Apriori-based frequent itemset mining algorithms (Chapter 5). The search for frequent graphs starts with graphs of small "size," and proceeds in a bottom-up manner by generating candidates having an extra vertex, edge, or path. The definition of graph size depends on the algorithm used.

The general framework of Apriori-based methods for frequent substructure mining is outlined in Figure 9.3.We refer to this algorithm as AprioriGraph. Sk is the frequent substructure set of size k.Wewill clarify the definition of graph sizewhenwe describe specific Apriori-based methods further below. AprioriGraph adopts a level-wise mining methodology. At each iteration, the size of newly discovered frequent substructures is increased by one. These new substructures are first generated by joining two similar but slightly different frequent subgraphs that were discovered in the previous call to AprioriGraph. This candidate generation procedure is outlined on line 4. The frequency of the newly formed graphs is then checked. Those found to be frequent are used to generate larger candidates in the next round.

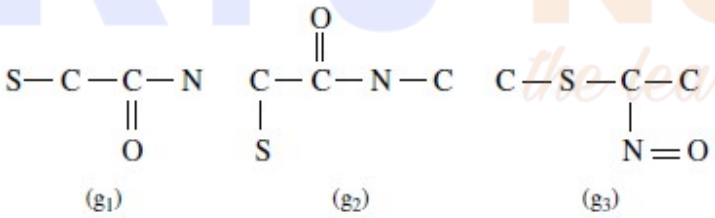


Figure 9.1 A sample graph data set.

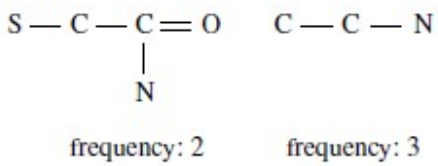


Figure 9.2 Frequent graphs.

The main design complexity of Apriori-based substructure mining algorithms is the candidate generation step. The candidate generation in frequent itemset mining is straightforward. For example, supposewe have two frequent itemsets of size-3: (abc) and (bcd). The frequent itemset candidate of size-4 generated from them is simply (abcd), derived from a join. However, the candidate generation problem in frequent substructure mining is harder than that in frequent itemset mining, because there are many ways to join two substructures.

Recent Apriori-based algorithms for frequent substructure mining include AGM, FSG, and a path-join method. AGM shares similar characteristics with Apriori-based itemset mining. FSG and the path-join method explore edges and connections in an Apriori-based fashion. Each of these methods explores various candidate generation strategies.

The AGM algorithm uses a vertex-based candidate generation method that increases the substructure size by one vertex at each iteration of AprioriGraph. Two size-k frequent graphs are joined only if they have the same size-(k-1) subgraph. Here, graph size is the number of vertices in the graph. The newly formed candidate includes the size-(k-1) subgraph in common and the additional two vertices from the two size-k patterns. Because it is undetermined whether there is an edge connecting the additional two vertices, we actually can form two substructures. Figure 9.4 depicts the two substructures joined by two chains (where a chain is a sequence of connected edges).

Algorithm: AprioriGraph. Apriori-based frequent substructure mining.

Input:

- D, a graph data set;
- min_sup, the minimum support threshold.

Output:

- S_k, the frequent substructure set.

Method:

$S_1 \leftarrow$ frequent single-elements in the data set;
Call AprioriGraph(D, min_sup, S_1);

procedure AprioriGraph(D, min_sup, S_k)

(1) $S_{k+1} \leftarrow \varnothing$;
(2) **for each** frequent $g_i \in S_k$ **do**
(3) **for each** frequent $g_j \in S_k$ **do**
(4) **for each** size $(k+1)$ graph g formed by the merge of g_i and g_j **do**
(5) **if** g is frequent in D and $g \notin S_{k+1}$ **then**
(6) insert g into S_{k+1};
(7) **if** $s_{k+1} \neq \varnothing$ **then**
(8) *AprioriGraph*(D, min_sup, S_{k+1});
(9) **return**;

Figure 9.3 AprioriGraph.

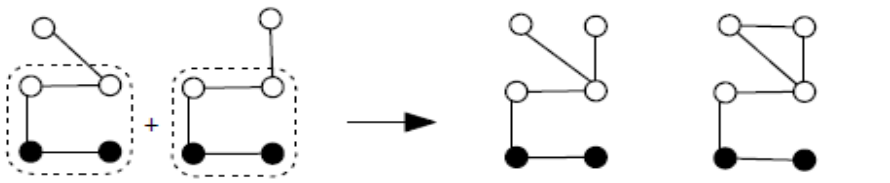


Figure 9.4 AGM: Two substructures joined by two chains.

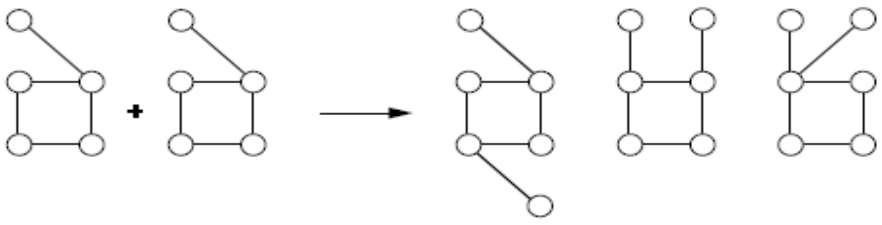


Figure 9.5 FSG: Two substructure patterns and their potential candidates.

The FSG algorithm adopts an edge-based candidate generation strategy that increases the substructure size by one edge in each call of AprioriGraph. Two size-k patterns are merged if and only if they share the same subgraph having k - 1 edges, which is called the core. Here, graph size is taken to be the number of edges in the graph. The newly formed candidate includes the core and the additional two edges fromthe size-k patterns.

Figure 9.5 shows potential candidates formed by two structure patterns. Each candidate has one more edge than these two patterns. This example illustrates the complexity of joining two structures to form a large pattern candidate.

In the third Apriori-based approach, an edge-disjoint path method was proposed, where graphs are classified by the number of disjoint paths they have, and two paths are edge-disjoint if they do not share any common edge. A substructure pattern with k+1 disjoint paths is generated by joining substructures with k disjoint paths. Apriori-based algorithms have considerable overhead when joining two size-k frequent substructures to generate size-(k +1) graph candidates. In order to avoid such overhead, non-Apriori-based algorithms have recently been developed, most of which adopt the pattern-growth methodology. This methodology tries to extend patterns directly from a single pattern. In the following, we introduce the pattern-growth approach for frequent subgraph mining.

Social Network Analysis

The notion of social networks, where relationships between entities are represented as links in a graph, has attracted increasing attention in the past decades. Thus social network analysis, from a data mining perspective, is also called link analysis or link mining. In this section, we introduce the concept of social networks in Section 9.2.1, and study the characteristics of social networks in Section 9.2.2. In Section 9.2.3, we look at the tasks and challenges involved in link mining

9.2.1 **What Is a Social Network?**

From the point of view of data mining, a social network is a heterogeneous and multirelational data set represented by a graph. The graph is typically very large, with nodes corresponding to objects and edges corresponding to links representing relationships or interactions between objects. Both nodes and links have attributes. Objects may have class labels. Links can be one-directional and are not required to be binary.

Social networks need not be social in context. There are many real-world instances of technological, business, economic, and biologic social networks. Examples include electrical power grids, telephone call graphs, the spread of computer viruses, the World Wide Web, and coauthorship and citation networks of scientists. Customer networks and collaborative filtering problems (where product recommendations are made based on the preferences of other customers) are other examples. In biology, examples range from epidemiological networks, cellular and metabolic networks, and food webs, to the neural network of the nematode worm Caenorhabditis elegans (the only creature whose neural network has been completely mapped). The exchange of e-mail messages within corporations, newsgroups, chat rooms,

friendships, sex webs (linking sexual partners), and the quintessential “old-boy” network (i.e., the overlapping boards of directors of the largest companies in the United States) are examples from sociology.

