3.1 INTRODUCTION

With the explosive growth of information resources available on WWW, it has become increasingly necessary for users to utilize automated tools in order to find, extract, filter, and evaluate the desired information and resources. In addition, with the emergence of electronic commerce, it has become imperative for organizations to track and analyze user access patterns. These factors gave rise to the necessity of creating server-side and client-side intelligent systems that can effectively mine for knowledge both across the Internet and in particular web localities. These systems particularly employ Web Mining techniques to achieve this.

Web Mining lies at the cross point between database technology, IR and artificial intelligence (AI) [18]. It is the application of data mining techniques to discover and retrieve hidden information (also called knowledge) from WWW documents and services [12]. Data Mining [13] can be defined as extraction of interesting (non-trivial, implicit, previously unknown and potentially useful) information or patterns from large databases. Thus, Web mining is amalgam of “WWW and Data mining”.

Web mining process is traditionally decomposed of four stages [21]:

1. **Resource Identification**: It is the process of retrieving intended web documents and is accomplished by either web search engines or Meta search engines or simply by crawlers [75, 76]. This stage focuses on one-time analysis of web sites and cannot deal with constantly changing web sites such as news sites. Web server logs also serve as important resources for web mining process.

2. **Preprocessing**: This stage consists of two tasks: selecting interesting data from the downloaded web documents, and transforming this data into a formal representation. Most methods use wrappers (e.g. IBM’s Data Joiner) for extracting task relevant data.
(e.g. names, prices, phone numbers, e-mail addresses, etc.) from documents, and construct tables as formal representations [76]. The content and hyperlink information retrieved from documents can also be regarded as stepping stone for further analysis.

3. **Generalization:** It is the automatic discovery of patterns across multiple web documents. Most methods use data mining functionalities for discovering patterns such as association rules, clusters and classification trees or rules. For instance, Singh et al. [75] proposed a method for detecting association rules that describe the content of a set of scientific online papers; Ghani et al. [76] suggested extracting companies’ data from the web and constructing classification trees for predicting the growth of the economic sectors; Crimmins and Smeaton [77] concentrated on clustering web sites by their content; Gelbukh et al. [78] proposed a method to categorize documents based on a weighted topic hierarchy; and Alexandrov et al. [79] described a method for clustering and classifying interdisciplinary documents based on qualitative and quantitative properties.

4. **Analysis:** Analysis involves the validation and interpretation of the mined patterns. The expert analysts, sometimes supported by graphical interfaces [77], carry out this interpretation.

Web mining research can be classified into three categories as described below.

### 3.2 CATEGORIES OF WEB MINING

Web mining can be divided into three categories [12, 18] as given below:

- Web Content Mining
- Web Structure Mining
- Web Usage Mining

The taxonomy of Web Mining is shown in Fig. 3.1 and their comparison is provided in Table 3.1. These categories of web mining and their applications in the context of search engines are explained in the next subsections.
3.2.1 WEB CONTENT MINING (WCM)

Web Content Mining (WCM) is the process of extracting useful information from the content of Web documents. Content corresponds to the collection of facts contained in a web page consisting of text, images, audio, video, or structured records such as lists and tables.

WCM is related, but is different from data mining and text mining. Data mining deals primarily with structured data, while text mining focuses on unstructured texts. It is related to
data mining because many data mining techniques can be applied in WCM. It is related to
text mining because much of web contents are text based whereas, it is different from data
mining because web data are mainly semi-structured or unstructured. It is also different from
text mining because of the semi-structured nature of the web.

The WCM can be looked upon from two different points of view:

- IR View
- Database (DB) View.

In IR view, most of the researches consider a web page as bag of words. It is based on the
statistics about single words in isolation wherein single words act as features. Thus, it helps
in representing unstructured text. For the semi-structured data, all the research works utilize
the internal structure of HTML web document. R. Kosala et al. [21] summarized the research
works done for unstructured data and semi-structured data from IR view.

In the DB view, a web site is considered as a database. In order to have better information
management and querying on the Web, the techniques involved in this view always try to
infer the structure of the Web site so as to transform it into a database. Most of the WCM
research considers Vector Space Model for document representation. A brief discussion on
the techniques is given in next section.

3.2.1.1 Document Representation: Vector Space Model

The depiction of a set of documents as vectors in a common vector space is known as the
vector space model [43]. This representation is used for many IR operations ranging from
scoring documents on a query, document classification and document clustering. A document
vector captures the relative importance of the terms in a document, wherein each term is
assigned a weight depending on its number of occurrences in the document. The
representation of document collection in vector space model is shown in Fig. 3.2.

The simplest approach is to assign the weight to be equal to the number of occurrences of
term \( t \) in document \( d \). This weighting scheme is referred to as Term Frequency and is denoted
\( TF_{t,d} \), with the subscripts denoting the term and the document in order.
Raw term frequency as above suffers from a critical problem i.e. all terms are considered equally important when it comes to assessing relevancy on a query. On the contrary, certain terms have little or no discriminating power in determining relevance. An immediate solution to this problem is to scale down the weights of terms with high *Collection Frequency*, defined to be the total number of occurrences of a term in the collection.

Instead, it is more commonplace to use a measure called *Document Frequency* $DF_t$, defined to be the number of documents in the collection that contain a term $t$. This is because when trying to discriminate between documents for the purpose of scoring, it is better to use a document-level statistic than to use a collection-wide statistic for the term. Let $N$ be the total number of documents in a collection, the *Inverse Document Frequency* $IDF_t$ of a term $t$ is defined as follows:

$$IDF_t = \log \frac{N}{DF_t}$$  \hspace{1cm} (3.1)

It may be noted that $IDF$ of a rare term will be high, whereas that of a frequent term is likely to be low. Another weighing scheme called *TF-IDF* assigns to term $t$ a weight in document $d$ given by:

$$TF-IDF_t = TF_{t,d} \times IDF_t$$  \hspace{1cm} (3.2)

In other words, *TF-IDF* assigns to term $t$ a weight in document $d$ which is

(1) highest when $t$ occurs many times within a small number of documents (thus lending high discriminating power to those documents);

(2) lower when $t$ occurs fewer times in a document, or occurs in many documents (thus offering a less pronounced relevance signal);
(3) lowest when $i$ occurs in virtually all documents.

At this point, each document can be viewed as a vector with one component corresponding to each term in the dictionary, together with a weight for each component that is given by (3.1). For dictionary terms that do not occur in a document, this weight is zero. This vector form is crucial to score and rank in Web content mining.

To compensate for the effect of document length, the standard way of quantifying the similarity between two documents $d_i$ and $d_j$ is to compute the cosine similarity [80] of their vector representations $V(d_i)$ and $V(d_j)$:

$$\text{Sim}(d_i, d_j) = \frac{V(d_i) \cdot V(d_j)}{|V(d_i)| \cdot |V(d_j)|} \quad (3.3)$$

where the numerator represents the dot product of the vectors $V(d_i)$ and $V(d_j)$, while the denominator is the product of their Euclidean lengths.

In IR, cosine similarity is also applicable in computing the similarity of user queries with documents in the corpus. A query does not uniquely identify a single document in the collection. Instead, several documents may match the query, perhaps with different degrees of relevancy. To compute the relevancy, a query can also be considered as a vector in the Vector Space Model. By viewing a query as a “bag of words”, it can be treated as a very short document. As a consequence, the cosine similarity between the query vector and a document vector can be used as a measure of the score of the document for that query. This type of score can be calculated as:

$$\text{Score}(q, d) = \frac{V(q) \cdot V(d)}{|V(q)| \cdot |V(d)|} \quad (3.4)$$

The resulting scores can then be used to select the top-scoring documents for a query. A document may have a high score for a query even if it does not contain all query terms.

