Mining Time-Series Data

"What is a time-series database?" A time-series database consists of sequences of values or events obtained over repeated measurements of time. The values are typically measured at equal time intervals (e.g., hourly, daily, weekly). Time-series databases are popular in many applications, such as stock market analysis, economic and sales forecasting, budgetary analysis, utility studies, inventory studies, yield projections, workload projections, process and quality control, observation of natural phenomena (such as atmosphere, temperature, wind, earthquake), scientific and engineering experiments, and medical treatments. A time-series database is also a sequence database. However, a sequence database is any database that consists of sequences of ordered events, with or without concrete notions of time. For example, Web page traversal sequences and customer shopping transaction sequences are sequence data, but they may not be time-series data.

With the growing deployment of a large number of sensors, telemetry devices, and other on-line data collection tools, the amount of time-series data is increasing rapidly, often in the order of gigabytes per day (such as in stock trading) or even per minute (such as from NASA space programs). How can we find correlation relationships within time-series data? How can we analyze such huge numbers of time series to find similar or regular patterns, trends, bursts (such as sudden sharp changes), and outliers, with fast or even on-line real-time response? This has become an increasingly important and challenging problem. In this section, we examine several aspects of mining time-series databases, with a focus on trend analysis and similarity search.

Trend Analysis

A time series involving a variable Y, representing, say, the daily closing price of a share in a stock market, can be viewed as a function of time t, that is, Y = F(t). Such a function can be illustrated as a time-series graph, as shown in Figure 8.4, which describes a point moving with the passage of time.

"How can we study time-series data?" In general, there are two goals in time-series analysis: (1) modeling time series (i.e., to gain insight into the mechanisms or underlying forces that generate the time series), and (2) forecasting time series (i.e., to predict the future values of the time-series variables).

Trend analysis consists of the following four major components or movements for characterizing time-series data:

Trend or long-term movements: These indicate the general direction in which a time series graph is moving over a long interval of time. This movement is displayed by a trend curve, or a trend line. For example, the trend curve of Figure 8.4 is indicated by a dashed curve. Typical methods for determining a trend curve or trend line include the weighted moving average method and the least squares method, discussed later.

Cyclic movements or cyclic variations: These refer to the cycles, that is, the long-term oscillations about a trend line or curve, which may or may not be periodic. That is, the cycles need not necessarily follow exactly similar patterns after equal intervals of time.

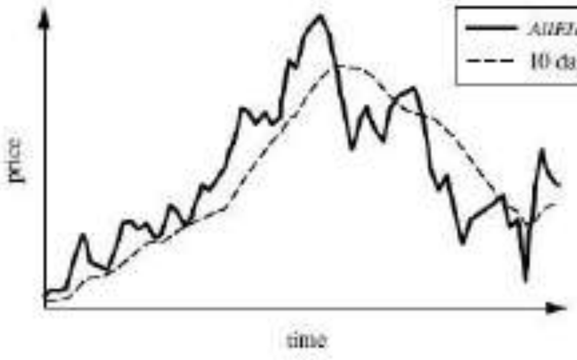


Figure 8.4 Time-series data of the stock price of *AllElectronics* over time. The *trend* is shown with a dashed curve, calculated by a moving average.

Seasonal movements or seasonal variations: These are systematic or calendar related. Examples include events that recur annually, such as the sudden increase in sales of chocolates and flowers before Valentine's Day or of department store items before Christmas. The observed increase in water consumption in summer due to warm weather is another example. In these examples, seasonal movements are the identical or nearly identical patterns that a time series appears to follow during corresponding months of successive years.

Irregular or random movements: These characterize the sporadic motion of time series due to random or chance events, such as labor disputes, floods, or announced personnel changes within companies.

Mining Sequence Patterns in Transactional Databases

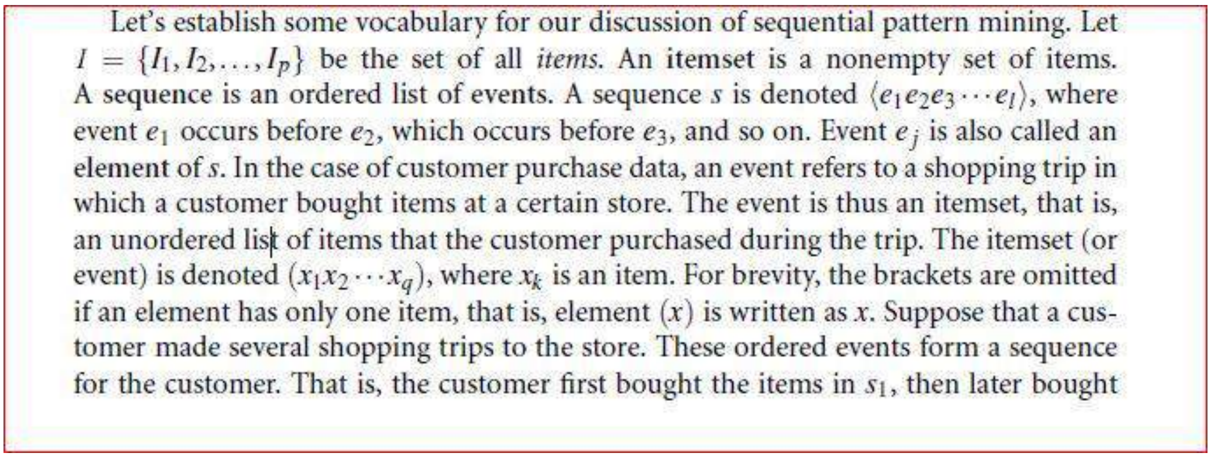
A sequence database consists of sequences of ordered elements or events, recorded with or without a concrete notion of time. There are many applications involving sequence data. Typical examples include customer shopping sequences, Web clickstreams, biological sequences, sequences of events in science and engineering, and in natural and social developments.

1 Sequential Pattern Mining: Concepts and Primitives

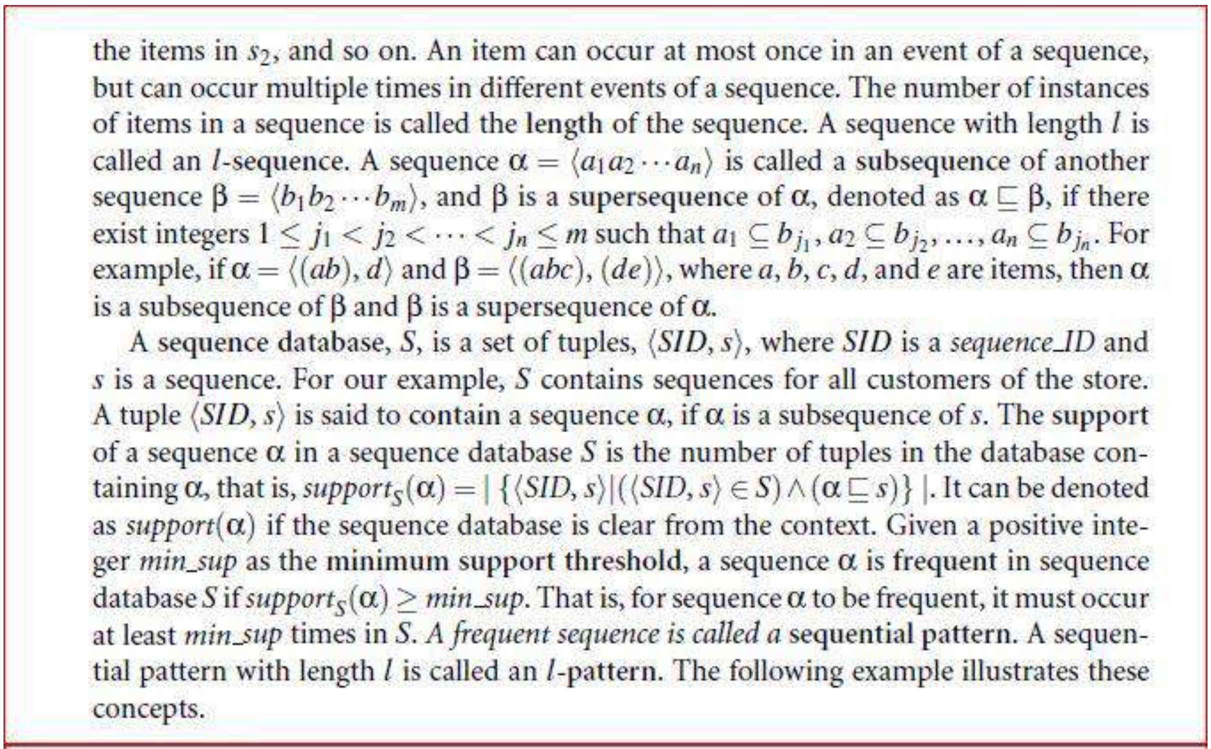
"What is sequential pattern mining?" Sequential pattern mining is the mining of frequently occurring ordered events or subsequences as patterns. An example of a sequential pattern is "Customers who buy a Canon digital camera are likely to buy an HP color printer within a month." For retail data, sequential patterns are useful for shelf placement and promotions. This industry, as well as telecommunications and other businesses, may also use sequential patterns for targeted marketing, customer retention, and many other tasks. Other areas in which sequential patterns can be applied include Web access pattern analysis, weather prediction, production processes, and network intrusion detection. Notice that most studies of sequential pattern mining concentrate on categorical (or symbolic) patterns,whereas numerical curve analysis usually belongs to the scope of trend analysis and forecasting in statistical time-series analysis, as discussed in Section 8.2.