Small world (social) networks have received considerable attention as of late. They reflect the concept of “small worlds,” which originally focused on networks among individuals.

The phrase captures the initial surprise between two strangers (“What a small world!”) when they realize that they are indirectly linked to one another throughmutual acquaintances. In 1967, Harvard sociologist, Stanley Milgram, and his colleagues conducted experiments in which people in Kansas and Nebraska were asked to direct letters to strangers in Boston by forwarding them to friends who they thought might know the strangers in Boston. Half of the letters were successfully delivered through no more than five intermediaries. Additional studies by Milgram and others, conducted between other cities, have shown that there appears to be a universal “six degrees of separation” between any two individuals in theworld. Examples of smallworld networks are shown in Figure 9.16. Small world networks have been characterized as having a high degree of local clustering for a small fraction of the nodes (i.e., these nodes are interconnected with one another), which at the same time are no more than a few degrees of separation from the remaining nodes. It is believed that many social, physical, human-designed, and biological networks exhibit such small world characteristics.

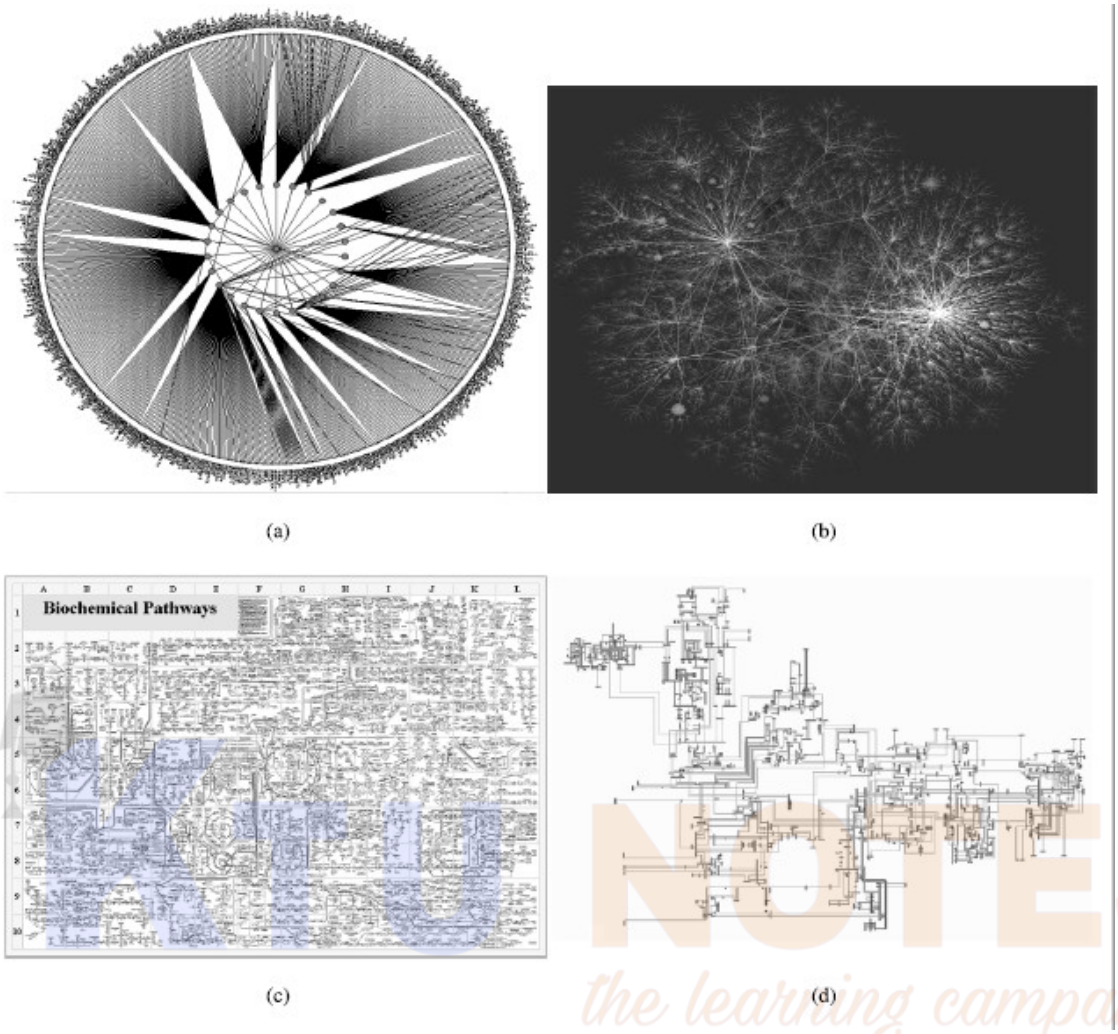


Figure 9.16 Real-world examples of social networks: (a) science coauthor network, (b) connected pages on a part of the Internet, (c) biochemical pathway network, and (d) New York state electric power grid. Figure 9.16 (a), (b), and (c) are from *www.nd.edu/~networks/publications.html#talks0001* by Barabási, Oltvai, Jeong et al. Figure 9.11(d) is from [Str01], available at *http://tam.cornell.edu/Strogatz.html#pub.*

"Why all this interest in smallworld networks and social networks, in general? What is the interest in characterizing networks and in mining themto learn more about their structure?" state and Ohio, and led to widespread power blackouts in many parts of theNortheastern United States and Southeastern Canada,which affected approximately 50 million people.

The interest in networks is part of broader research in the accurate and complete description of complex systems. Previously, the networks available for experimental study were small and few, with very little information available regarding individual nodes. Thanks to the Internet, huge amounts of data on very large social networks are now available.

These typically contain from tens of thousands to millions of nodes. Often, a great deal of information is available at the level of individual nodes. The availability of powerful computers has made it possible to probe the structure of networks. Searching social networks can help us better understand howwe can reach other people. In addition, research on small worlds, with their relatively small separation between nodes, can help us design networks that facilitate the efficient transmission of information or other resources without having to overload the network with too many redundant connections.

For example, it may help us design smarter search agents on theWeb, which can find relevant websites in response to a query, all within the smallest number of degrees of separation from the initial website (which is, typically, a search engine).

9.2.2 **Characteristics of Social Networks**

As seen in the previous section, knowing the characteristics of small world networks is useful in many situations. We can build graph generation models, which incorporate the characteristics. These may be used to predict how a network may look in the future, answering “what-if ” questions. Taking the Internet as an example, we may ask “What will the Internet look like when the number of nodes doubles?” and “What will the number of edges be?”. If a hypothesis contradicts the generally accepted characteristics, this raises a flag as to the questionable plausibility of the hypothesis. This can help detect abnormalities n existing graphs, which may indicate fraud, spam, or Distributed Denial of Service (DDoS) attacks. Models of graph generation can also be used for simulations when real graphs are excessively large and thus, impossible to collect (such as a very large network of friendships). In this section, we study the basic characteristics of social networks as well as a model for graph generation.

“What qualities can we look at when characterizing social networks?” Most studies examine the nodes’ degrees, that is, the number of edges incident to each node, and the distances between a pair of nodes, as measured by the shortest path length. (This measure embodies the small world notion that individuals are linked via short chains.) In particular, the network diameter is the maximum distance between pairs of nodes. Other node-to-node distances include the average distance between pairs and the effective diameter (i.e., the minimum distance, d, such that for at least 90% of the reachable node pairs, the path length is at most d).