### 3.2.1.2 WCM Research

In the past few years, there has been a rapid expansion of activities in the WCM area. The traditional search engines such as Lycos, Alta Vista, WebCrawler, ALIWEB, MetaCrawler
etc. provide efficient retrieval of information, but they do not generally provide structural information nor categorize, filter, or interpret documents. In recent years these factors have prompted researchers to develop more intelligent tools for IR such as intelligent web agents, as well as to extend DB and data mining techniques to provide a higher level of organization for semi-structured data available on the web. S. Chakrabarti [81] provides an in-depth survey of the research done in the application of the techniques from machine learning, statistical pattern recognition and data mining to analyzing hypertext. The agent based approach to web mining involves the development of sophisticated AI systems that can act autonomously or semi-autonomously on behalf of a particular user to discover and organize web-based information. A brief discussion on the present research directions in this area is given below:

a) **Data/information extraction**: Here, focus is on extraction of structured data from the web pages. There are mainly two types of techniques for information extraction: machine learning and automatic extraction.

b) **Web information integration and schema matching**: Many web sites represent similar information in different ways. To identify or match similar data is an important problem with many practical applications.

c) **Opinion extraction from online sources**: There are many online opinion sources, e.g. customer reviews of products, forums, and chat rooms. Mining opinions is of importance for marketing intelligence and product benchmarking.

d) **Knowledge synthesis**: Concept hierarchies or ontologies are useful in many applications. However, to generate them manually is a very time consuming task. The main challenge is to synthesize and organize the pieces of information on the Web to give the user a coherent picture of the topic domain.

e) **Duplicate Page Detection**: Web contains duplicate pages and mirrored web pages in abundance due to which search engines face a couple of problems in indexing, crawling and caching of web pages. According to various studies [82, 83], the web pages traversed by crawlers comprise of 1.7% to 7% of near duplicates. Thus, efficient identification of near duplicates is a vital issue that needs to be addressed. These pages contain similar or near similar content and vary only in minimal areas of the document like the
advertisements and timestamps. Some of the existing Duplicate Data Detection methods are given below:

- **Shingling Method:** A technique for the estimation of the degree of similarity among pairs of documents was presented in 1997 by Broder et al. [84], known as shingling, which does not rely on any linguistic knowledge other than tokenizing the documents into a list of words. Here, all word sequences (called shingles) of adjacent words are extracted and if two documents contain the same set of shingles they are considered equivalent and if their sets of shingles appreciably overlap, they are exceedingly similar. In their experiments, it was identified that in the sample dataset, almost one third of the pages were near duplicates of other pages.

- **Method of descriptive Words:** Ilyinsky et al. [85] suggested a method of descriptive words for detection of near-duplicates. It is based on the choice of \( N \) words from the index to determine a signature of the document. Any search engine based on the inverted index can apply this method. The authors compared their proposed method with shingling method and concluded that the proposed method was efficient in the presence of inverted index.

- **Fingerprinting Technique:** Manku et al. [86] developed a near-duplicate detection system intended for a multi-billion page repository. They presented an algorithmic technique to identify the existing \( f \)-bit fingerprints that vary from a given fingerprint in at most \( k \) bit positions, provided that value of \( k \) is small. Both online queries (single fingerprints) and batch queries (multiple fingerprints) are aided by this technique. The expediency of their design is confirmed by the experimental evaluation over real data. A. K. Sharma et al. [87] described a novel scheme to reduce the duplication of the documents devised wherein a 64 bit document fingerprints for a static document is generated. They claimed that if two fingerprints were different then the corresponding two documents were definitely different.

- **Prefix Filter technique:** A multi-level prefix-filter, which reduces the number of similarity calculations more efficiently and maintains the advantage of the current prefix-filter by applying multiple different prefix-filters, was proposed by Tateishi and Kusui [88]. The results of their experiments illustrate that when compared with
the current prefix-filter, the presented method reduces the number of similarity calculations to \(\frac{1}{4}\).

- **Copy detection Approach:** An approach that performs copy detection on web documents was presented by Yerra and Yiu Kai Ng [89]. Their approach determines the similar web documents, similar sentences and graphically captures the similar sentences in any two web documents. Besides handling wide range of documents, the approach is applicable to web documents in different subject areas as it does not require static word lists.

- **Dust Buster Method:** A novel algorithm, *Dust Buster*, for uncovering DUST (Different URLs with Similar Text) was presented by Bar Yossef et al. [90]. They intended to discover rules that transform a given URL to others that are likely to have similar content. Dust Buster employs previous crawl logs or web server logs instead of probing the page contents to mine the dust efficiently. Search engines can increase the effectiveness of crawling, reduce indexing overhead, and improve the quality of page ranks by employing this technique.

Because of dynamic nature of WWW and its vast number of documents, there is a need for new solutions in the area of WCM that are not depending on accessing the complete data on the outset. Another important aspect is the presentation of query results. A web query can retrieve thousands of resulting web pages. Thus, meaningful methods for presenting these large results are necessary so as to help user select the most interesting content.

The next category of web mining i.e. Web Structure Mining is discussed in the subsequent subsection.

### 3.2.2 WEB STRUCTURE MINING (WSM)

The goal of *Web Structure Mining (WSM)* [15] is to generate structural summary about the website and web pages. WSM tries to discover the link structure of the hyper-links at the inner document Level. Based on the topology of the hyperlinks as shown in Fig. 3.3. WSM categorizes the web pages and generates the information such as similarity and relationship between different websites and web pages. WSM can also be employed in another direction...
to discover the structure of web document itself. This type of structure mining can be used to reveal the structure (schema) of web pages and thereby facilitate introducing database techniques for accessing information in web pages by providing a reference schema.

![Graph Model of WWW](image)

**Fig. 3.3 Graph Model of WWW**

Much of the research in WSM area considers the *Bow-Tie structure* of the web graph, which is illustrated in the next subsection.

### 3.2.2.1 Bow-tie Structure of the Web Graph

A special property of the Web graph is its bow-tie structure. The authors in [91] conducted a series of experiments including analyses on Weakly Connected Components (WCC), Strongly Connected Components (SCC), and random start Breadth First Search (BFS) using large-scale Web graphs induced from two AltaVista crawls, each with over 200 million nodes (pages) and 1.5 billion links. Based on the results of these experiments, the authors in [91] also inferred and depicted macroscopic structure of the Web graph as a bow-tie structure. Fig. 3.4 shows an illustration of the bow-tie structure, which consists of:

- **SCC** is the core of the structure, corresponding to the largest strongly connected component whose pages can reach each other via directed paths. Most of the well known websites are in this core component.

- **IN** consists of upstream pages that can reach the SCC via a directed path of hyperlinks, but cannot be reached from the SCC in a similar way.

- **OUT** consists of downstream pages that can be reached from the SCC via a directed path of hyperlinks, but it cannot reach the SCC in a similar way.
- TENDRILS consists of pages that can only be reached from the IN component (TENDRILIN) and those that can only reach to the OUT component (TENDRIL-OUT).
- DISCONNECTED consists of pages outside the largest weakly connected component (WCC) in the Web graph.
- TUBES are directed paths from a page in the TENDRIL-IN component to a page in the OUT component.

![Fig. 3.4 Bow-Tie Structure of Web Graph](image)

As there is a disconnected component in the bow-tie structure, it is clear that there are sizable portions of the Web that cannot be reached from other portions.