The sequential pattern mining problem was first introduced by Agrawal and Srikant in 1995 [AS95] 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 elements) and each event consists of a set of items, and given a user-specified minimum support threshold of min sup, sequential pattern mining finds all frequent subsequences, that is, the subsequences whose occurrence frequency in the set of sequences is no less than min sup."

Let's establish some vocabulary for our discussion of sequential pattern mining. Let

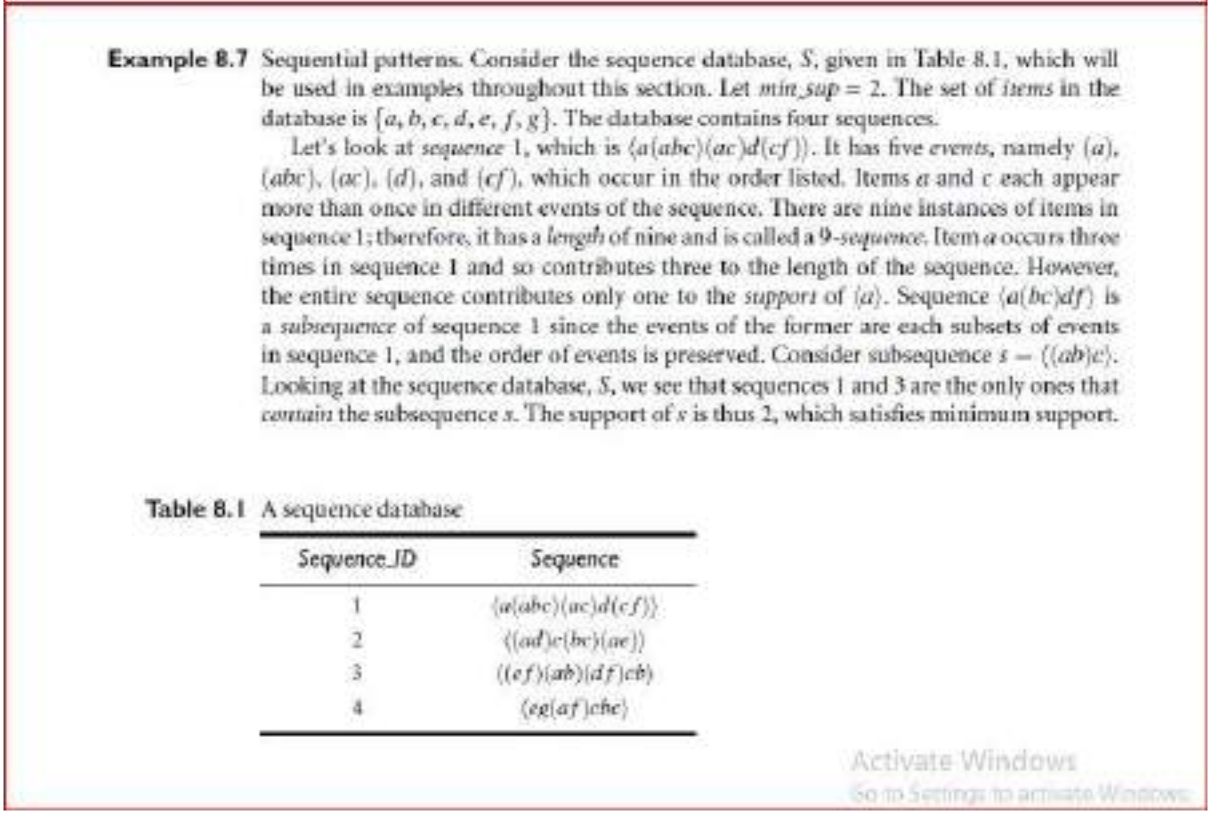


Let's establish some vocabulary for our discussion of sequential pattern mining. Let $I = \{I_1, I_2, \ldots, I_p\}$ be the set of all *items*. An itemset is a nonempty set of items. A sequence is an ordered list of events. A sequence s is denoted $\langle e_1 e_2 e_3 \cdots e_l \rangle$, where event e_1 occurs before e_2, which occurs before e_3, and so on. Event e_j is also called an element of s. In the case of customer purchase data, an event refers to a shopping trip in which a customer bought items at a certain store. The event is thus an itemset, that is, an unordered list of items that the customer purchased during the trip. The itemset (or event) is denoted $(x_1 x_2 \cdots x_q)$, where x_k is an item. For brevity, the brackets are omitted if an element has only one item, that is, element (x) is written as x. Suppose that a customer made several shopping trips to the store. These ordered events form a sequence for the customer. That is, the customer first bought the items in s_1, then later bought



the items in s_2, and so on. An item can occur at most once in an event of a sequence, but can occur multiple times in different events of a sequence. The number of instances of items in a sequence is called the **length** of the sequence. A sequence with length l is called an l**-sequence**. A sequence $\alpha = \langle a_1 a_2 \cdots a_n \rangle$ is called a **subsequence** of another sequence $\beta = \langle b_1 b_2 \cdots b_m \rangle$, and β is a **supersequence** of α, denoted as $\alpha \sqsubseteq \beta$, if there exist integers $1 \leq j_1 < j_2 < \cdots < j_n \leq m$ such that $a_1 \subseteq b_{j_1}, a_2 \subseteq b_{j_2}, \ldots, a_n \subseteq b_{j_n}$. For example, if $\alpha = \langle (ab), d \rangle$ and $\beta = \langle (abc), (de) \rangle$, where a, b, c, d, and e are items, then α is a subsequence of β and β is a supersequence of α.

A sequence database, S, is a set of tuples, $\langle SID, s \rangle$, where SID is a *sequence_ID* and s is a sequence. For our example, S contains sequences for all customers of the store. A tuple $\langle SID, s \rangle$ is said to contain a sequence α, if α is a subsequence of s. The **support** of a sequence α in a sequence database S is the number of tuples in the database containing α, that is, $support_S(\alpha) = |\{\langle SID, s \rangle | (\langle SID, s \rangle \in S) \wedge (\alpha \sqsubseteq s)\}|$. It can be denoted as $support(\alpha)$ if the sequence database is clear from the context. Given a positive integer min_sup as the **minimum support threshold**, a sequence α is **frequent** in sequence database S if $support_S(\alpha) \geq min\_sup$. That is, for sequence α to be frequent, it must occur at least min_sup times in S. *A frequent sequence is called a* **sequential pattern**. A sequential pattern with length l is called an l**-pattern**. The following example illustrates these concepts.