Social networks are rarely static. Their graph representations evolve as nodes and edges are added or deleted over time. In general, social networks tend to exhibit the following phenomena:

1. Densification power law: Previously, it was believed that as a network evolves, the number of degrees grows linearly in the number of nodes. This was known as the constant average degree assumption.However, extensive experiments have shown that, on the contrary, networks becomeincreasingly dense over timewith the average degree increasing (and hence, the number of edges growing superlinearly in the number of nodes). The densification follows the densification power law (or growth power law), which states

$$e(t) \propto n(t)^a,$$

where e(t) and n(t), respectively, represent the number of edges and nodes of the graph at time t, and the exponent a generally lies strictly between 1 and 2. Note that if a = 1, this corresponds to constant average degree over time, whereas a = 2 corresponds to an extremely dense graph where each node has edges to a constant fraction of all nodes.

2. Shrinking diameter: It has been experimentally shown that the effective diameter tends to decrease as the network grows. This contradicts an earlier belief that the diameter slowly increases as a function of network size. As an intuitive example, consider a citation network, where nodes are papers and a citation from one paper to another is indicated by a directed edge. The out-links of a node, v (representing the papers cited by v), are "frozen" at the moment it joins the graph. The decreasing distances between pairs of nodes consequently appears to be the result of subsequent papers acting as "bridges" by citing earlier papers from other areas.

3. Heavy-tailed out-degree and in-degree distributions: The number of out-degrees for a node tends to follow a heavy-tailed distribution by observing the power law, $1/n^a$, where n is the rank of the node in the order of decreasing out-degrees and typically, $0 < a < 2$ (Figure 9.17). The smaller the value of a, the heavier the tail. This phenomena is represented in the preferential attachment model, where each new node attaches to an existing network by a constant number of out-links, following a "rich-get-richer" rule. The in-degrees also followa heavy-tailed distribution, although it tends be more skewed than the out-degrees distribution.

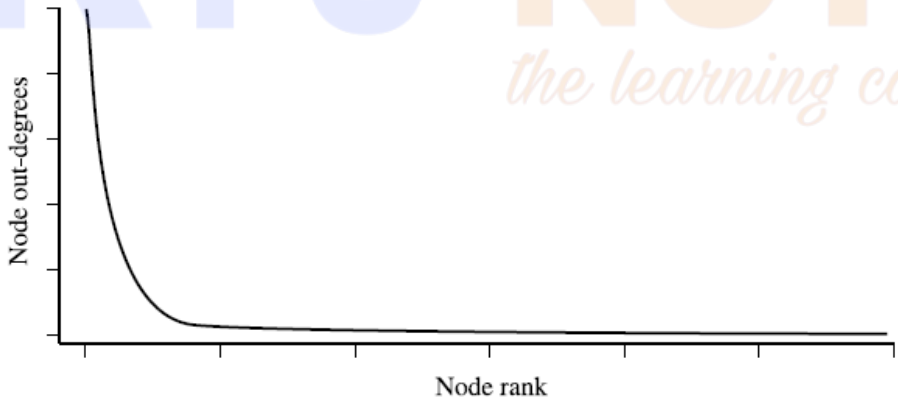


Figure 9.17 The number of out-degrees (*y*-axis) for a node tends to follow a heavy-tailed distribution. The node rank (*x*-axis) is defined as the order of deceasing out-degrees of the node.

A Forest Fire model for graph generation was proposed, which captures these characteristics of graph evolution over time. It is based on the notion that new nodes attach to the network by "burning" through existing edges in epidemic fashion. It uses two parameters, forward burning probability, p, and backward burning ratio, r,

which are described below. Suppose a new node, v, arrives at time t. It attaches to G_t , the graph constructed so far, in the following steps:

1. It chooses an ambassador node, w, at random, and forms a link to w.

2. It selects x links incident to w, where x is a random number that is binomially distributed with mean $(1\text{-}p)^{-1}$. It chooses from out-links and in-links of w but selects in-links with probability r times lower than out-links. Let w1 , w2 , ... wx denote the nodes at the other end of the selected edges.

3. Our new node, v, forms out-links to w1 , w2 , ... wx and then applies step 2 recursively to each of w1 , w2 , ... wx. Nodes cannot be visited a second time so as to prevent the construction from cycling. The process continues until it dies out.

For an intuitive feel of the model, we return to our example of a citation network. The author of a newspaper, v, initially consults w, and follows a subset of its references (which may be either forward or backward) to the papers w1;w2; : : : ;wx. It continues accumulating references recursively by consulting these papers.

Several earlier models of network evolution were based on static graphs, identifying network characteristics from a single or small number of snapshots, with little emphasis on finding trends over time. The Forest Fire model combines the essence of several earlier models, while considering the evolution of networks over time. The heavy-tailed out-degrees property is observed in that, owing to the recursive nature of link formation, new nodes have a good chance of burning many edges and thus producing large outdegrees.

The heavy-tailed in-degrees property is preserved in that Forest Fire follows the “rich-get-richer” rule: highly linked nodes can easily be reached by a new node, regardless of which ambassador the new node starts from. The flavor of a model known as the copying model is also observed in that a new node copies many of the neighbors of its ambassador. The densification power law is upheld in that a new node will have many links near the community of its ambassador, a few links beyond this, and much fewer farther away. Rigorous empirical studies found that the shrinking diameter property was upheld. Nodes with heavy-tailed out-degrees may serve as “bridges” that connect formerly disparate parts of the network, decreasing the network diameter.

9.2.3 **Link Mining: Tasks and Challenges**

“How can we mine social networks?” Traditional methods of machine learning and data mining, taking, as input, a random sample of homogenous objects from a single relation, may not be appropriate here. The data comprising social networks tend to be heterogeneous, multirelational, and semi-structured. As a result, a new field of research has emerged called link mining. Link mining is a confluence of research in social networks, link analysis, hypertext and Web mining, graph mining, relational learning, and inductive logic programming. It embodies descriptive and predictive modeling. By considering links (the relationships between objects), more information is made available to the mining process. This brings about several new tasks. Here, we list these tasks with examples from various domains:

1. Link-based object classification. In traditional classificationmethods, objects are classified based on the attributes that describe them. Link-based classification predicts the category of an object based not only on its attributes, but also on its links, and on the attributes of linked objects.

Web page classification is a well-recognized example of link-based classification. It predicts the category of aWeb page based on word occurrence (words that occur on the page) and anchor text (the hyperlink words, that is, the words you click on when you click on a link), both of which serve as attributes. In addition, classification is based on links between pages and other attributes of the pages and links. In the bibliography domain, objects include papers, authors, institutions, journals, and conferences.

A classification task is to predict the topic of a paper based on word occurrence, citations (other papers that cite the paper), and cocitations (other papers that are cited within the paper), where the citations act as links. An example from epidemiology is the task of predicting the disease type of a patient based on characteristics (e.g., symptoms) of the patient, and on characteristics of other people with whomthe patient has been in contact. (These other people are referred to as the patients' contacts.)

2. Object type prediction. This predicts the type of an object, based on its attributes and its links, and on the attributes of objects linked to it. In the bibliographic domain, we may want to predict the venue type of a publication as either conference, journal, or workshop. In the communication domain, a similar task is to predict whether a communication contact is by e-mail, phone call, or mail.

3. Link type prediction. This predicts the type or purpose of a link, based on properties of the objects involved. Given epidemiological data, for instance, we may try to predict whether two people who know each other are family members, coworkers, or acquaintances. In another example, we may want to predict whether there is an advisor-advisee relationship between two coauthors. Given Web page data, we can try to predict whether a link on a page is an advertising link or a navigational link.