The structural information generated from the web graph [43] may include the following:

- The information measuring the frequency of the local links in the web pages.
- The information measuring the frequency of Web pages containing links that are interior i.e. the links that are within the same document.
- The information measuring the frequency of Web pages that contains links that are global and the links that span different Web sites.
- The information measuring the frequency of identical Web pages.
3.2.2.2 WSM Research

WSM has been largely influenced by research in social network analysis and citation analysis among Web pages. Usually, the larger the number of inlinks, the more useful a page is considered to be. The rationale is that a page referenced by many people is likely to be more important than a page that is seldom referenced. In addition, it is reasonable to give a link from an authoritative source (such as Yahoo!) a higher weight than a link from an unimportant personal home page. By analyzing the pages containing a URL, one can also obtain the anchor text that describes it. Anchor text shows how other Web page authors annotate a page, which can be useful in predicting the content of the target page. Several algorithms have been developed in the area of WSM. Among various WSM algorithms, PageRank [24] and HITS [26] are the two most widely used page ranking algorithms and have been discussed in the previous chapter. Besides page ranking, other research directions include Web community discovery, Web page classification and Clustering. Below are introduced two specific types of citation analysis [92] related to most of the WSM algorithms: co-citation and bibliographic coupling.

1. Co-citation

Co-citation is used to measure the similarity of two documents based on their link information. If page $i$ and $j$ are cited together by many pages, it means that $i$ and $j$ have a strong relationship or similarity. The more pages they cite together, the stronger their relationship is. Fig. 3.5(a) below shows this idea behind it.

![Fig. 3.5 (a) Page $i$ and $j$ co-cited by $k$](image)

![Fig. 3.5 (a) Page $i$ and $j$ co-cited by $k$](image)

![Fig. 3.5 (b) Page $i$ and $j$ cite page $k$](image)
Let $L$ be the citation matrix. Each cell of the matrix is defined as follows: $L_{i,j} = 1$ if page $i$ cites page $j$, and 0 otherwise. Co-citation (denoted by $C_{i,j}$) is defined as the number of pages that co-cite $i$ and $j$, and is computed as given below:

$$
C_{i,j} = \sum_{k=1}^{n} L_{i,k} L_{j,k} \left( L^T L \right)_{ii}
$$

where, $n$ is the total number of pages. A square matrix $C$ can be formed with $C_{i,j}$, and it is called the co-citation matrix.

2. Bibliographic Coupling

Bibliographic coupling is also a similarity measure. It can be looked upon as the mirror image of co-citation. The main idea here is that if page $i$ and $j$ both cite page $k$, they may be said to be related, even though they do not cite each other directly as shown in Fig. 3.5(b). Let $B_{i,j}$ denotes the number of pages that are cited by both pages $i$ and $j$:

$$
B_{i,j} = \sum_{k=1}^{n} L_{i,k} L_{j,k} \left( L^T L \right)_{ii}
$$

A square matrix $B$ can be formed with $B_{i,j}$, and it is called the bibliographic coupling matrix.

The next section describes another approach to Web mining called Web Usage Mining.

3.2.3 WEB USAGE MINING (WUM)

Web Usage Mining (WUM) [17, 93] is a technique of discovering usage patterns from web data in order to understand and better serve needs of web based applications. WUM has seen a rapid increase in interest, from both the research and practice communities. Web server logs, proxies and client applications can quite easily capture data about web usage. Web server logs contain information about every visit to the pages hosted on a server as described in the previous chapter. Some of the useful information includes what files have been requested from the server, when they were requested, the Internet Protocol (IP) address of the request, the error code, the number of bytes sent to the user, and the type of browser used. Web servers can also capture referrer logs giving information about page from which a visitor makes the next request. Client-side applications, such as Web browsers or personal agents, can also be designed to monitor and record a user’s actions.
By performing analysis on Web usage data, Web mining systems can discover useful knowledge about a system's usage characteristics and the users' interests. This knowledge has various applications such as personalization and collaboration in Web-based systems, marketing, Web site design, Web site evaluation, and decision support etc.

The framework of WUM shown in Fig. 3.6 consists of three phases namely:

- **Preprocessing**
- **Pattern discovery**
- **Pattern analysis**

The first is preprocessing phase which involves data cleansing and identification of user sessions from log data. The second phase searches for patterns in the data by making use of standard data mining techniques, such as association rules or mining for sequential patterns. In the third phase, an information filter based on domain knowledge and the web site structures is applied to the mined patterns so as to search for the interesting patterns. Links between pages and the similarity between contents of pages provide evidence that pages are related. The preprocessing phase allows the option of converting the server sessions into episodes prior to performing knowledge discovery.

![Fig. 3.6 Web Usage Mining Process](image)
Cooley et al. [12, 94] proposes that the WUM process can be divided into two main parts as shown in the architecture outlined in Fig. 3.7. The first part includes the domain dependent processes of transforming the Web data into suitable transaction form. This includes preprocessing, transaction identification, and data integration components. The second part includes some data mining and pattern matching techniques such as association rule and sequential pattern discovery.

![Fig. 3.7 General Architecture of Web Usage Mining](image)

The functioning of three phases of WUM and their subtasks [95] are briefly explained below:

### 3.2.3.1 Data Preprocessing

It is necessary to perform a data preparation to convert the raw data for further processing. This task can be applied to either content of web pages or their structure or to the server logs as given below:

- **Content Preprocessing**: Content preprocessing is the process of converting text, image, scripts and other files into the forms that can be used by the usage mining.
- **Structure Preprocessing:** The structure of a web site is formed by the means of hyperlinks between page views. The structure preprocessing can be treated similar to content preprocessing. However, each server session may have to construct a different site structure than others.

- **Usage Preprocessing:** As described before, the inputs of this preprocessing phase may include the web server logs, proxy logs, referral logs, registration files, index server logs, and optionally usage statistics from a previous analysis. The outputs are the user session file, transaction file, site topology, and page classifications. A user session is defined as a sequence of requests made by a single user over a certain navigation period. Session identification is the process of segmenting the access log of each user into individual access sessions, which is not a simple task due to proxy servers, dynamic addresses, and cases where multiple users access the same computer or one user uses multiple browsers or computers. Two time oriented heuristic methods: session-duration based method and page-stay-time based method have been specifically proposed in the literature [95] for session identification.

An example session file is illustrated in Table 3.2.

<table>
<thead>
<tr>
<th>SessionID</th>
<th>IP Address</th>
<th>Date &amp; Time</th>
<th>URL Accessed</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>128.102.204.20</td>
<td>2006-07-23 21:10:30</td>
<td>/shuttle/countdown/countdown.html</td>
</tr>
<tr>
<td>...</td>
<td>..................</td>
<td>..................</td>
<td>....................................</td>
</tr>
</tbody>
</table>

The preprocessed data can also be represented in the form of a session-pageview matrix [12] as shown in Fig. 3.8, where two sets of Web objects are constructed: web session set $S$ and web page set $P$. Furthermore, each user session is considered as a sequence of page-weight pairs. The various components in Web usage data model of session-pageview matrix are given below:

- $S = \{s_i \mid i = 1, \ldots, m\}$: a set of $m$ user sessions.
- \( P = \{p_j, j = 1, \ldots, n\} \): a set of \( n \) Web pageviews.
- For each user, a user session is represented as a vector of visited pageviews with corresponding weights: \( s_i \) \( \{a_{ij}, j = 1, \ldots, n\} \), where \( a_{ij} \) denotes the weight for page \( p_j \) visited in \( s_i \) user session. The corresponding weight is usually determined by the number of hit or the amount time spent on the specific page.
- \( SP_{m \times n} = \{a_{ij}, i = 1, \ldots, m, j = 1, \ldots, n\} \): the ultimate usage data in the form of a weight matrix with a dimensionality of \( m \times n \).