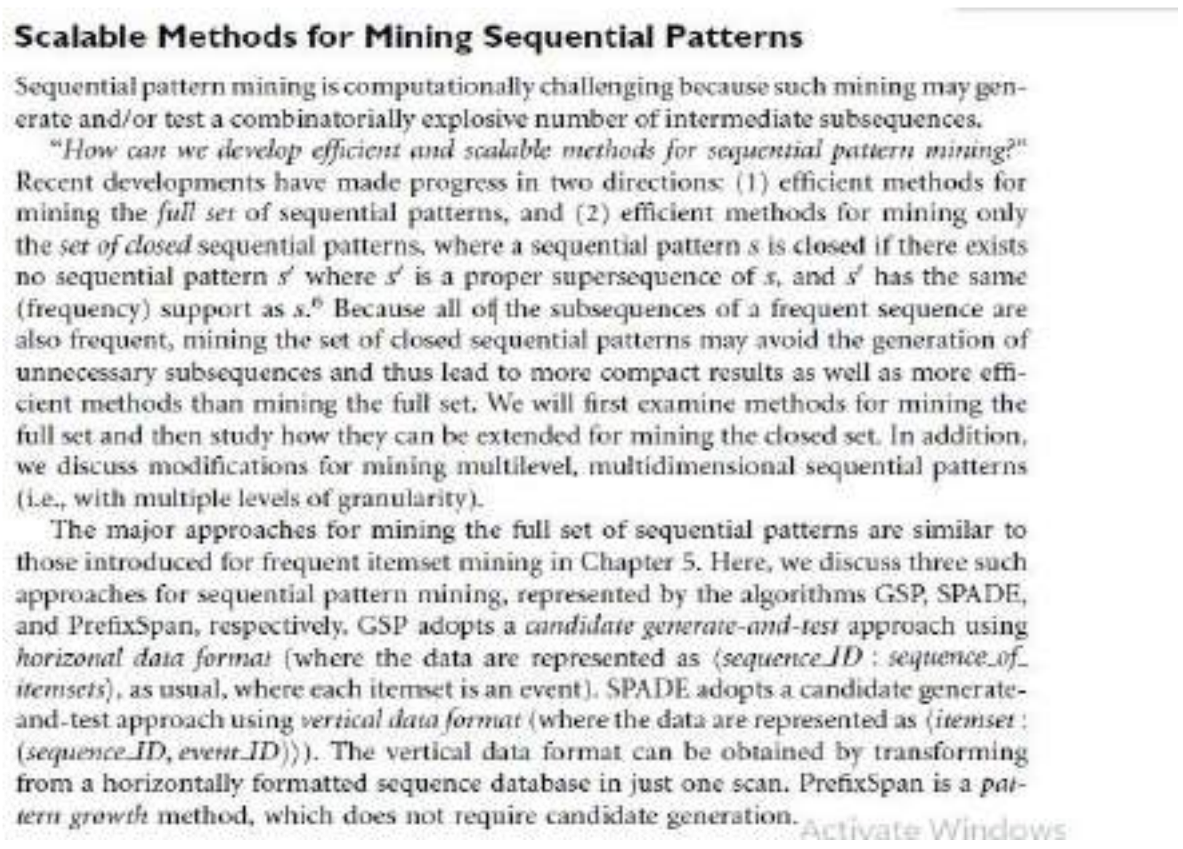
Example 8.7 Sequential patterns. Consider the sequence database, S, given in Table 8.1, which will be used in examples throughout this section. Let $min\_sup = 2$. The set of *items* in the database is $\{a, b, c, d, e, f, g\}$. The database contains four sequences.

Let's look at *sequence* 1, which is $\langle a(abc)(ac)d(cf) \rangle$. It has five *events*, namely (a), (abc), (ac), (d), and (cf), which occur in the order listed. Items a and c each appear more than once in different events of the sequence. There are nine instances of items in sequence 1; therefore, it has a *length* of nine and is called a 9-*sequence*. Item a occurs three times in sequence 1 and so contributes three to the length of the sequence. However, the entire sequence contributes only one to the *support* of $\langle a \rangle$. Sequence $\langle a(bc)df \rangle$ is a *subsequence* of sequence 1 since the events of the former are each subsets of events in sequence 1, and the order of events is preserved. Consider subsequence $s = \langle (ab)c \rangle$. Looking at the sequence database, S, we see that sequences 1 and 3 are the only ones that *contain* the subsequence s. The support of s is thus 2, which satisfies minimum support.

Table 8.1 A sequence database

Sequence_ID	Sequence
1	$\langle a(abc)(ac)d(cf) \rangle$
2	$\langle (ad)c(bc)(ae) \rangle$
3	$\langle (ef)(ab)(df)cb \rangle$
4	$\langle eg(af)cbc \rangle$

Therefore, s is frequent, and so we call it a sequential pattern. It is a 3-pattern since it is a sequential pattern of length three.



Scalable Methods for Mining Sequential Patterns

Sequential pattern mining is computationally challenging because such mining may generate and/or test a combinatorially explosive number of intermediate subsequences.

"How can we develop efficient and scalable methods for sequential pattern mining?" Recent developments have made progress in two directions: (1) efficient methods for mining the *full set* of sequential patterns, and (2) efficient methods for mining only the *set of closed* sequential patterns, where a sequential pattern s is closed if there exists no sequential pattern s' where s' is a proper supersequence of s, and s' has the same (frequency) support as s.[6] Because all of the subsequences of a frequent sequence are also frequent, mining the set of closed sequential patterns may avoid the generation of unnecessary subsequences and thus lead to more compact results as well as more efficient methods than mining the full set. We will first examine methods for mining the full set and then study how they can be extended for mining the closed set. In addition, we discuss modifications for mining multilevel, multidimensional sequential patterns (i.e., with multiple levels of granularity).

The major approaches for mining the full set of sequential patterns are similar to those introduced for frequent itemset mining in Chapter 5. Here, we discuss three such approaches for sequential pattern mining, represented by the algorithms GSP, SPADE, and PrefixSpan, respectively. GSP adopts a *candidate generate-and-test* approach using *horizonal data format* (where the data are represented as ⟨*sequence_ID* : *sequence_of_itemsets*⟩, as usual, where each itemset is an event). SPADE adopts a candidate generate-and-test approach using *vertical data format* (where the data are represented as ⟨*itemset* : (*sequence_ID*, *event_ID*)⟩). The vertical data format can be obtained by transforming from a horizontally formatted sequence database in just one scan. PrefixSpan is a *pattern growth* method, which does not require candidate generation.

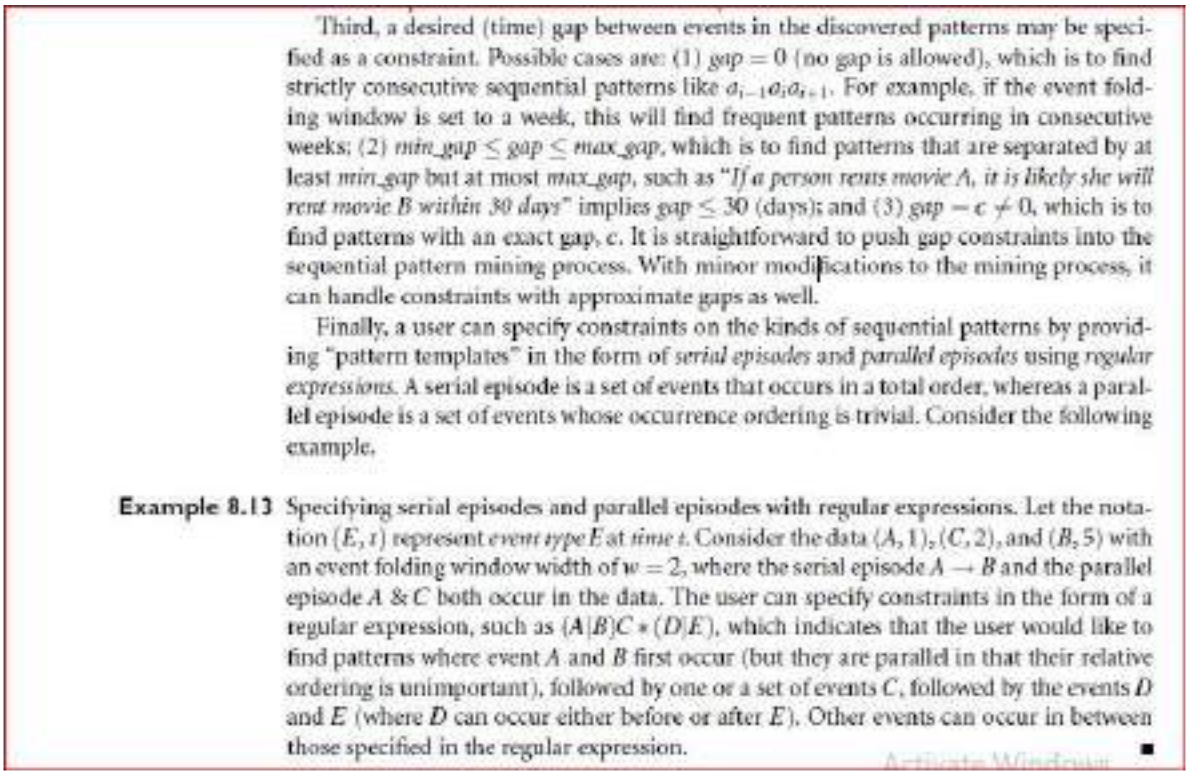
All three approaches either directly or indirectly explore the Apriori property, stated as follows: every nonempty subsequence of a sequential pattern is a sequential pattern. (Recall that for a pattern to be called sequential, it must be frequent. That is, it must satisfy minimum support.) The Apriori property is anti monotonic (or downward-closed) in that, if a sequence cannot pass a test (e.g., regarding minimum support), all of its super sequences will also fail the test. Use of this property to prune the search space can help make the discovery of sequential patterns more efficient.