4. Predicting link existence. Unlike link type prediction, where we know a connection exists between two objects and we want to predict its type, instead we may want to predict whether a link exists between two objects. Examples include predicting whether there will be a link between two Web pages, and whether a paper will cite another paper. In epidemiology, we can try to predict with whom a patient came in contact.

5. Link cardinality estimation. There are two forms of link cardinality estimation. First, we may predict the number of links to an object. This is useful, for instance, in predicting the authoritativeness of aWeb page based on the number of links to it (in-links). Similarly, the number of out-links can be used to identifyWeb pages that act as hubs, where a hub is one or a set ofWeb pages that point to many authoritative pages of the same topic. In the bibliographic domain, the number of citations in a paper may indicate the impact of the paper—the more citations the paper has, the more influential it is likely to be. In epidemiology, predicting the number of links

between a patient and his or her contacts is an indication of the potential for disease transmission.

A more difficult form of link cardinality estimation predicts the number of objects reached along a path from an object. This is important in estimating the number of objects that will be returned by a query. In theWeb page domain, we may predict the number of pages that would be retrieved by crawling a site (where crawling refers to a methodological, automated search through the Web, mainly to create a copy of all of the visited pages for later processing by a search engine). Regarding citations, we can also use link cardinality estimation to predict the number of citations of a specific author in a given journal.

6. Object reconciliation. In object reconciliation, the task is to predict whether two objects are, in fact, the same, based on their attributes and links. This task is common in information extraction, duplication elimination, object consolidation, and citation matching, and is also known as record linkage or identity uncertainty. Examples include predicting whether two websites are mirrors of each other, whether two

citations actually refer to the same paper, and whether two apparent disease strains are really the same.

7. Group detection. Group detection is a clustering task. It predicts when a set of objectsbelong to the same group or cluster, based on their attributes as well as their link structure. An area of application is the identification of Web communities, where a Web community is a collection of Web pages that focus on a particular theme or topic. A similar example in the bibliographic domain is the identification of research communities.

8. Subgraph detection. Subgraph identification finds characteristic subgraphs within networks. This is a formof graph search and was described in Section 9.1. An example from biology is the discovery of subgraphs corresponding to protein structures. In chemistry, we can search for subgraphs representing chemical substructures.

9. Metadata mining. Metadata are data about data. Metadata provide semi-structured data about unstructured data, ranging from text and Web data to multimedia databases. It is useful for data integration tasks in many domains. Metadata mining can be used for schema mapping (where, say, the attribute customer id from one database is mapped to cust number from another database because they both refer to the same entity); schema discovery, which generates schema from semi-structured data; and schema reformulation, which refines the schema based on the mined metadata.

Examples include matching two bibliographic sources, discovering schema from unstructured or semi-structured data on the Web, and mapping between two medical ontologies.

In summary, the exploitation of link information between objects brings on additional tasks for link mining in comparison with traditional mining approaches. The implementation of these tasks, however, invokes many challenges. We examine several of these challenges here:

1. Logical versus statistical dependencies. Two types of dependencies reside in the graph—link structures (representing the logical relationship between objects) and probabilistic dependencies (representing statistical relationships, such as correlation between attributes of objects where, typically, such objects are logically related). The coherent handling of these dependencies is also a challenge for multirelational data mining, where the data to be mined exist in multiple tables.We must search over the different possible logical relationships between objects, in addition to the standard search over probabilistic dependencies between attributes. This takes a huge search space, which further complicates finding a plausible mathematical model. Methods developed in inductive logic programmingmay be applied here,which focus on search over logical relationships.

2. Feature construction. In link-based classification, we consider the attributes of an object as well as the attributes of objects linked to it. In addition, the links may also have attributes. The goal of feature construction is to construct a single feature representing these attributes. This can involve feature selection and feature aggregation. In feature selection, only the most discriminating features are included. Feature aggregation takes amultiset of values over the set of related objects and returns a summary of it. This summary may be, for instance, the mode (most frequently occurring value); the mean value of the set (if the values are numerical); or the median or “middle” value (if the values are ordered). However, in practice, this method is not always appropriate.

3. Instances versus classes. This alludes to whether the model refers explicitly to individuals or to classes (generic categories) of individuals. An advantage of the former model is that it may be used to connect particular individuals with high probability. An advantage of the latter model is that it may be used to generalize to new situations, with different individuals.

4. Collective classification and collective consolidation. Consider training a model for classification, based on a set of class-labeled objects. Traditional classification methods consider only the attributes of the objects. After training, suppose we are given a new set of unlabeled objects. Use of the model to infer the class labels for the new objects is complicated due to possible correlations between objects—the labels of linked objects may be correlated. Classification should therefore involve an additional iterative step that updates (or consolidates) the class label of each object based on the labels of objects linked to it. In this sense, classification is done collectively rather than independently.

5. Effective use of labeled and unlabeled data. Arecent strategy in learning is to incorporate a mix of both labeled and unlabeled data.Unlabeled data can help infer the object attribute distribution. Links between unlabeled (test) data allowus to use attributes of linked objects. Links between labeled (training) data and unlabeled (test) data induce dependencies that can help make more accurate inferences.

6. Link prediction. A challenge in link prediction is that the prior probability of a particular link between objects is typically extremely low. Approaches to link prediction have been proposed based on a number of measures for analyzing the proximity of nodes in a network. Probabilistic models have been proposed as well. For large data sets, it may be more effective to model links at a higher level.

7. Closed versus open world assumption. Most traditional approaches assume that we know all the potential entities in the domain. This “closed world” assumption is unrealistic in real-world applications.Work in this area includes the introduction of a language for specifying probability distributions over relational structures that involve a varying set of objects.

8. Community mining from multirelational networks. Typical work on social network analysis includes the discovery of groups of objects that share similar properties. This is known as community mining. Web page linkage is an example, where a discovered community may be a set of Web pages on a particular topic. Most algorithms for community mining assume that there is only one social network, representing a relatively homogenous relationship. In reality, there exist multiple, heterogeneous social networks, representing various relationships. A new challenge is the mining of hidden communities on such heterogeneous social networks, which is also known as community mining on multirelational social networks.

These challenges will continue to stimulate much research in link mining.

Module 6

Hierarchical clustering Method :- BIRCH, Density based clustering - DBSCAN and OPTICS

Advanced data mining techniques: Introduction to webmining - Web content mining, web structure mining, web usage mining : Text mining

Graph Mining :- Apriori based approach for mining frequent subgraphs. Social Network Analysis :- Characteristics of social networks. Link mining - Tasks and challenges.

Hierarchical based clustering Method - BIRCH

Balanced Iterative Reducing and Clustering Using Hierarchies

→ BIRCH is designed for clustering a large amount of numerical data by integration of hierarchical clustering and other clustering methods such as iterative partitioning.

→ It overcomes the two difficulties of agglomerative clustering methods,

1) Scalability
2) The inability to undo what was done in the previous step.

→ BIRCH introduces two concepts, clustering feature (CF) and clustering feature tree (CF tree), which is used to summarize cluster representations.

→ Given n d-dimensional data objects or points in a cluster,

The centroid x_0 is defined as

$$x_0 = \frac{\sum_{i=1}^{n} x_i}{n}$$

Radius R of the cluster is

$$R = \sqrt{\frac{\sum_{i=1}^{n} (x_i - x_0)^2}{n}}$$ which is the average distance from member objects to the centroid.

Diameter D of the cluster is

$$D = \sqrt{\frac{\sum_{i=1}^{n}\sum_{j=1}^{n} (x_i - x_j)^2}{n(n-1)}}$$, which is the average pairwise distance within a cluster.