![Fig. 3.8 Schematic Structure of Session-Pageview Matrix](image)

### 3.2.3.2 Pattern Discovery

This is the key component of the usage mining process. Pattern discovery encompasses the algorithms and techniques related to several research areas, such as data mining, machine learning, statistics and pattern recognition. Some of the techniques are briefly outlined below:

- **Statistical Analysis**: Statistical analysts may perform different kinds of descriptive statistical analyses based on different variables when analyzing the session files. By analyzing the statistical information contained in the periodic web system report, the extracted information can be potentially useful for improving the system performance, enhancing the security of the system, facilitating the site modification task and providing support for marketing decisions.

- **Association Rules**: In the web domain, the pages, which are most often referenced together, can be put in one single server session by applying the association rule
generation. Association rule mining techniques can be used to discover unordered correlation between items found in a database of transactions.

- **Clustering:** Cluster analysis is a technique of grouping together users or data items (web pages) with similar characteristics. Clustering can facilitate the development and execution of future decision strategies.

- **Classification:** Classification is a technique to map a data item into one of the several predefined classes. The classification can be done by using supervised inductive learning algorithms such as decision tree classifiers, naïve Bayesian classifiers, k-nearest neighbor classifier, Support Vector Machines etc.

- **Sequential Pattern Detection:** This technique intends to find the inter-session pattern, such that a set of the items follows the presence of another in a time-ordered set of sessions or episodes. Sequential patterns also include some other types of temporal analysis such as trend analysis, change point detection, or similarity analysis.

- **Dependency Modeling:** The goal of this technique is to establish a model that is able to represent significant dependencies among the various variables in the web domain. The modeling technique provides a theoretical framework for analyzing the behavior of users and is potentially useful for predicting future web resource consumption.

### 3.2.3.3 Pattern Analysis

Pattern Analysis is the final stage of the process of web usage mining with the goal to eliminate the irrelative rules or patterns and to extract the interesting rules or patterns from the output of the pattern discovery process. The output of web mining algorithms is often not in the form suitable for direct human consumption, and thus need to be transformed to a format which can be assimilated easily. There are two most common approaches for the pattern analysis: one is to use the knowledge query mechanism such as SQL, while another is to construct multi-dimensional data cube before performing Online Analytical Processing (OLAP) operations.

The next section describes the prevalent data mining techniques in the context of web mining techniques.
3.3 WEB MINING TECHNIQUES

In the context of web mining, there are lot of algorithms and approaches reported in the available literature. These approaches and techniques are well studied and implemented for different applications and scenarios by researchers. In the next few subsections, some prevalent web mining techniques have been described.

3.3.1 CLUSTER ANALYSIS

The process of grouping a set of physical or abstract objects into classes of similar objects is called “cluster analysis” or “clustering” [6, 84, 96]. It is an unsupervised learning technique and has been widely used in numerous applications including market research, data analysis and image processing. It also plays a major role in the area of Web mining Applications, wherein it can be employed for grouping similar web documents, queries, users and their sessions in potential clusters.

The following are the typical requirements of clustering in web mining.

- **Scalability:** Many Clustering Algorithms works well on small data sets containing fewer than several hundred documents; however, a large repository may contain millions of web pages. Therefore, highly scalable algorithms are needed.
- **Ability to deal with different types of attributes:** Many algorithms are designed to cluster text based web pages. However, applications may require clustering other type of data also, such as images.
- **Discovery of clusters with arbitrary shape:** Many clustering algorithms work on the basis of Euclidean or Manhattan distance measures, therefore, leading to clusters with similar sizes and density. However, a cluster could be of any size and density. There is a need of algorithms that can detect clusters of arbitrary shapes and sizes.
- **Incremental Clustering:** Some of the existing clustering algorithms cannot incorporate updated data into existing clusters. Therefore, it is important to develop incremental clustering algorithms.

3.3.1.1 Major Categories of Clustering

In general, the major clustering methods can be classified into the following categories.
1. **Partitioning methods**: Given a database of \( n \) objects, a partitioning method constructs \( K \) \((K \leq n)\) partitions of the data, where each partition represents a cluster. The clusters satisfy the following requirements:

- Each group must contain at least one object, and
- Each object must belong to exactly one group.

The general criterion of a good partitioning is that objects in the same cluster are "closed" or related to each other, whereas objects of different clusters are "far apart" or very different. *K-means, K-medoids* [13, 120] are few popular algorithms based on partitioning method. The *K-means* algorithm is given in Fig. 3.9.

**Algorithm: K-means()**

Input: A dataset \( D \), a user specified number \( k \)
Output: \( k \) clusters

```
{  
  Randomly Initialize cluster centroids;
  While not convergent
  {  
    For each object \( o \) in \( D \) do 
      Find the cluster \( c \) whose centroid is most close to \( o \); 
      Allocate \( o \) to \( c \);  
    For each cluster \( c \) do 
      Recalculate the centroid of \( c \) based on the objects allocated to \( c \); 
  }
}
```

**Fig. 3.9 The K-means Algorithm**

The key idea of *K-means* is simple and is as follows: In the beginning, the number of clusters \( K \) is determined. Then the algorithm randomly assumes the centroids (or centers) of these \( K \) clusters. If the number of objects is less than the number of clusters, then each object is treated as the centroid of a cluster and allocated a cluster number. Otherwise, the algorithm computes the distance (i.e., Euclidean distance) between each object and all centroids to get the minimum distance. Because the location of the real centroid is unknown during the process, the algorithm needs to revise the centroid location with regard to the updated information. After updating the values of the centroids, all the objects are reallocated to the \( K \) clusters. The process is repeated until the assignment of objects to clusters ceases to change, or when the centroids move by negligible distances in successive iterations.
2. **Hierarchical methods:** Hierarchical clustering [97] constructs a hierarchy of clusters that can be illustrated in a tree structure which is also known as a *dendrogram* as shown in Fig. 3.10. Each node of the dendrogram, including the root, represents a cluster and the parent-child relationship among them enables to explore different levels of clustering granularity. There are mainly two types of algorithms for hierarchical clustering:

- Agglomerative
- Divisive

![Fig. 3.10 An Example Dendogram with 10 Objects](image)

The *agglomerative approach*, also called the bottom up approach, starts with each object forming a separate group. It successively merges the objects or groups that are closed to one another, until all of the groups are merged into one, or until a termination condition holds. The Hierarchical Agglomerative Clustering (HAC) algorithm is presented in Fig. 3.11. The *divisive approach*, also called the top-down approach, starts with all of the objects in the same cluster. In successive iterations, a cluster is split up into smaller clusters, until eventually each object is in one cluster, or until a termination condition holds.