Constraint-Based Mining of Sequential Patterns

As shown in our study of frequent-pattern mining in Chapter 5, mining that is performed without user- or expert-specified constraints may generate numerous patterns that are of no interest. Such unfocused mining can reduce both the efficiency and usability of frequent-pattern mining. Thus, we promote constraint-based mining, which incorporates user-specified constraints to reduce the search space and derive only patterns that are of interest to the user.

Constraints can be expressed in many forms. They may specify desired relationships between attributes, attribute values, or aggregates within the resulting patterns mined. Regular expressions can also be used as constraints in the form of "pattern templates," which specify the desired form of the patterns to be mined. The general concepts introduced for constraint-based frequent pattern mining in Section 5.5.1 apply to constraint-based sequential pattern mining as well. The key idea to note is that these kinds of constraints can be used during the mining process to confine the search space, thereby improving (1) the efficiency of the mining and (2) the interestingness of the resulting patterns found. This idea is also referred to as "pushing the constraints

deep into the mining process."
We now examine some typical examples of constraints for sequential pattern mining.
First, constraints can be related to the duration, T, of a sequence. The duration may
be the maximal or minimal length of the sequence in the database, or a user-specified
duration related to time, such as the year 2005. Sequential pattern mining can then be
confined to the data within the specified duration, T.
Constraints relating to the maximal or minimal length (duration) can be treated as
antimonotonic or monotonic constraints, respectively. For example, the constraint T _10
is ant monotonic since, if a sequence does not satisfy this constraint, then neither will
any of its super sequences (which are, obviously, longer). The constraint T >10 is monotonic.
This means that if a sequence satisfies the constraint, then all of its super sequences
will also satisfy the constraint. We have already seen several examples in this chapter
of how antimonotonic constraints (such as those involving minimum support) can be
pushed deep into the mining process to prune the search space. Monotonic constraints
can be used in a way similar to its frequent-pattern counterpart as well.
Constraints related to a specific duration, such as a particular year, are considered
succinct constraints. A constraint is succinct if we can enumerate all and only those
sequences that are guaranteed to satisfy the constraint, even before support counting
begins. Suppose, here, T = 2005. By selecting the data for which year = 2005, we can
enumerate all of the sequences guaranteed to satisfy the constraint before mining begins.
In other words, we don't need to generate and test. Thus, such constraints contribute
toward efficiency in that they avoid the substantial overhead of the generate-and-test
paradigm.
Durations may also be defined as being related to sets of partitioned sequences, such
as every year, or every month after stock dips, or every two weeks before and after an
earthquake. In such cases, periodic patterns (Section 8.3.4) can be discovered.
Second, the constraint may be related to an event folding window, w. A set of events
occurring within a specified period can be viewed as occurring together. If w is set to be as
long as the duration, T, it finds time-insensitive frequent patterns—these are essentially
frequent patterns, such as "In 1999, customers who bought a PC bought a digital camera
as well" (i.e., without bothering about which items were bought first). If w is set to 0
(i.e., no event sequence folding), sequential patterns are found where each event occurs
at a distinct time instant, such as "A customer who bought a PC and then a digital camera
is likely to buy an SD memory chip in a month." If w is set to be something in between
(e.g., for transactions occurring within the same month or within a sliding window of
24 hours), then these transactions are considered as occurring within the same period,
and such sequences are "folded" in the analysis.



Third, a desired (time) gap between events in the discovered patterns may be specified as a constraint. Possible cases are: (1) $gap = 0$ (no gap is allowed), which is to find strictly consecutive sequential patterns like $a_{i-1}a_ia_{i+1}$. For example, if the event folding window is set to a week, this will find frequent patterns occurring in consecutive weeks; (2) $min\_gap \leq gap \leq max\_gap$, which is to find patterns that are separated by at least *min_gap* but at most *max_gap*, such as "*If a person rents movie A, it is likely she will rent movie B within 30 days*" implies $gap \leq 30$ (days); and (3) $gap = c \neq 0$, which is to find patterns with an exact gap, c. It is straightforward to push gap constraints into the sequential pattern mining process. With minor modifications to the mining process, it can handle constraints with approximate gaps as well.

Finally, a user can specify constraints on the kinds of sequential patterns by providing "pattern templates" in the form of *serial episodes* and *parallel episodes* using *regular expressions*. A serial episode is a set of events that occurs in a total order, whereas a parallel episode is a set of events whose occurrence ordering is trivial. Consider the following example.

Example 8.13 Specifying serial episodes and parallel episodes with regular expressions. Let the notation (E, t) represent *event type* E at *time* t. Consider the data $(A, 1), (C, 2)$, and $(B, 5)$ with an event folding window width of $w = 2$, where the serial episode $A \rightarrow B$ and the parallel episode A & C both occur in the data. The user can specify constraints in the form of a regular expression, such as $(A|B)C * (D|E)$, which indicates that the user would like to find patterns where event A and B first occur (but they are parallel in that their relative ordering is unimportant), followed by one or a set of events C, followed by the events D and E (where D can occur either before or after E). Other events can occur in between those specified in the regular expression. ■

Mining Objects

Many advanced, data-intensive applications, such as scientific research and engineering design, need to store, access, and analyze complex but relatively structured data objects.
These objects cannot be represented as simple and uniformly structured records (i.e., tuples) in data relations. Such application requirements have motivated the design and development of object-relational and object-oriented database systems. Both kinds of systems deal with the efficient storage and access of vast amounts of disk-based complex structured data objects. These systems organize a large set of complex data objects into classes, which are in turn organized into class/subclass hierarchies. Each object in a class is associated with
(1) an object-identifier,
(2) a set of attributes that may contain sophisticated data structures, set- or list-valued data, class composition hierarchies, multimedia data, and
(3) a set of methods that specify the computational routines or rules associated with the object class. There has been extensive research in the field of database systems on how to efficiently index, store, access, and manipulate complex objects in object-relational and object-oriented database systems.

One step beyond the storage and access of massive-scaled, complex object data is the systematic analysis and mining of such data.
This includes two major tasks:
(1) construct multidimensional data warehouses for complex object data and perform online analytical processing (OLAP) in such data warehouses, and
(2) develop effective and scalable methods for mining knowledge from object databases and/or data warehouses.
The second task is largely covered by the mining of specific kinds of data (such as spatial, temporal, sequence, graph- or tree-structured, text, and multimedia data), since these data form the major new kinds of complex data objects.
A major limitation of many commercial data warehouse and OLAP tools for multidimensional database analysis is their restriction on the allowable data types for dimensions and measures. Most data cube implementations confine dimensions to nonnumeric data, and measures to simple, aggregated values. To introduce data mining and multidimensional data analysis for complex objects, this section examines how to perform generalization on complex structured objects and construct object cubes for OLAP and mining in object databases.
To facilitate generalization and induction in object-relational and object-oriented databases, it is important to study how each component of such databases can be generalized, and how the generalized data can be used for multidimensional data analysis and data mining.

Generalization of Structured Data

An important feature of object-relational and object-oriented databases is their capability of storing, accessing, and modeling complex structure-valued data, such as set- and list-valued data and data with nested structures.
"How can generalization be performed on such data?"
Typically, set-valued data can be generalized by (1) generalization of each value in the set to its corresponding higher-level concept, or (2) derivation of the general behavior of the set, such as the number of elements in the set, the types or value ranges in the set, the weighted average for numerical data, or the major clusters formed by the set. Moreover, generalization can be

performed by applying different generalization operators to explore alternative generalization paths. In this case, the result of generalization is a heterogeneous set.