Both R & D reflect the tightness of the cluster around the centroid.

→ A clustering feature (CF) is a three dimensional vector summarizing information about clusters of objects.

→ Given n d-dimensional objects or points in a cluster, $\{x_i\}$, then CF of the cluster is defined as

$$CF = \langle n, LS, SS \rangle$$

n = no of points in the cluster

LS = linear sum of the n points $\left(\sum_{i=1}^{n} x_i\right)$

SS = Square sum of the datapoints $\left(\sum_{i=1}^{n} x_i^2\right)$

→ A clustering feature is essentially a summary of the statistics of the given cluster. The zeroth, first and second moments of the cluster.

→ Clustering features are additive.

ie Let C_1 & C_2 be two disjoint clusters with clustering features CF1 & CF2. The clustering feature for the cluster formed by merging C_1 & C_2 is $CF_1 + CF_2$.

→ Clustering features are sufficient for calculating all the measurements that are needed for making clustering decisions in BIRCH.

Eg: Clustering feature

Let $C_1 = \{(2,5), (3,2), (4,3)\}$

CF of C_1

$CF_1 = \{< 3, (9,10), (29,38) >$

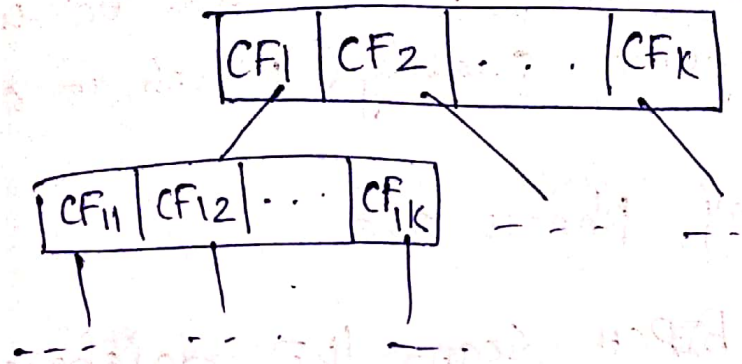
Suppose that C_1 is disjoint to a second cluster C_2 where $CF2 = <3, (35,36), (417,440)>$

CF of the new cluster C_3 formed by merging $C_1 + C_2$ will be

$CF_3 = <3+3, (35+9, 10+36), (29+417, 38+440)>$

$= <6, (44,46), (446,478)>$

→ A CF-tree is a height-balanced tree that stores the clustering features for a hierarchical clustering



→ A non leaf in a tree has descendants or children. The non leaf nodes stores sum of the CFs of their children, and thus summarize clustering information about their children.

→ A CF tree has two parameters :- Branching factor B and threshold T.

→ Branching factor specifies the maximum number of children per nonleaf node.

→ The threshold parameter specifies the maximum diameter of subclusters stored at the leaf node of the tree.

→ BIRCH applies a multiphase clustering technique :- a single scan of the database yields a basic good clustering and one or more additional scans can be used to further improve quality (which is optional).

→ The primary phases are

Phase 1: BIRCH scans the database to build an initial in memory CF-tree which can be viewed as a multilevel compression of the data that preserves the inherent clustering structure of the data.

Phase 2: BIRCH applied a (selected) clustering algorithm to cluster the leaf nodes of the CF tree, which removes the sparse clusters as outliers and groups dense clusters into large ones.

→ For phase 1, CF tree is built dynamically as objects are inserted. Thus the method is incremental.

→ An object is inserted into the ~~smallest~~ closest leaf entry (subcluster). If the diameter of the subcluster stored in the leaf node after insertion is larger than the threshold value then the leaf node and possibly other nodes are split. This is similar to insertion and node split in B+ trees.

→ After the insertion of the new object, the information about it is passed towards the root of the tree.

→ Once the CF tree is built, any clustering algm, such as partitioning algm can be used with the CF tree in Phase 2.

→ Complexity of algm is $O(n)$ n = no of objects to be clustered.

Disadvantage

→ CF tree can hold a limited no of entries

→ BIRCH do not perform well when clusters are not spherical in shape

Advantage

→ Scalability

→ good quality of clusters

Density Based Methods

→ To discover clusters with arbitrary shape, density based clustering methods have been developed.

→ These typically regard clusters as dense regions of objects in a data space that are separated by regions of ~~objects is~~ low density (representing noise).

1. DBSCAN

[Density Based Spatial clustering of Applications with Noise]

→ It is a density based clustering method based on connected regions with sufficiently high density.

→ This algorithm grows regions with sufficiently high density into clusters and discovers clusters of arbitrary shapes in spatial database with noise.

→ It defines a cluster as a maximal set of density connected points.

Terminologies or Definitions used in DBSCAN

→ The neighborhood within a radius ε of a given object is called ε-neighborhood of the object.

→ If the ε-neighborhood of an object contains at least a minimum number, MinPts, of objects, then the object is called core object

→ Given a data set of objects, D, we say that an object p is directly density reachable from object q if p is within the ε-neighborhood of q, and q is a core object.

→ An object p is density reachable from object q with respect to ε and MinPts in a set of objects, D, if there is a chain of objects $P_1, P_2 \ldots P_n$ where $P_1 = q$ and $P_n = p$ such that P_{i+1} is directly density reachable from P_i with respect to ε and

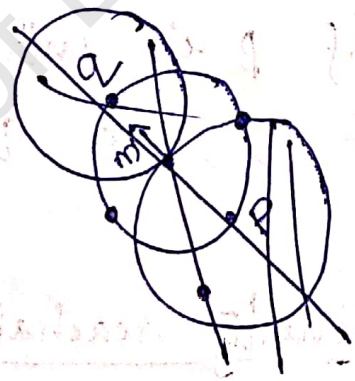
MinPts, for $1 \leq i \leq n$, $p_i \in D$

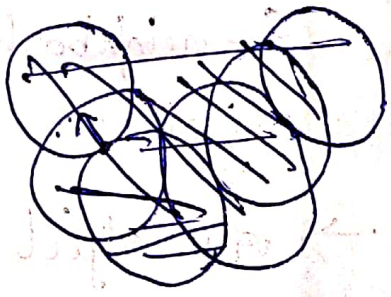
→ An object p is <u>density connected</u> to object q with respected to ε and MinPts in a set of objects D, if there is an object $o \in D$ such that both p and q are density-reachable from o with respect to ε and MinPts.

Density reachability is the transitive closure of directly density reachability and this relationship is asymmetric. Only core objects are mutually density reachable. Density connectivity is a symmetric relation.