3. **Density based Methods:** The partitioning and hierarchical methods can find only spherical-shaped clusters and encounter difficulty at discovering clusters of arbitrary shapes and sizes. Other clustering methods have been developed [13, 121] based on notion of density. wherein if a number of data objects in the “neighborhood” exceeds some threshold,
then they are grouped together. It means, for each data point within a given cluster, the neighborhood of a given radius has to contain at least a minimum number of points. The \textit{DBSCAN} algorithm and its extension \textit{OPTICS} are typical density-based methods \cite{13} that perform clustering according to a density-based connectivity analysis.

\begin{algorithm}
\textbf{Algorithm: HAC()}
\begin{itemize}
\item Input: A dataset $D$
\item Output: A hierarchy tree of clusters
\item \{ \\
\hspace{0.5cm} Allocate each object $o$ in $D$ as a single cluster:
\hspace{0.5cm} Let $C$ be the set of the clusters:
\hspace{0.5cm} While $|C| > 1$ do
\hspace{1cm} For all clusters $X, Y \in C$ do
\hspace{1.5cm} Compute the between-cluster similarity $S(X, Y)$;
\hspace{1.5cm} $Z = X \cup Y$, where $S(X, Y)$ is the minimum;
\hspace{1.5cm} Remove $X$ and $Y$ from $C$;
\hspace{1.5cm} $C = C \cup Z$;
\hspace{0.5cm} \}
\end{itemize}
\end{algorithm}

\textbf{Fig. 3.11 The Hierarchical Agglomerative Clustering (HAC) Algorithm}

The next subsections illustrate different distance or similarity measures involved in clustering query log datasets, clustering different users and clustering web documents.

\subsection*{3.3.1.2 Clustering query Log data set}

Query mining \cite{98, 99} is a sub-area of WUM and is used to determine the underlying structure and relationships in large amounts of queries stored in search engine's query logs. It can be used to discover common interests of online information seekers and to exploit their collective search experience for the benefit of others. It aims at grouping semantically related queries which were accumulated with the interactions between users and the search engine.

The key problem underlying query clustering is to determine an adequate similarity function so that truly similar queries can be grouped together using a clustering algorithm. In this section, some similarity functions have been revised, which are used for clustering queries. There are different ways to represent query contents: keywords, words in their order, and phrases. They provide different measures of similarity, each with its own useful information.
• **Similarity based on keywords or phrases**

This measure directly comes from IR studies. Keywords are the words, except for function words included in a stop-list. The keyword-based similarity function is defined as follows:

\[
similarity_{\text{keyword}}(p, q) = \frac{KN(p, q)}{\max(kn(p), kn(q))}
\]  

(3.7)

where \( kn(.) \) is the number of keywords in a query, \( KN(p, q) \) is the number of common keywords in two queries. If query terms are weighted, the following modified formula can be used instead:

\[
similarity_{w\text{-keyword}}(p, q) = \frac{\sum_{i=1}^{N} w(k_i(p)) \cdot w(k_i(p))}{\max(kn(p), kn(q))}
\]  

(3.8)

where \( w(k_i(p)) \) is the weight of the \( i \)th common keyword in query \( p \) and \( kn(.) \) becomes the sum of weights of the keywords in a query.

The above measures can be easily extended to phrases. Phrases are a more precise representation of meaning than single words. For example, the two queries “history of China” and “history of the United States” are very close queries (both asking about the history of a country). Their similarity is 0.33 on the basis of keywords. If we can recognize “the United States” as a phrase and take it as a single term, the similarity between these two queries is increased to 0.5.

• **Similarity based on string matching**

This measure uses all the words in the queries for similarity estimation including the stop-words. Comparison between queries becomes an inexact string-matching problem as formulated [98]. Similarity may be determined by the edit distance, which is a measure based on the number of edit operations (insertion, deletion, etc.) necessary to unify two strings (queries). The similarity is inversely proportional to edit distance:

\[
similarity_{\text{edit}}(p, q) = 1 - \text{edit distance}(p, q)
\]  

(3.9)

The advantage of this measure is that it takes into account the word order, as well as words that denote query types such as "who" and "what" if they appear in a query.
**• Similarity based on user feedback**

Let $D(q_j)$ be the set of documents the system presents to the user as search results. The document set $D_C(q_j)$ and $D_C(q_j)$, on which the users clicked on for queries $q_j$ and $q$, may be seen as follows:

$$D_C(q_j) = \{d_{c_{j1}}, d_{c_{j2}}, \ldots, d_{c_{jn}}\} \subseteq D(q_j)$$

(3.10)

$$D_C(q_j) = \{d_{c_{j1}}, d_{c_{j2}}, \ldots, d_{c_{jn}}\} \subseteq D(q_j)$$

(3.11)

Similarity based on user clicks follows the following principle:

If $D_C(q_j) \cap D_C(q_j) \neq \emptyset$, then documents $d_{c_{j1}}, d_{c_{j2}}, \ldots, d_{c_{jn}}$ represent the common topics of queries $q_j$ and $q$. Therefore, a similarity between queries $q_j$ and $q$ is determined by $D_C(q_j) \cap D_C(q_j)$.

**• Similarity through clicked documents**

This feedback-based similarity measure considers each document in isolation. This similarity is proportional to the shared number of clicked (or selected) documents taken individually [99], as follows:

$$similarity_{document}(p, q) = \frac{RD(p, q)}{Max(rd(p), rd(q))}$$

(3.12)

where $rd(\cdot)$ is the number of clicked documents for a query. $RD(p, q)$ is the number of document clicks in common. This measure is very useful in distinguishing between queries that happen to be worded similarly but stem from different information needs.

**• Combination of multiple measures**

Similarities based on query contents and user clicks represent two different points of view. In general, content-based measures tend to cluster queries with the same or similar terms. Feedback-based measures tend to cluster queries related to the same or similar topics. Since user information needs may be partially captured by both query texts and relevant documents, a combined measure can be defined that takes advantage of both strategies. A simple way to do this is to combine both measures linearly, as follows:

$$similarity_{comp} = \alpha \times similarity_{content} + \beta \times similarity_{feedback}$$

(3.13)
where $\alpha$ and $\beta$ are constants with their values between 0 and 1.

### 3.3.1.3 Document Clustering

In the context of web document clustering [100, 101, 102], objects are replaced by web documents and are grouped together based upon some measure like similarity of content or of hyperlinked structure. As discussed earlier, most of the search engines return a large and unmanageable list of documents containing the user specified query keywords. Finding the user required documents from such a large list is usually tedious, often impossible. As a solution, the search engines could apply web mining tools to group a set of documents returned in response to a query with the aim of finding semantically meaningful clusters, rather than a list of ranked documents. Web page clustering may be based on content alone, may be based on both content and links or may only be based on links.

Popular clustering techniques like k-means and the hierarchical clustering [97] can be used for Web document cluster analysis, but these algorithms assume that each document has a fixed set of attributes that must appear in all documents. Similarity between documents can then be computed based on these attribute values. One approach for Web document cluster analysis: Suffix Tree Clustering (STC) [100] uses a phrase-based clustering approach rather than using single word frequency.

Vector space model is generally employed in such type of clustering, where similarity between Web pages usually means content-based similarity [103]. It is also possible to consider link-based similarity and usage-based similarity. Link-based similarity [7] is related to the concept of co-citation and is primarily used for discovering a core set of web pages on a topic. Usage-based similarity is useful in grouping pages or users into meaningful groups. In the proposed work, focus is on content-based similarity which is based on comparing the textual content of the web pages. Following is given a similarity measure [15] to find out the similarity between two documents:

$$SS(D_i, D_j) = \frac{\sum_{k=1}^{n} KW_{k,i} \cdot KW_{k,j}}{\text{length of } doci \cdot \text{length of docj}}$$

(3.14)
where $T_i$ and $T_j$ represent the tables containing tokens of documents $D_i$ and $D_j$, respectively. The numerator gives the summation of products of term frequencies of common tokens/keywords (which is $n$ in number) in $T_i$ and $T_j$. The length of a document $i$ can be calculated by (3.15)

$$length \text{ of } Doc_i = \sqrt{\sum_{k=1}^{n} w_{i,k}^2}$$  \hspace{1cm} (3.15)

where $w_{i,k}$ represent the $k^{th}$ tokens in document $i$. The length is calculated by summation of products of term frequencies of tokens (which is $n$ in number) in a document.