Spatial Data Mining

A spatial database stores a large amount of space-related data, such as maps, pre-processed remote sensing or medical imaging data, and VLSI chip layout data. Spatial databases have many features distinguishing them from relational databases. They carry topological and/or distance information, usually organized by sophisticated, multidimensional spatial indexing structures that are accessed by spatial data access methods and often require spatial reasoning, geometric computation, and spatial knowledge representation techniques.
Spatial data mining refers to the extraction of knowledge, spatial relationships, or other interesting patterns not explicitly stored in spatial databases. Such mining demands an integration of data mining with spatial database technologies. It can be used for understanding spatial data, discovering spatial relationships and relationships between spatial and non-spatial data, constructing spatial knowledge bases, reorganizing spatial databases, and optimizing spatial queries. It is expected to have wide applications in geographic information systems, geo marketing, remote sensing, image database exploration, medical imaging, navigation, traffic control, environmental studies, and many other areas where spatial data are used. A crucial challenge to spatial data mining is the exploration of efficient spatial data mining techniques due to the huge amount of spatial data and the complexity of spatial data types and spatial access methods.
“What about using statistical techniques for spatial data mining?”
Statistical spatial data analysis has been a popular approach to analyzing spatial data and exploring geographic information. The term geo statistics is often associated with continuous geographic space, whereas the term spatial statistics is often associated with discrete space. In a statistical model that handles non spatial data, one usually assumes statistical independence among different portions of data. However, different from traditional data sets, there is no such independence among spatially distributed data because in reality, spatial objects are often interrelated, or more exactly spatially co-located, in the sense that the closer the two objects are located, the more likely they share similar properties.

1. **Spatial Data Cube Construction and Spatial OLAP**

“Can we construct a spatial data warehouse?” Yes, as with relational data, we can integrate spatial data to construct a data warehouse that facilitates spatial data mining. A spatial data warehouse is a subject-oriented, integrated, time-variant, and non-volatile collection of both spatial and nonspatial data in support of spatial data mining and spatial-data related decision-making processes.
I need to use spatial measures in a spatial data cube then this notion raises some challenging issues on efficient implementation as,
Numerical versus spatial measures.
A star schema for the BC weather warehouse of Example 10.5 is shown in Figure 10.2.

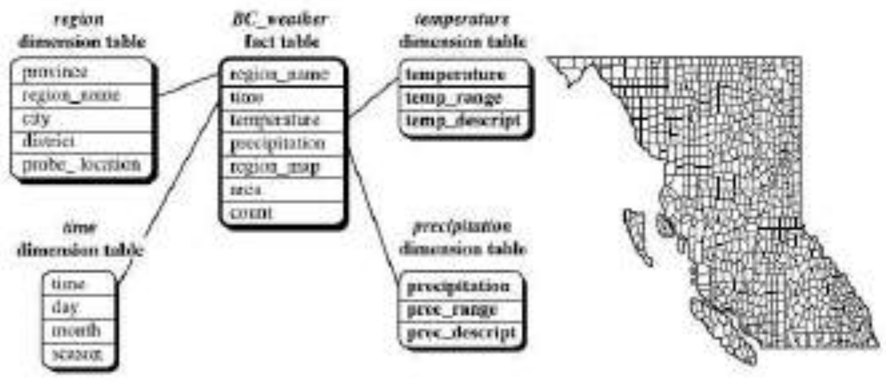


Figure 10.2 A star schema of the *BC_weather* spatial data warehouse and corresponding BC weather probes map.

It consists of four dimensions: region temperature, time, and precipitation, and three measures: region map, area, and count. A concept hierarchy for each dimension can be created by users or experts, or generated automatically by data clustering analysis.

Multimedia Data Mining

"What is a multimedia database?" A multimedia database system stores and manages a large collection of multimedia data, such as audio, video, image, graphics, speech, text, document, and hypertext data, which contain text, text markups, and linkages. Multimedia database systems are increasingly common owing to the popular use of audio video equipment, digital cameras, CD-ROMs, and the Internet. Typical multimedia database systems include NASA's EOS (Earth Observation System), various kinds of image and audio-video databases, and Internet databases.

1 **Similarity Search in Multimedia Data**

"When searching for similarities in multimedia data, can we search on either the data description or the data content?" That is correct. For similarity searching in multimedia data, we consider two main families of multimedia indexing and retrieval systems:

(1) description-based retrieval systems, which build indices and perform object retrieval based on image descriptions, such as keywords, captions, size, and time of creation;

(2) content-based retrieval systems, which support retrieval based on the image content, such as color histogram, texture, pattern, image topology, and the shape of objects and their layouts and locations within the image.

Description-based retrieval is labour-intensive if performed manually. If automated, the results are typically of poor quality. For example, the assignment of keywords to images can be a tricky and arbitrary task. Recent development of Web-based image clustering and classification methods has improved the quality of description-based Web image retrieval, because image surrounded text information as well as Web linkage information can be used to extract proper description and group images describing a similar theme together. Content-based retrieval uses visual features to index images and promotes object retrieval based on feature similarity, which is highly desirable in many applications.

Content-based image retrieval system, there are often two kinds of queries: image sample-based queries and image feature specification queries. Image-sample-based queries find all of the images that are similar to the given image sample. This search compares the feature vector (or signature) extracted from the sample with the feature vectors of images that have already been extracted and indexed in the image database. Based on this comparison, images that are close to the sample image are returned. Image feature specification queries specify or sketch image features like color, texture, or shape, which are translated into a feature vector to be matched with the feature vectors of the images in the database. Content-based retrieval has wide applications, including medical diagnosis, weather prediction, TV production, Web search engines for images, and e-commerce.

Color histogram–based signature: In this approach, the signature of an image includes color histograms based on the color composition of an image regardless of its scale or orientation. This method does not contain any information about shape, image topology, or texture. Thus, two images with similar color composition but that contain very different shapes or textures may be identified as similar, although they could be completely unrelated semantically.

Multifeature composed signature: In this approach, the signature of an image includes a composition of multiple features: color histogram, shape, image topology, and texture. The extracted image features are stored as metadata, and images are indexed based on such metadata. Often, separate distance functions can be defined for each feature and subsequently combined to derive the overall results.

2 **Multidimensional Analysis of Multimedia Data**

"Can we construct a data cube for multimedia data analysis?" To facilitate the multidimensional analysis of large multimedia databases, multimedia data cubes can be designed and constructed in a manner similar to that for traditional data cubes from relational data. A multimedia data cube can contain additional dimensions and measures for multimedia information, such as color, texture, and shape.

3 **Classification and Prediction Analysis of Multimedia Data**

Classification and predictive modelling have been used for mining multimedia data, especially in scientific research, such as astronomy, seismology, and geoscientific research.
.

4 **Mining Associations in Multimedia Data**

"What kinds of associations can be mined in multimedia data?" Association rules involving multimedia objects can be mined in image and video databases. At least three categories
can be observed:

Associations between image content and nonimage content features:
A rule like "If at least 50% of the upper part of the picture is blue, then it is likely to represent sky" belongs to this category since it links the image content to the keyword sky.

Associations among image contents that are not related to spatial relationships: A rule like "If a picture contains two blue squares, then it is likely to contain one red circle
As well" belongs to this category since the associations are all regarding image contents.

Associations among image contents related to spatial relationships: A rule like "If
a red triangle is between two yellow squares, then it is likely a big oval-shaped object
is underneath" belongs to this category since it associates objects in the image with
spatial relationships.

other content-based multimedia features, such as color, shape, texture, and keywords,
may form interesting associations. Thus, spatial data mining methods and properties of
topological spatial relationships become important for multimedia mining.

5 **Audio and Video Data Mining**
esides still images, an incommensurable amount of audio visual information is becoming available in digital form, in digital archives, on the World Wide Web, in broadcast data streams, and in personal and professional databases. This amount is rapidly growing. There are great demands for effective content-based retrieval and data mining methods for audio and video data.

Text Mining

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 the World Wide Web (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 semi structured data in that they are neither completely unstructured nor completely structured.
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.

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 system problems are usually not present in information retrieval systems, such as concurrency control, recovery, transaction management, and update.