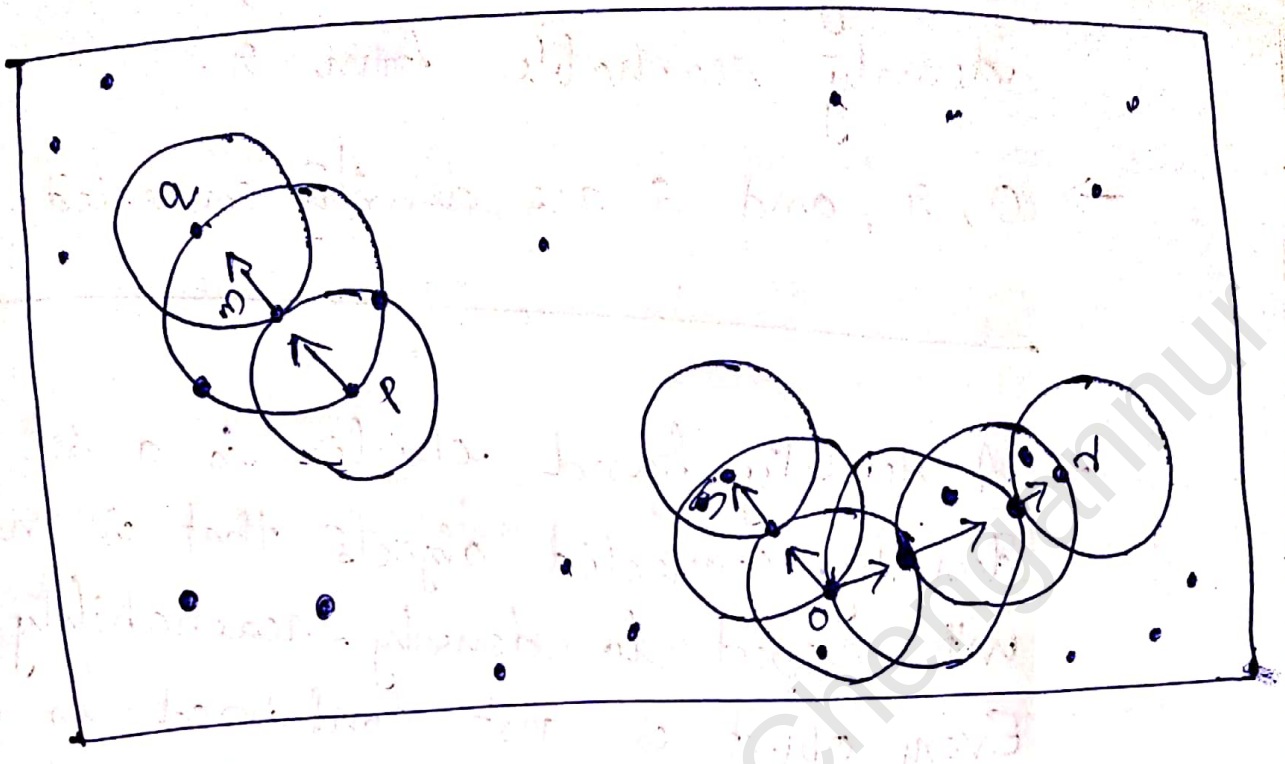
Eg:

Edition 2: Kamber
Page no 419
Refer figure





Eg:



Let Min pts = 3

→ Of the labled points m, p, o, and r are core objects because each is in an ε-neighborhood containing at least three points.

→ q is directly density reachable from m. m is directly density reachable from p and vice versa.

→ q is (indirectly) density-reachable from p because q is directly density-reachable from m and m is directly density reachable from p. However, p is not density-reachable from q because q is not a core object. Similarly, r and s are

density reachable from o, and o is density reachable from r.

→ o, r, and s are density connected.

→ A density based cluster is a set of density-connected objects that is maximal with respect to density-reachability. Every object is not contained in any cluster is considered to be noise.

* → How does DBSCAN find clusters.?

DBSCAN searches for clusters by checking the ε-neighborhood of each point in the database. If the ε-neighborhood of a point p contains more than MinPts, a new cluster with p as a core object is created. DBSCAN then iteratively collects directly density reachable objects from these core objects, which may involve the merge of a few density reachable clusters. The process terminates when no new point can be added to any cluster.

→ Complexity is $O(n \log n)$ if a spatial index is used, otherwise $O(n^2)$

→ With appropriate setting of the user specified parameters ε & MinPts, the algm is effective at finding arbitrary shaped clusters.

OPTICS [Ordering Points to Identify Clustering Structure]

Disadvantages of DBSCAN

* User should specify the parameter values ε and MinPts. It is difficult to set these parameters for real world high dimensional datasets. Slight different settings may lead to very different clustering of the data.

→ To overcome these difficulties, a cluster analysis method called OPTICS was Proposed.

→ Rather than produce a data set clustering explicitly, OPTICS computes an augmented clustering ordering for automatic and interactive cluster analysis.
This ordering represents the density based clustering structure of the data.
It contains information that is equivalent to density based clustering obtained from a wide range of parameter settings.
The cluster ordering can be used to extract basic clustering information (such as cluster centers or arbitrary shaped clusters) as well as provide the intrinsic clustering structure

→ In DBSCAN, we can see that for a constant MinPts value, density based clusters with respect to a higher density (ie a lower value for ε) are completely contained in a density connected set obtained with respect to lower density. The parameter ε is a distance, ie neighborhood radius

Therefore, in order to produce a set or ordering of density based clusters, we can extend the DBSCAN algm to process a set of distance parameter values at the same time. To construct the different clustering simultaneously, the objects should be processed in specific order. This order selects an object that is density reachable with respect to the lowest ε value so that clusters with higher density will be finished first.

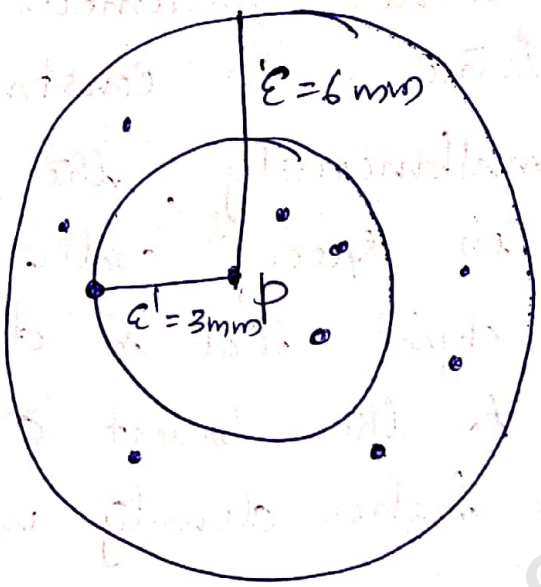
→ Based on this idea, two values need to be stored for each object

1. Core distance :- Core distance of an object p is the smallest ε' value that makes {p} a core object. If p is not a core object, the core distance of p is undefined.

2. The reachability distance of an object q with respect to another object p is the greater value of the core distance of p and Euclidean distance between p & q. If p is not a core

object, the reachability distance b/w p & q is undefined

Eg:

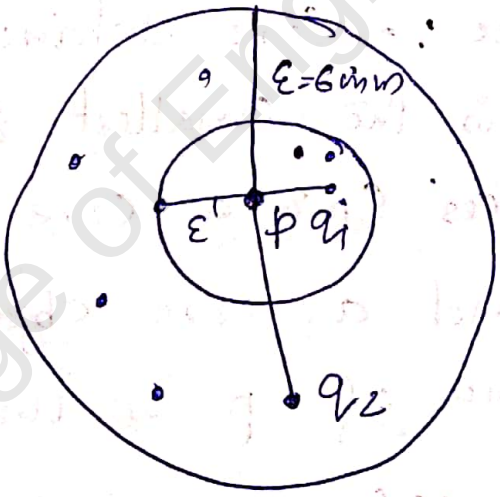


If $\varepsilon = 6$ mm &
$Minpts = 5$.