3.3.1.4 Clustering Users and Sessions

Capturing the characteristics of web users is an important task for the web site designers. By mining users’ historical access patterns, not only the information about how the Web is being used, but also some demographics and behavioral characteristics of users could be determined. The navigation path of the users carries valuable information about the user interests. The purpose of finding similar interests among the users is to discover knowledge from the user profile. If a web site is well designed, there will be strong correlation among the similarity of the navigation paths and similarity among the user interests. Therefore, clustering of the former could be used to cluster the latter [104].

Suppose, for a given web site $S$, there are $m$ users $U = \{u_1, u_2, \ldots, u_m\}$ who accessed $n$ different pages $P = \{p_1, p_2, \ldots, p_n\}$ in some time interval. For each page $p_i$ and each user $u_j$, it is associated with a usage value, denoted as $use(p_i, u_j)$, and defined as:

$$use(p_i, u_j) = \begin{cases} 1 & \text{if } p_i \text{ is accessed by } u_j \\ 0 & \text{otherwise} \end{cases}$$  \hspace{1cm} (3.16)

The $use$ vector can be obtained by retrieving the access logs of the site. If two users accessed the same pages, they might have some similar interests in the sense that they are interested in the same information (e.g., news, electrical products etc). The number of common pages they accessed can measure this similarity. The measure is defined by:

$$Sim(u_i, u_j) = \frac{\sum_{k=1}^{n} (use(p_k, u_i) * use(p_k, u_j))}{\sqrt{\sum_{k=1}^{n} use(p_k, u_i) * \sum_{j=1}^{n} use(p_k, u_j)}}$$  \hspace{1cm} (3.17)
where $\sum_{use(pk,ui)}$ is the total number of pages that were accessed by user $ui$, and the product of $\sum_{use(pk,ui)} * \sum_{use(pk,uj)}$ is the number of common pages accessed by both user $ui$ and $uj$. If two users access the exact same pages, their similarity will be 1. The similarity measure defined in this way is called *Usage Based (UB)* measure.

Generally, the similarity between two users can be measured by counting the number of times they access the common pages at all sites. In this case, the measure is defined by:

$$Sim(ui,uj) = \frac{\sum_{i} (acc,(pk,ui) * acc,(pk,uj))}{\sqrt{\sum_{i} use(pk,ui) * \sum_{i} use(pk,uj)}}$$

where $acc,(pk,ui)$ is the total number of times that user $ui$ accesses the page $pk$ at site $S$. This measure is called *Frequency Based (FB)* measure.

The similarity between two users can be measured more precisely by taking into account the actual time the users spent on viewing each Web page. Let $t(pk,uj)$ be the time the user $uj$ spent on viewing page $pk$ (assume that $t(pk,uj) = 0$ if $uj$ did not access page $pk$). In this case, the similarity between users can be expressed by:

$$Sim(ui,uj) = \frac{\sum_{i} (t(pk,ui) * t(pk,uj))}{\sqrt{\sum_{i} (t(pk,ui))^2 * \sum_{i} (t(pk,uj))^2}}$$

where $\sum_{i} (t(pk,ui))^2$ is the square sum of the time user $ui$ spent on viewing pages at the site, and $\sum_{i} (t(pk,ui) * t(pk,uj))$ is the inner-product over time spent on viewing the common pages by users $ui$ and $uj$. Even if two users access exact the same pages, their similarity might be less than 1 in this case, if they view a page in different amount of time. This measure is called *Viewing-Time Based (VTB)* measure.

The next section is devoted to the discussion of another technique of web mining called Sequential Pattern Detection.

### 3.3.2 ASSOCIATION RULE MINING

The most important and basic principle in data mining is association rule mining. The purpose of finding association rules is to analyze the co-existence relations between items (or
The issue has attracted a great deal of interest during the recent surge in web mining research because it is the basis of many applications, such as web user behavior analysis, usage trend prediction, finding relations between different pages and queries, which can further be utilized to build effective recommendation systems.

### 3.3.2.1 Association Rule Mining Problem

The problem of association rule discovery can be stated as follows [105]:

Let \( I = \{i_1, i_2, \ldots, i_n\} \) be a set of items. A subset of \( I \) is called an itemset. The itemset, \( t_j \), is denoted as \( \langle x_1, x_2, \ldots, x_m \rangle \), where \( x_k \) is an item, i.e., \( x_k \in I \) for \( 1 \leq k \leq m \). The support of an itemset \( X \), denoted as \( \text{support}(X) \), is the number of transactions in which it occurs as a subset. A \( 'k' \) length subset of an itemset is called a \( k \)-subset.

An itemset is frequent if its support is greater than a user-specified minimum support (say \( \text{minsup} \)) value. The set of frequent \( k \)-itemsets is denoted as \( F_k \). An Association Rule is an expression \( A \Rightarrow B \), where \( A \) and \( B \) are itemsets. The support of the rule is given as:

\[
\text{Support} (A \Rightarrow B) = \frac{\text{Support}(A \cup B)}{\text{Support}(A)}
\]

The confidence of the rule \( A \Rightarrow B \) is calculated as the conditional probability that a transaction contains \( B \), given that it contains \( A \) also as given below:

\[
\text{conf}(A \Rightarrow B) = \frac{\text{support}(A \cup B)}{\text{support}(A)}
\]

A rule is confident if its confidence is greater than a user-specified minimum confidence (\( \text{minconf} \)). The associate rule mining task is to generate all the rules, whose supports are greater than \( \text{minsup} \), and the confidences of the rules are greater than \( \text{minconf} \). Below is specified the most popular \textit{Apriori} algorithm to find association rules.

### 3.3.2.2 Apriori: An Association Rule Mining Algorithm

Agrawal et al. [106] proposed \textit{Apriori} algorithm to address the association rule mining issue. It applies a two-stage approach to discover frequent itemsets and confident association rules. The two stages are:

1. **Frequent Itemset Discovery.** To find all frequent itemsets, the basic idea is to first generate the candidate \( k \)-itemsets (\( k \) is 1 at the beginning), then these candidates will
be evaluated whether frequent or not. To discover the frequent 2-itemsets, Apriori generates candidate 2-itemsets by joining 1-itemsets. In a similar way, all frequent (k+1)-itemsets can be found based on already known frequent k-itemsets. The frequent itemset mining of the Apriori algorithm is presented in Fig. 3.12.

2. **Association Rule Mining.** Given all frequent itemsets, for each itemset \(I\), all its subsets \(S\) are extracted and candidate rules of the form \(S \rightarrow (l-S)\) are generated. The candidate rules which qualify the \(\text{minconf}\) measure (calculated by (3.21)) are considered confident association rules.