Text Retrieval Methods

"What methods are there for information retrieval?" Broadly speaking, retrieval methods fall into two categories: They generally either view the retrieval problem as a document selection problem or as a document ranking problem. In document selection methods, the query is regarded as specifying constraints for selecting relevant documents. A typical method of this category is the Boolean retrieval model, in which a document is represented by a set of keywords and a user provides a Boolean expression of keywords, such as "car and repair shops," "tea or coffee," or "database systems but not Oracle." The retrieval system would take such a Boolean query and return documents that satisfy the Boolean expression. Because of the difficulty in prescribing a user's information need exactly with a Boolean query, the Boolean retrieval method generally only works well when the user knows a lot about the document collection and can formulate a good query in this way.
Document ranking methods use the query to rank all documents in the order of relevance. For ordinary users and exploratory queries, these methods are more appropriate than document selection methods. Most modern information retrieval systems present a ranked list of documents in response to a user's keyword query. There are many different ranking methods based on a large spectrum of mathematical foundations, including algebra, logic, probability, and statistics. The common intuition behind all of these methods is that we may match the keywords in a query with those in the documents and score each document based on how well it matches the query. The goal is to approximate the degree of relevance of a document with a score computed based on information such as the frequency of words in the document and the whole collection.

Mining the World Wide Web

The World Wide Web serves as a huge, widely distributed, global information service center for news, advertisements, consumer information, financial management, education, government, e-commerce, and many other information services. The Web also contains a rich and dynamic collection of hyperlink information and Web page access and usage information, providing rich sources for data mining.
The Web seems to be too huge for effective data warehousing and data mining. The size of the Web is in the order of hundreds of terabytes and is still growing rapidly. Many organizations and societies place most of their public-accessible information on the Web. It is barely possible to set up a data warehouse to replicate, store, or integrate all of the data on the Web.

The complexity of Web pages is far greater than that of any traditional text document collection. Web pages lack a unifying structure. They contain far more authoring style and content variations than any set of books or other traditional text-based documents. The Web is considered a huge digital library; however, the tremendous number of documents in this library are not arranged according to any particular sorted order. There is no index by category, nor by title, author, cover page, table of contents, and so on. It can be very challenging to search for the information you desire in such a library!
The Web is a highly dynamic information source. Not only does the Web grow rapidly, but its information is also constantly updated. News, stock markets, weather, sports, shopping, company advertisements, and numerous other Web pages are updated regularly on the Web. Linkage information and access records are also updated frequently.

Only a small portion of the information on the Web is truly relevant or useful. It is said that 99% of the Web information is useless to 99% of Web users. Although this may not seem obvious, it is true that a particular person is generally interested in only a tiny portion of the Web, while the rest of the Web contains information that is uninteresting to the user and may swamp desired search results. Hyperlinks are for endorsement, then the collective opinion will still dominate. Second, for commercial or competitive interests, one authority will seldom have its Web page point to its rival authorities in the same field. For example, *Coca-Cola* may prefer not to endorse its competitor *Pepsi* by not linking to *Pepsi*'s Web pages. Third, authoritative pages are seldom particularly descriptive. For example, the main Web page of Yahoo! may not contain the explicit self-description *"Web search engine."* These properties of Web link structures have led researchers to consider another important category of Web pages called a *hub*. A hub is one or a set of Web pages that provides collections of links to authorities. Hub pages may not be prominent, or there may exist few links pointing to them; however, they provide links to a collection of prominent sites on a common topic. Such pages could be lists of recommended links on individual home pages, such as recommended reference sites from a course home page, or professionally assembled resource lists on commercial sites. Hub pages play the role of implicitly conferring authorities on a focused topic.

Mining Multimedia Data on the Web

A huge amount of multimedia data are available on the Web in different forms. These include video, audio, images, pictures, and graphs. There is an increasing demand for effective methods for organizing and retrieving such multimedia data. Compared with the general-purpose multimedia data mining, the multimedia data on the Web bear many different properties. Web-based multimedia data are embedded on the Web page and are associated with text and link information. These texts and links can also be regarded as features of the multimedia data. Using some Web page layout mining techniques (like VIPS), a Web page can be partitioned into a set of semantic blocks.

Automatic Classification of Web Documents
In the automatic classification of Web documents, each document is assigned a class label from a set of predefined topic categories, based on a set of examples of pre classified documents. For example, Yahoo!'s taxonomy and its associated documents can be used as training and test sets in order to derive a Web document classification scheme. This scheme may then be used to classify new Web documents by assigning categories from the same taxonomy.

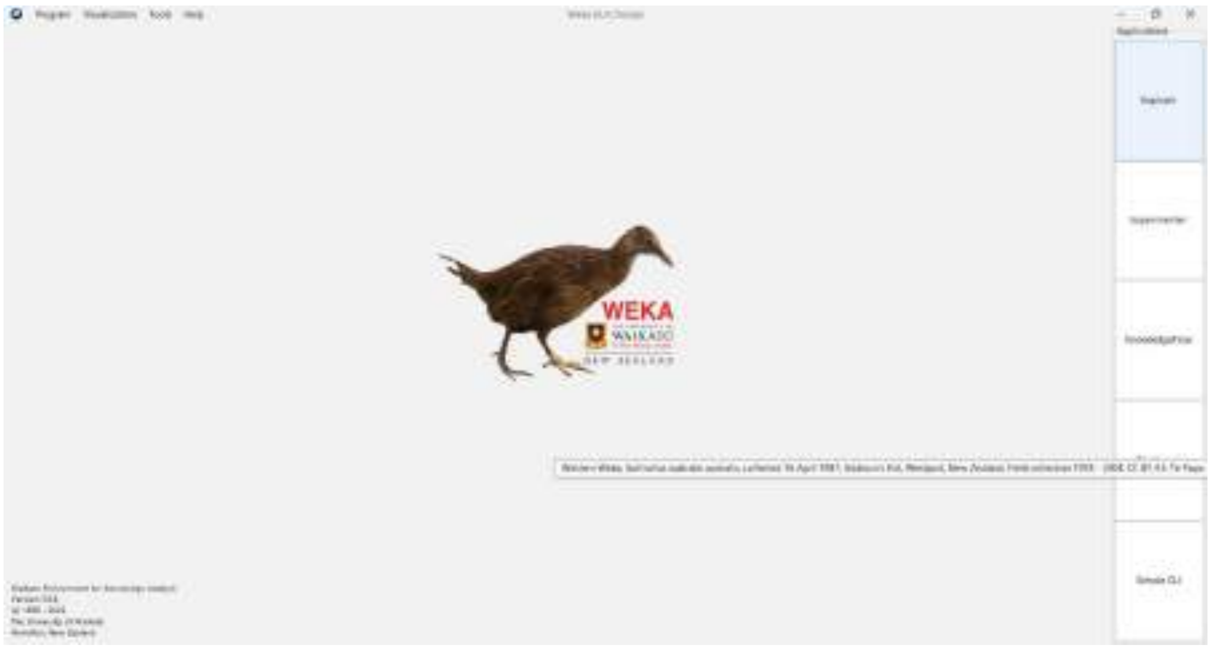
Web Usage Mining
"What is Web usage mining?" Besides mining Web contents and Web linkage structures, another important task for Web mining is Web usage mining, which mines Weblog records to discover user access patterns of Web pages. Analysing and exploring regularities in Weblog records can identify potential customers for electronic commerce, enhance the quality and delivery of Internet information services to the end user, and improve Web server system performance.
A Web server usually registers a (Web) log entry, or Weblog entry, for every access of a Web page. It includes the URL requested, the IP address from which the request originated,
and a timestamp. For Web-based e-commerce servers, a huge number of Web access log records are being collected. Popular websites may register Weblog records in the order of hundreds of megabytes every day. Weblog databases provide rich information about Web dynamics. Thus it is important to develop sophisticated Weblog mining techniques.