The core distance of p is ε' [b/w p & fourth closest data object

Core distance of p

$$\text{Core distance} = \begin{cases} \text{undefined} & \text{if } |N_\varepsilon(p)| < Minpts \\ \text{Min dist to smallest distance to } N_\varepsilon(p) & \text{otherwise.} \end{cases}$$



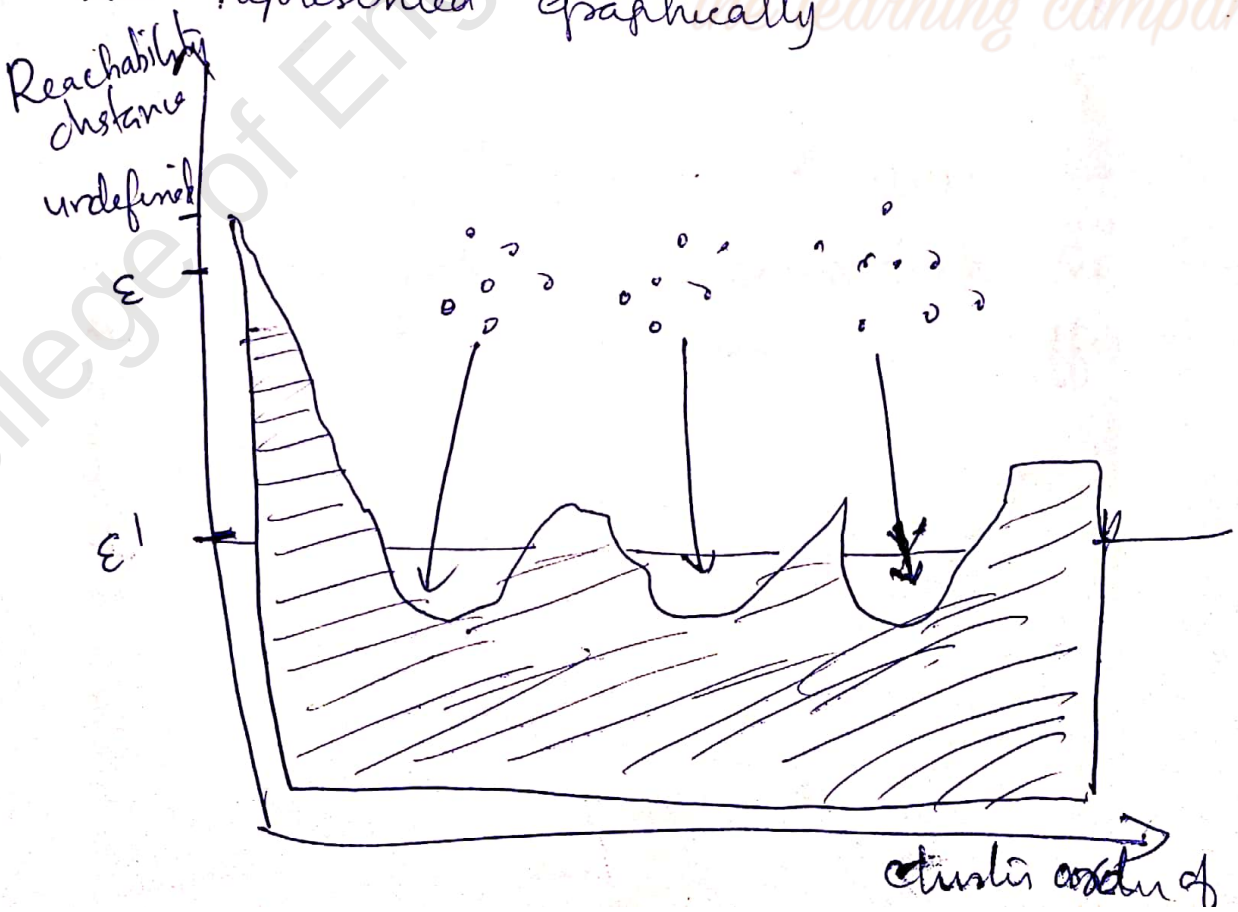
Reachability distance (p, q_1)
$= \varepsilon' = 3$mm

Reachability distance (p, q_2)
$= d(p, q_2)$

→ How these values are used ?

The OPTICS algm creates an ordering of the objects in a database, additionally storing the core distance and a suitable reachability distance for each object. An algm was proposed to extract clusters based on the ordering information produced by OPTICS. Such information is sufficient for the extraction of all density based clustering with respect to any distance ε' that is smaller than the distance ε used in generating the order.

→ The clustering ordering of a dataset can be represented graphically



It is the reachability plot of a simple 2D data set which presents a general overview of how the data are structured & clustered. The data objects are plotted in cluster order together with their respective reachability distance. The three Gaussian bumps reflects three clusters in data set

Complexity of OPTICS algm:- $O(n \log n)$

Graph Mining

→ Graphs become increasingly important in modeling complicated structures, such as circuits, images, chemical compounds etc.

→ Frequent subpatterns are the very basic patterns that can be discovered in a collection of graphs.

→ They are useful for characterizing graph sets, discriminating different groups of graphs, classifying and clustering graphs, building graph indices and facilitating similarity search in graph databases.

Eg of Applications

→ Discovery of active chemical structures in HIV screening datasets by contrasting the support of frequent graphs between different classes

Methods for Mining Frequent Subgraphs.

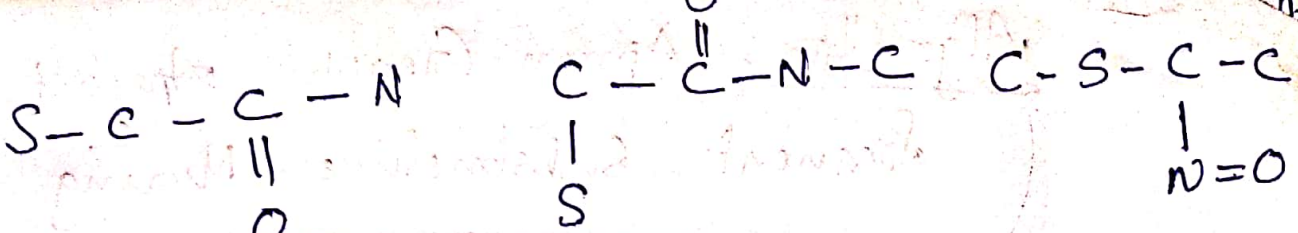
→ Vertex set of a graph g is denoted by V(g) and the edge set by E(g).

→ A label function L, maps a vertex or an edge to a label.

→ A graph g is a subgraph of another graph g', if there exists a subgraph isomorphism from g to g'.

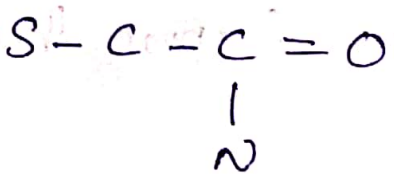
→ Given a labeled graph data set $D = \{G_1, G_2 \ldots G_n\}$, support(g) or frequency(g) is the percentage (number) of graphs in D, where g is a subgraph.

→ A frequent graph is a graph whose support is not less than a minimum support threshold.

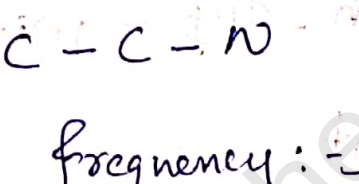
eg 

(g_1) (g_2) (g_3)

↑ A sample graph data set.



frequency: 2



frequency: 3

∴ Frequent subgraphs.

? → How can we discover frequent substructures?

It contains two steps.

1. Generate frequent substructure candidates
2. Check the frequency of each candidate

Apriori based Approach

→ The search for frequent graphs starts with graphs of small size and proceeds in bottom-up manner by generating candidates ~~with~~ having an extra vertex, edge or path.

Algorithm: Apriori Graph: Apriori based frequent substructure Mining

Input:

D, a graph set.

min_sup - minimum support threshold.

Output

S_k, the frequent substructure set.