![Fig. 3.12 The Apriori Algorithm](image)

### Algorithm: Apriori - Frequent Itemset Mining()

**Input:** A transaction database \(D\), a user specified threshold \(\text{minsup}\)

**Output:** Frequent itemsets \(F\)

```plaintext
C_1 = \{1-itemsets\};
\(k=1\):
While (\(C_k \neq \text{NULL}\)) do // Test candidate itemsets
\{ For transaction \(T \in D\) do
\{ For candidate itemsets \(A \in C_k\) do
  If \(A \subseteq T\) then \(X\.\text{support}++\);
\}
  \(F_k = F_k \cup X\), where \(X\.\text{support} \geq \text{minsup}\);
// Generate candidate itemsets
  For all \(|i_1|, \ldots, i_l, i_{l+1}, \ldots, i_k, i_{k+1}| \in F_k\) such that \(i_k < i_{k+1}\) do
    \{ c = \{i_1, \ldots, i_l, i_{l+1}, \ldots, i_k, i_{k+1}\};
    If all \(k\)-subsets of \(c\) are frequent then
      \(C_{k+1} = C_{k+1} \cup c\);
    \}
  \}
\}
```

3.3.3 **SEQUENTIAL PATTERN DETECTION**

Sequential Pattern Mining is the task of mining frequently occurring ordered events or subsequences as patterns. The sequential pattern mining problem was first introduced by Agrawal and Srikant in 1995 [107] based on their study of customer purchase sequences, as follows:

"Given a set of sequences, where each sequence consists of a list of events (or sessions) and each event consists of a set of items (accessed pages), and given a user-specified
minimum support threshold minsup, sequential pattern mining finds all frequent subsequences i.e. all subsequences whose occurrence frequency in the set of sequences is no less than minsup”.

This technique generally refers to WUM and thus is applied to server log’s data. Table 3.3 shows a sample transactional web database, on which sequential pattern discovery can be applied.

<table>
<thead>
<tr>
<th>User Id</th>
<th>Page Views</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>&lt; ac(bc)d(abc)ad &gt;</td>
</tr>
<tr>
<td>200</td>
<td>&lt; b(cd)ac(bd) &gt;</td>
</tr>
<tr>
<td>300</td>
<td>&lt; d(bc)(ac)(cd) &gt;</td>
</tr>
</tbody>
</table>

Following are given two prevalent algorithms for handling such type of problem.

3.3.3.1 AprioriALL.

Sequential pattern mining was first introduced by Agrawal in [107] where Apriori based mining was proposed through three algorithms but AprioriALL is the most important out of them. In this algorithm, the candidate generation process is similar to the AprioriGen in [106]. The Apriori property prunes those candidate sequences whose subsequence is not frequent. The difference is that when generating the candidates by joining the frequent patterns in the previous pass, different orders of combination make different candidates. For example: from the items, a and b, three candidates <ab>, <ba> and <(ab)> can be generated. But in association rule mining only <(ab)> is generated. The reason being that in association rule mining, the time order is not taken into account. Obviously the number of candidate sequences in sequential pattern mining is much larger than the size of the candidate itemsets in association rule mining during the generation of candidate sequences. An example of generating candidate 5-sequences by joining large 4-sequences is: Suppose <(bc)ad> and <(bc)(ac)> are frequent 4-sequences, as they share their first three items, according to the join condition of Apriori they are joined to produce the candidate sequence <(bc)(ac)d>.
The check process is simple and straightforward. It scans the database and counts the supports of candidate sequences to find the frequent sequential patterns. AprioriAll is the first algorithm on mining sequential patterns and its core idea has been commonly applied by many later algorithms. The disadvantages of AprioriAll are that too many candidates are generated and multiple passes over the database are necessary and thus, the cost of computation is high.

3.3.3.2 Generalized Sequential Patterns (GSP)

Srikant and Agrawal [108] generalized the definition of sequential pattern mining problem by incorporating some new properties, i.e., time constraints, transaction relaxation, and taxonomy. For the time constraints, the maximum gap and the minimal gap are defined to specify the gap between any two adjacent transactions in the sequence. When testing a candidate, if any gap of the candidate falls out of the range between the maximum gap and the minimal gap, then the candidate is not a pattern. Furthermore, the authors relaxed the definition of transaction by using a sliding window i.e. when the time range between two items is smaller than the sliding window, then two items are considered to be in the same transaction.

In [108], the authors proposed a new algorithm which is named GSP that efficiently finds the generalized sequential patterns. Similar to the AprioriAll algorithm, there are two steps in GSP, i.e., candidate generation and test.

The various steps need to perform GSP algorithm are as follows:

**Step 1**: Make first pass over the database D to yield all 1-length frequent sequences.

**Step 2**: Repeat until no new frequent sequences are found.

- **Candidate Generation**: Merge pairs of frequent subsequences found in $(k-1)^{th}$ pass to generate candidate sequences that contain $k$ items.
- **Candidate Pruning**: Prune candidate $k$-sequences that contain infrequent $(k-1)$-subsequences.
- **Support Counting**: Make a new pass over the sequence database $D$ to find the support for these candidate sequences.
**Candidate Elimination**: Eliminate candidate \( k \)-sequences whose actual support is less than \( \text{minsup} \).

The algorithm of GSP is given in Fig. 3.13. The Candidate generation and pruning comes under *Generation Process*, while Support counting and Elimination under *Test Process*.

---

**Algorithm: GSP()**

Input: A transactional database \( D \), a user specified threshold \( \text{minsup} \)

Output: Frequent Sequences \( F \)

\[
\begin{align*}
F_1 &= \text{the set of frequent 1-sequence} \\
k &= 2; \\
\text{while } ( F_{k-1} \neq \text{Null}) \text{ do} \\
&\quad \text{Generate candidate sets } C_k \text{ (set of candidate } k \text{-sequences);} \\
&\quad \text{For (all input sequences } s \text{ in the database } D) \\
&\quad \quad \text{Increment count of all } a \text{ in } C_k \text{ if } s \text{ supports } a \\
&\quad \quad F_k = \{ a \in C_k \text{ such that its frequency exceeds the threshold} \}
&\quad k = k+1; \\
&\quad \text{Result } = \text{Set of all frequent sequences is the union of all } F_k \text{'s}
\end{align*}
\]

---

**Fig. 3.13 The GSP Algorithm**

To understand its working, consider the transactional database given in Table 3.4. The frequent length-1 sequences extracted from the database are given in Table 3.5.

<table>
<thead>
<tr>
<th>UserID</th>
<th>min_sup-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>&lt;(bd)cb(ac)&gt;</td>
</tr>
<tr>
<td>20</td>
<td>&lt;(bf)(ce)b(fd)&gt;</td>
</tr>
<tr>
<td>30</td>
<td>&lt;(ah)(bf)abf&gt;</td>
</tr>
<tr>
<td>40</td>
<td>&lt;(be)(ce)d&gt;</td>
</tr>
<tr>
<td>50</td>
<td>&lt;a(bd)bcb(ade)&gt;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 3.5: Length 1-sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Candidate</td>
</tr>
<tr>
<td>&lt;a&gt;</td>
</tr>
<tr>
<td>&lt;b&gt;</td>
</tr>
<tr>
<td>&lt;c&gt;</td>
</tr>
<tr>
<td>&lt;d&gt;</td>
</tr>
<tr>
<td>&lt;e&gt;</td>
</tr>
<tr>
<td>&lt;f&gt;</td>
</tr>
</tbody>
</table>

The following rules are followed to generate candidate sequences of length more than 1.

- **Base case** (\( k=2 \)): Merging two frequent 1-sequences \(<\{i1\}\> \text{ and } <\{i2\}> \) will produce two candidate 2-sequences: \(<\{i1\} \{i2\}\> \text{ and } <\{i1\}i2\>\).
- General case (k>2): A frequent (k-1)-sequence \( w_1 \) is merged with another frequent (k-1)-sequence \( w_2 \) to produce a candidate k-sequence if the subsequence obtained by removing the first event in \( w_1 \) is the same as the subsequence obtained by removing the last event in \( w_2 \).