EMPLOYMENT DATASET

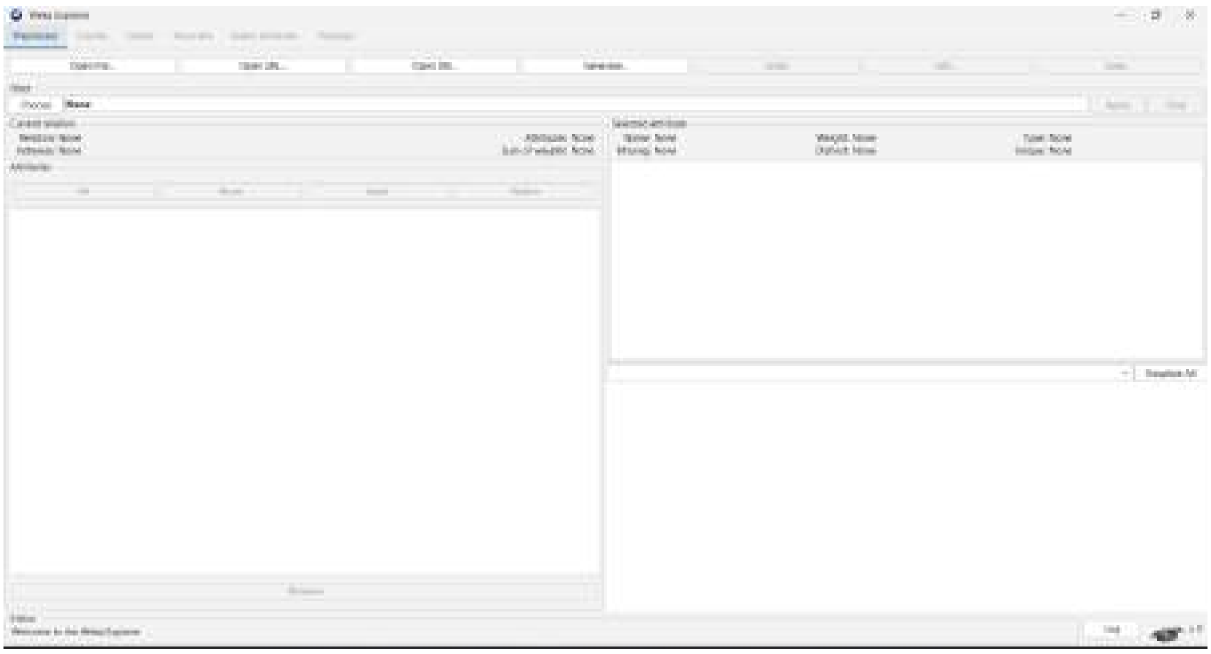
Weka Installation:

Step 1: Go to Google and Search Weka 3.8.6 Download
Step 2: Download from any links.

After setup is done the weka GUI will like this as below:



After Clicking on Explorer you will get UI as below:



Data Pre-processing:

Start the Weka Explorer:

Step 1: Open the Weka GUI Chooser.

Step 2: Click the “Explorer” button to open the Weka Explorer.

Step 3: Load the Employee dataset from the labor.arff file.

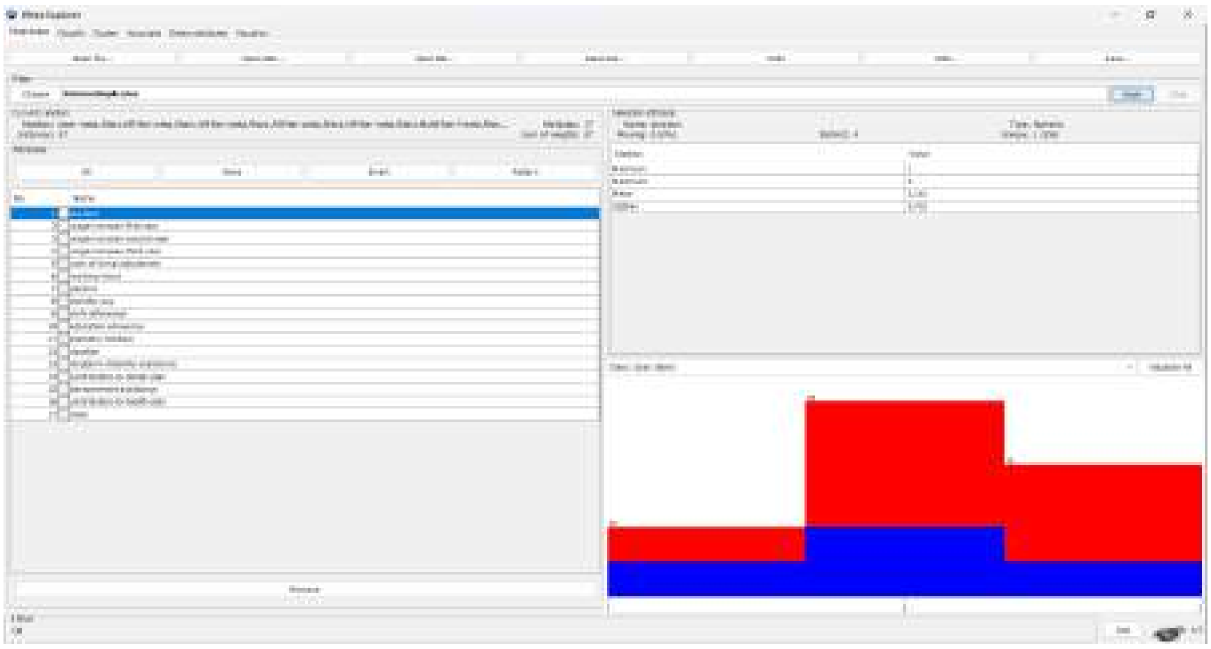
Step 4: Click “Classify” to open the Classify tab.

Step 5: Click the button “Preprocess” on the top left corner

Choose in the following manner.

Weka->Preprocess

Ex: Filter->Choose->Allfilters or any one of your choice



Classification (k-Nearest Neighbour’s (**`IBk`**))

Start the Weka Explorer:

Step 1: Open the Weka GUI Chooser.

Step 2: Click the “Explorer” button to open the Weka Explorer.

Step 3: Load the Employee dataset from the labor.arff file.

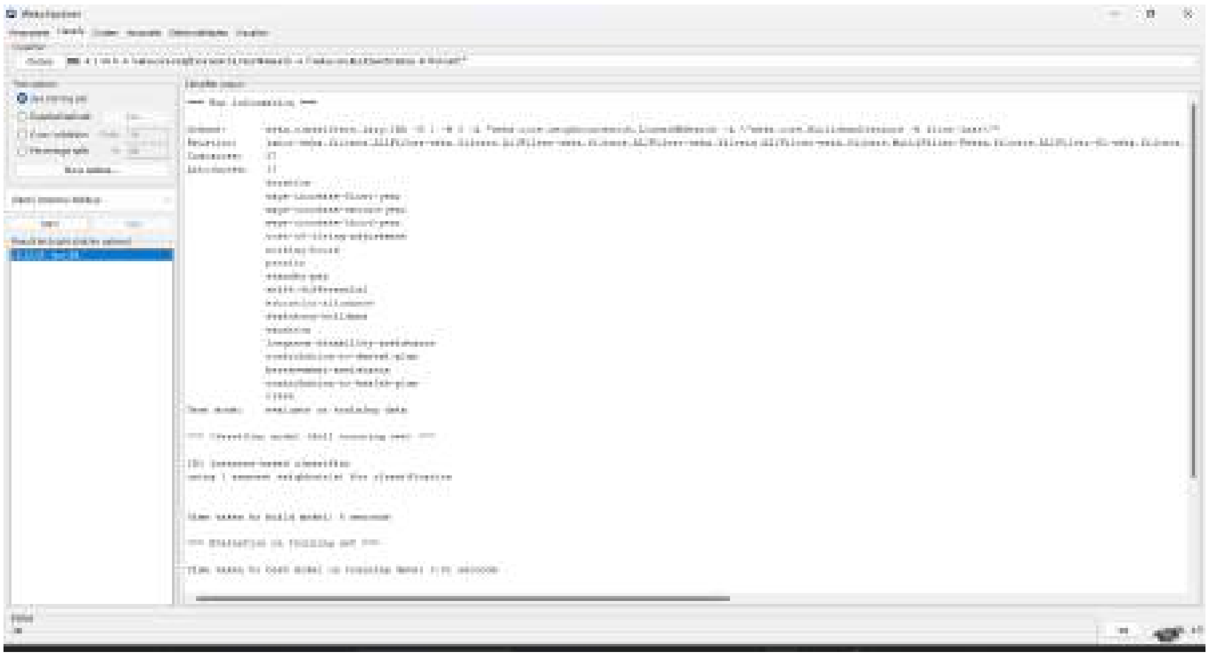
Step 4: Click “Classify” to open the Classify tab.