Method:

S1 ← frequent single elements in the data set;

Call AprioriGraph (D, min_sup, S1);

Procedure: AprioriGraph (D, min_sup, S_k)

1. $S_{k+1} \leftarrow \phi$
2. for each frequent $g_i \in S_k$ do
3. for each frequent $g_j \in S_k$ do
4. for each size (k+1) graph g formed by merge of g_i & g_j do

```
5.      if g is frequent in D & g ∉ S_{k+1} then
6.          insert g into S_{k+1}
7: if S_{k+1} ≠ ϕ
8.      AprioriGraph (D, min-sup, S_{k+1});
9. return
```

S_k is the frequent substructure set of size k. AprioriGraph adopts a level-wise mining methodology. At each iteration, the size of newly discovered frequent substructures is increased by one. The new substructures are first generated by joining two similar but slightly different subgraphs that are discovered in previous call to AprioriGraph. The frequency of the newly formed graphs are then checked. Those found to be frequent are used to generate larger candidates in the next round.

→ Recent Apriori based algms for frequent-substructure mining include AGM, FSG and a path join method (disjoint paths method)

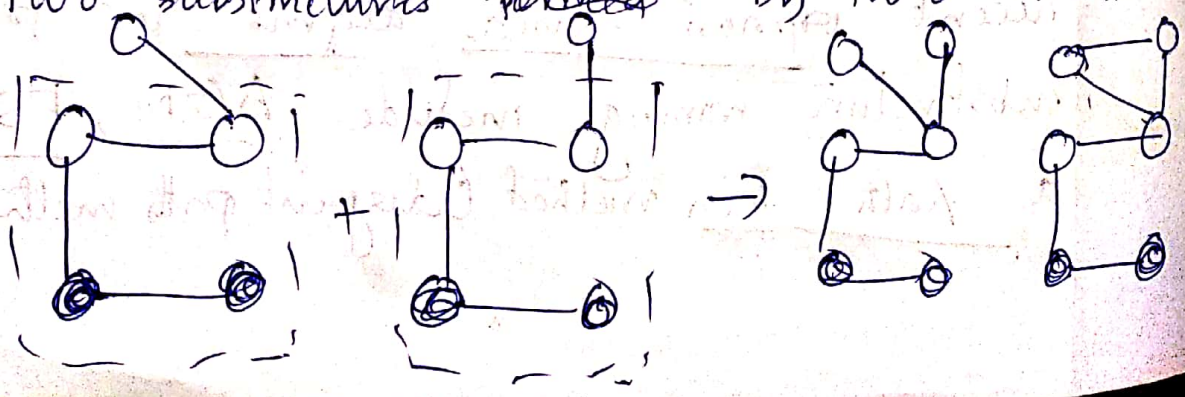
→ The AGM algm uses a vertex-based candidate generation method that increases the substructure size by one vertex at each iteration of Apriori Graph.

→ The size-k frequent graphs are joined only if they have the same size (k-1) subgraphs.

→ Here the Graph size is the no of vertices in the graph.

→ The newly formed candidate includes the size (k-1) subgraphs in common and the additional two vertices form the two size-k patterns. Because it is undetermined whether there is an edge connecting the additional two vertices, we actually can form two substructures.

eg. Two substructures ~~formed~~ joined by two chains

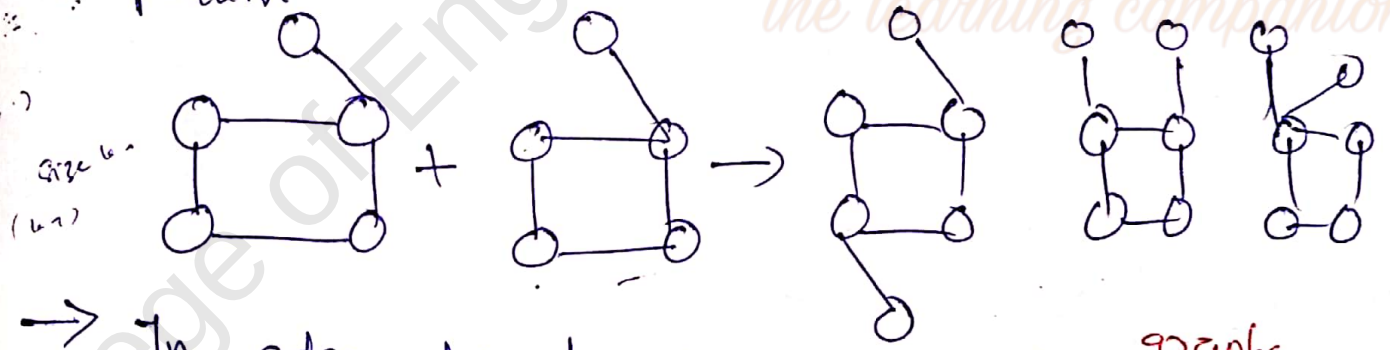


→ The FSM FSG algorithm adopts an edge based candidate generation strategy that increases the substructure size by one edge in each call of Apriori Graph.

→ ~~Two~~ Two size-k patterns are merged if and only if they share the same subgraph having k-1 edges, which is called the core.

→ Here, graph size is taken to be the no of edges in the graph. The newly formed candidate includes the core and the additional two edges from the size k-patterns.

Eg: Potential candidate formed by two ~~size~~ structure ~~pattern~~.



→ In edge disjoint path method, the ~~paths~~ graphs are classified by their the no of disjoint paths they have and the two paths are edge disjoint if they do not share any common edge. A substructure pattern with k+1 disjoint paths is generated by joining substructures with k-disjoint paths

30)

3

1, 4, 5, 8, 12, 13, 14, 15, 16, ~~20~~, 21
23, 24, ~~25~~ 26, 28, 30, 32, 33, 34, 36
37, 39, 43, 48, 49, 50, 52, 54, 56, 57, 58

4

1, 4, 5, ~~11~~, ~~13~~, 14, 15, 16, ~~20~~, 21, 23, 24, 26,
30, 32, 33, ~~34~~ 35, ~~36~~ 50, 54, ~~55~~, 57

→ web mining
→ Types of web data
→ Taxonomy

→ Web Content Mining

→ Most search engines – keyword based
→ WCM improves the IR below.
WCM – agent based – slow since agent performs content mining
database based. → view the database as belong to a database & can use query languages

→ Text mining – WCM is a type of TM.
hierarchy:

→ Pblm associated with retrieval of data from web.

→ Documents are not structured as in DB
HTML – semi structured
XML – structured data

Crawlers [Robot / spider]

→ program that traverses the hypertext structure of data

→ Seed @URL.

- Periodic
  - ↓
  - visit certain no of pages & then stop build an index & replace the existing index.
- Incremental
  - ↓
  - selectively searches the web & update index.

focus crawler – visits the pages related to topic.

Figure –

1° components

1. **hypertext classifier** – associates a relevance score for each document
2. **distiller** – identify hub pages. [pages that contains links to many relevant pages]
3. **Crawler**

Performance objective – high precision rate of harvest rate

→ hard focus
soft focus

→ Content focused crawler — 1st step: content graph; classifier are created; crawling.

~~Advance~~ Content graph updates when crawling.

→ page may not be relevant but contain links to relevant pages.

→ relevant pages may have no links from relevant pages. → Backward crawling

→ Harvest s/m
→ based on use caching & indexing & crawling
→ Two components
→ gatherer — obtains info for indexing from internet service provider → using Essence s/m
→ broker → provide index & query interface

→ Virtual Web View

→ Create Multilayered database (MLDB)
— to handle large unstructured data
→ Distributed DB.
→ Each layer is more generalized than layer beneath it
→ ~~upper layer~~)
→ View of MLDB — virtual web view

→ WebML – to access MLDB
Premitive operations based on concept hierarchies
→ covers
→ covered by
→ Like.
→ Close to

→ Personalization.

→ web access or content of web page modified to fit desires of user
→ It is a type of classification technique

Three types of web page personalization

→ Manual techniques – through user registration preferences
→ collaborative filtering – recommends pages have high rated from similar users
→ content based filtering :- similarity b/w web pages & user profiles.