- The resulting candidate after merging is given by the sequence \( w_1 \) extended with the last event of \( w_2 \). If the last two events in \( w_2 \) belong to the same element, then the last event in \( w_2 \) becomes part of the last element in \( w_1 \). Otherwise, the last event in \( w_2 \) becomes a separate element appended to the end of \( w_1 \). For example merging the sequences \( w_1=<\{1\}\{2\}3\{4\} \) and \( w_2=<\{2\}3\{4\}5\) will produce the candidate sequence \(<\{1\}\{2\}3\{4\}5\) because the last two events in \( w_2 \) (4 and 5) belong to the same element. Similarly, merging the sequences \( w_1=<\{1\}\{2\}3\{4\} \) and \( w_2=<\{2\}3\{4\}5\) will produce the candidate sequence \(<\{1\}\{2\}3\{4\}5\) because the last two events in \( w_2 \) (4 and 5) do not belong to the same element. There is no need to merge the sequences \( w_1=<\{1\}\{2\}6\{4\} \) and \( w_2=<\{1\}\{2\}\{4\}5\) to produce the candidate \(<\{1\}\{2\}6\{4\}5\) because if the latter is a viable candidate, then it can be obtained by merging \( w_1 \) with \(<\{2\}6\{4\}5\).

An example that generates the 4-length candidates from 3-length frequent sequences is shown in Fig. 3.14.

![Fig. 3.14 Candidate Generation in GSP](image)

GSP applies the Apriori property to prune the set of candidates. In the \( k \)-th pass, a sequence is considered to be candidate only if each of its length \( (k-1) \) subsequences is a sequential pattern found at the \( (k-1) \)-th pass. Thus, a pruning phase eliminates subsets of infrequent patterns. A new scan of the database collects the support for each candidate sequence and finds a new set of sequential patterns. For all patterns \( P \) in the candidate set with length \( k \), all
sessions are processed once and the count is incremented for each detected pattern in the candidate set. The algorithm terminates when no sequential pattern is found in a pass or when no candidate sequence is generated.

GSP suffers from some limitations also such as it still generates a large number of candidates e.g. 1,000 frequent length-1 sequences generate 1,499,500 length-2 candidates. some candidates generated by GSP may not appear in the database at all, resulting in wasted time and it also suffers from multiple scans of database in mining.

Advantages of GSP over AprioriALL is that GSP is 5% to 30% faster than AprioriAll. with the performance gap often increasing at low levels of minimum support [108]. Moreover, if the candidates do not fit in memory, the algorithm generates only as many candidates as will fit in memory and the data is scanned to count the support of these candidates.

In the next section, some research directions of Web Mining techniques in context of search engines are enumerated.

3.4 WEB MINING TECHNIQUES IN SEARCH ENGINES

The concept of query clustering, document clustering, user clustering, association rule mining and sequential pattern detection has been widely used by many researchers in optimizing the search and retrieval process of search engines. Some of the literature research is explained below:

a) **Building Effective Indexes**: The keyword clustering and document clustering can be utilized in building effective index structures for search engines, which in turn prompts the efficient index searching.

b) **Automatic Query Expansion**: The extracted information from query clustering can be used as source for automatic query expansion [109, 110, 111]. By clustering similar queries and then recommending these clusters to users, there becomes an opportunity for users to take advantage of previous queries and use the appropriate ones to meet his information need.
c) **Helpful in inferring Semantic concepts:** Besides learning about search engines or their users, query logs are also being used to infer semantic concepts or relations [112]. Query logs provide an excellent opportunity for gaining insight into how a search engine is used and what the users’ interests are since they form a complete record of what users searched for in a given time frame. The information contained in query logs is being used in different ways, to classify queries, to infer search intent, to facilitate personalization, to uncover different aspects of a topic, etc.

d) **Query Recommendation:** Providing related queries for search engine users can help them quickly find the desired content. Recently, some search engines started showing related search keywords in the bottom of the result page. Their main purpose is to give search engine users a comprehensive recommendation [113] when they search using a specific query. Recommending the most relevant search keywords set to users not only enhances the search engine’s hit rate, but also helps the user to find the desired information more quickly.

e) **Ranking:** The sheer amount of Web pages and the exponential growth of the Web suggest that users are becoming more and more dependent on the search engines’ ranking schemes [24, 25, 26] to discover information relevant to their needs. Typically, users expect to find such information in the top-ranked results, and more often they do not look at the document snippets except in the first few result pages and then they give up or reformulate the query. This can introduce a significant bias to their information finding process and calls for ranking schemes that take into account not only the overall page quality and relevance to the query, but also the match with the users’ real search intent when they formulate the query.

f) **Better Document Representation:** By clustering similar documents either by their content or by their usage, the results of a search query can be presented to the users in a much better way than the traditional ordered representation. By this way, user can restrict his browsing to particular clusters of his interest. Thus, Information Overkill problem can be abridged and search space can be reduced to a better scale.

g) **Web site modification/E-commerce applications:** A methodology to improve the “success” of web sites, could be based on the exploitation of navigation pattern discovery. Deciding how the pages of a site should be improved requires an
understanding of the users' navigation patterns. More recent studies [114, 115] have investigated query logs for online search engines for biomedical publications and multimedia search.

The next section provides a brief summary of the limitations found in the literature survey.

3.5 REVIEW SUMMARY

A critical look at the available literature indicates the following issues, which need to be addressed in building efficient representation, recommendation and prediction systems for search engines while utilizing Web mining techniques:

- In response to user queries, a search engine generally returns a large volume of results generally presented to the user in the form of a ranked list. To search for the desired information, user keeps on sifting between the pages and thus making extra efforts. Some more efficient representation either in the form of clusters or in the form of combination of cluster and ranked representation is actually needed so as to reduce the search space.

- Most of the search engines depict low precision. User can’t browse all the pages one by one, and most pages are irrelevant to the user’s interest, they are highlighted and returned by search engine just because these pages posses query keywords. Even, the most relevant Web pages to users’ query words are generally not shown at the top of the search results list. Hence, the time users spend for seeking out the required information from search result list is large.

- The traditional ranking methods employed by the search engines are generally based either on content-oriented or on the link-oriented approaches i.e. it assign a page score independent of users’ query words. Thus, the relation between Web pages and the requirement of a user could not be completely matched.

- In response to user queries, a search engine generally returns only those web pages whose contents are indexed by them. In some cases, there may exist pages which are linked to by pages already indexed, but are not indexed by search engines. Is it still possible to meaningfully index p and return it in search results?
Search engines generally organize their indexes in various tiers and partitions, not all of which are examined on every search. For instance, an important page deep inside a website may be indexed but not retrieved on general web searches; it is, however, retrieved as a result of a search that a user has explicitly restricted to that website. Is it possible for the search engine to find important pages in its index and present them in the results?

As indexer module of the search engine performs linguistic preprocessing on the parsed tokens, sometimes they are expected to lose considerable precision on user queries (e.g. operate, operates, operating, operation, operative, operatives, operational all get convert to ‘oper’ instead of ‘operate’ which is a common verb). The modern search engines apply complex Natural Language Processing techniques to resolve it. Could there be an alternative solution to this problem?

Although there are many efficient methods available for duplicate data detection but they are not much scalable due to the abundance of web pages on WWW. Therefore, a mechanism needs to be introduced for overcoming the problem of scalability and efficiently detect duplicates.

As most of the search engines are keyword based, semantics of keywords are generally ignored by them. For instance, the topical query “apple” given to a search engine may retrieve the pages related to “apple fruit” as well as “apple computer” together, thereby unnecessarily increasing the search space. Infact, there is a need of optimizing user search by the way of web mining techniques so as to restrict it towards the right direction either by building efficient semantic analyzers or by building efficient recommendation systems.

In the next chapter, a P & P Framework (Pre and Post web mining base framework) for optimizing the crawling, indexing and searching process of search engine is being proposed with a view to resolve most of the above said issues.