Step 5: Click the button “Classify” and click on “choose” and select the lazy and from that menu select IBK.

Choose in the following manner.

Weka->Classify->choose->lazy->IBK.

Output as shown below:



```
=== Run information ===

Scheme:       weka.classifiers.lazy.IBk -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""
Relation:     [illegible]
Instances:    57
Attributes:   17
              duration
              wage-increase-first-year
              wage-increase-second-year
              wage-increase-third-year
              cost-of-living-adjustment
              working-hours
              pension
              standby-pay
              shift-differential
              education-allowance
              statutory-holidays
              vacation
              longterm-disability-assistance
              contribution-to-dental-plan
              bereavement-assistance
              contribution-to-health-plan
              class
Test mode:    evaluate on training data

=== Classifier model (full training set) ===

IB1 instance-based classifier
using 1 nearest neighbour(s) for classification


Time taken to build model: 0 seconds

=== Evaluation on training set ===

Time taken to test model on training data: [illegible] seconds
```

Clustering(Simple K-Means):

Start the Weka Explorer:

Step 1: Open the Weka GUI Chooser.

Step 2: Click the “Explorer” button to open the Weka Explorer.

Step 3: Load the Employee dataset from the labor.arff file.

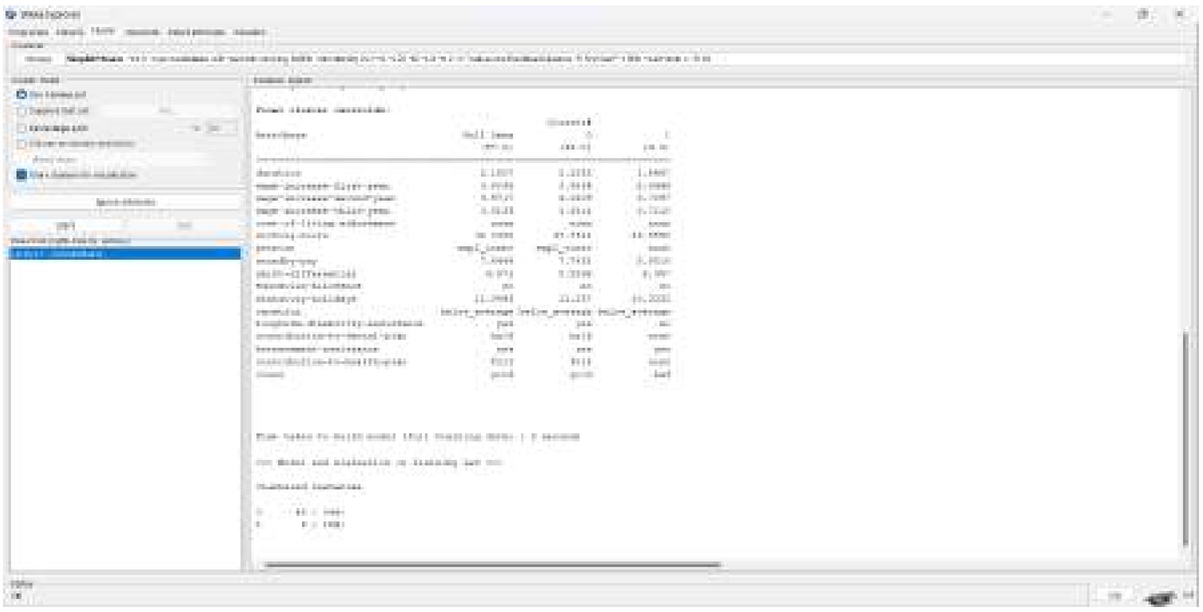
Step 4: Click “Cluster” to open the Classify tab.

Step 5: Click the button “choose” and select on Simple K Means

Choose in the following manner.

Weka->Cluster->Choose->SimpleKMeans

OUTPUT:



[illegible]

Regression(Linear Regression)

Start the Weka Explorer:

Step 1: Open the Weka GUI Chooser.

Step 2: Click the “Explorer” button to open the Weka Explorer.

Step 3: Load the Employee dataset from the labor.arff file.

Step 4: Click “Classify” to open the Classify tab.

Step 5: Click the button “Choose” and select Linear Regression from Functions in Classfiers

Choose in the following manner.

Weka->Classify->Choose->Functions->Linear Regression

Note: If you filter using Discretize you won’t get Linear Regression

Output:

```
=== Classifier model (full training set) ===


Linear Regression Model

[illegible] =

     [illegible]
     [illegible]

Time taken to build model: [illegible] seconds

=== Evaluation on training set ===

Time taken to test model on training data: [illegible] seconds

=== Summary ===

Correlation coefficient                 [illegible]
Mean absolute error                     [illegible]
Root mean squared error                 [illegible]
Relative absolute error                 [illegible] %
Root relative squared error             [illegible] %
Total Number of Instances               [illegible]
```

Visualization:

Start the Weka Explorer:

Step 1: Open the Weka GUI Chooser.

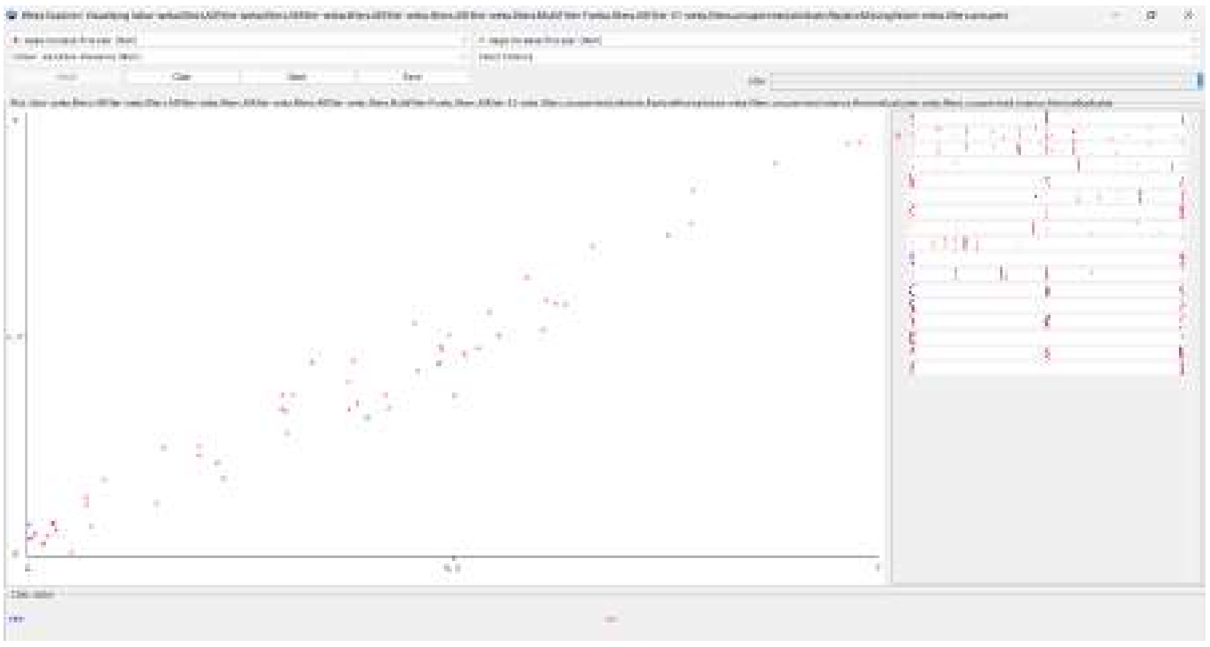
Step 2: Click the “Explorer” button to open the Weka Explorer.

Step 3: Load the Employee dataset from the labor.arff file.

Step 4: Click “Visualize” on the classify tab.

Choose in the following manner.

Weka->Visualize



Association(Apriori Algorithm)

Start the Weka Explorer:

Step 1: Open the Weka GUI Chooser.

Step 2: Click the “Explorer” button to open the Weka Explorer.

Step 3: Load the Employee dataset from the labor.arff file.

Step 4: Click “Classify” to open the Classify tab.

Step 5: Click the button “Associate” and click on Choose Apriori

Choose in the following manner.

Weka->Associate->choose->Apriori

Note:Click on Preprocess and choose filters and select Discretize

```
[illegible]
```