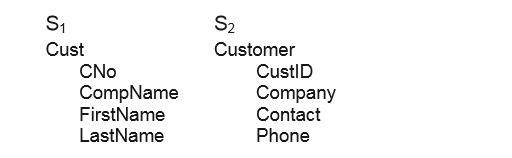
**Introduction to Schema Matching**

The fundamental problem is schema matching, which takes two (or more) database schemas to produce a mapping between elements (or attributes) of the two (or more) schemas that correspond semantically to each other. The objective is to merge the schemas into a single global schema. This problem arises in building a global database that comprises several distinct but related databases. One application scenario in a company is that each department has its database about customers and products that are related to the operations of the department. Each database is typically designed independently and possibly by different people to optimize database operations required by the functions of the department. This results in different database schemas in different departments. However, to consolidate the data about customers or company operations across the organization in order to have a more complete understanding of its customers and to better serve them, integration of databases is needed. The integration problem is clearly also important on the Web as we discussed above, where the task is to integrate data from multiple sites.

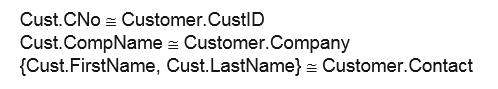
Schema matching is challenging for many reasons. First of all, schemas of identical concepts may have structural and naming differences. Schemas may model similar but not identical contents, and may use different data models. They may also use similar words for different meanings.

Although it may be possible for some specific applications, in general, it is not possible to fully automate all matches between two schemas because some semantic information that determines the matches between two schemas may not be formally specified or even documented. Thus, any automatic algorithm can only generate candidate matches that the user needs to verify, i.e., accept, reject or change. Furthermore, the user should also be allowed to specify matches for elements that the system is not able to find satisfactory match candidates. Let us see a simple example.

Example 1: Consider two schemas, S1 and S2, representing two customer relations, Cust and Customer.



We can represent the mapping with a similarity relation, , over the power sets of S1 and S2, where each pair in  represents one element of the mapping. For our example schemas, we may obtain



There are various types of matching based on the input information.

1. Schema-level only matching: In this type of matching, only the schema information (e.g. names and data types) is considered. No data instance is available.

2. Domain and instance-level only matching: In this type of match, only instance data and possibly the domain of each attribute are provided. No schema is available. Such cases occur quite frequently on the Web, where we need to match corresponding columns of the hidden schemas.

3. Integrated matching of schema, domain and instance data: In this type of match, both schemas and instance data (possibly domain information) are available. The match algorithm can exploit clues from all of them to perform matching.

**Pre-Processing for Schema Matching**

For pre-processing, issues such as concatenated words, abbreviations, and acronyms are dealt with. That is, they need to be normalized before being used in matching

Prep 1 (Tokenization): This process breaks an item, which can be a schema element (attribute) or attribute value, into atomic words. Such items are usually concatenated words. Delimiters (such as “-”, “\_”, etc.) and case changes of letters are used to suggest the breakdown. For example, we can break “fromCity” into “from City”, and “first-name” into “first name”. A domain dictionary of words is typically maintained to help the breakdown. Note that if “from”, “city”, “first” and “name” are not in the dictionary, they will be added to the dictionary. Existing dictionary words are also utilized to suggest the breakdown. For example, “deptcity” will be split into “dept” and “city” if “city” is a word. The dictionary may be constructed automatically, which consists of all the individual words appeared in the given input used in matching, e.g., schemas, instance data and domains. The dictionary is updated as the processing progresses. However, the tokenization step has to be done with care. For example, we have “Baths” and “Bathrooms” if we split “Bath” with “Room” it could be a mistake because “Rooms” could have a very different meaning (the number of rooms in the house). To be sure, we need to ensure that “Bathroom” is not an English word, for which an online English dictionary may be employed.

Prep 2 (Expansion): It expands abbreviations and acronyms to their full words, e.g., from “dept” to “departure”. The expansion is usually done based on the auxiliary information provided by the user or collected from other sources. Constraints may be imposed to ensure that the expansion is likely to be correct. For example, we may require that the word to be expanded is not in the English dictionary, with at least three letters, and having the same first letter as the expanding word. For example, “CompName” is first converted to (Comp, Name) in tokenization, and then “Comp” is expanded to “Company”.

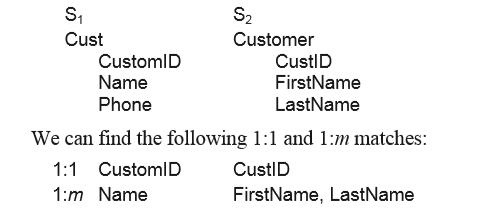
Prep 3 (Stopword removal and stemming): These are information retrieval pre-processing methods (see Chap. 6). They can be performed to attribute names and domain values. A domain specific stopword list may also be constructed manually. This step is useful especially in linguistic based matching methods discussed below.

Prep 4 (Standardization of words): Irregular words are standardized to a single form (e.g., “colour” --“color”, “Children” -- “Child”)

**Schema-Level Matching**

A schema level matching algorithm relies on information about schema elements, such as name, description, data type and relationship types (such as part-of, is-a, etc.), constraints and schema structures. Before introducing some matching methods using such information, let us introduce the notion of match cardinality, which describes the number of elements in one schema that match the number of elements in the other schema. In general, given two schemas, S1 and S2, within a single match in the match relation one or more elements of S1 can match one or more elements of S2. We thus have 1:1, 1:m, m:1 and m:n matches. 1:1 match means that one element of S1 corresponds to one element of S2, and 1:m means that one element of S1 corresponds to a set of m (m > 1) elements of S2.

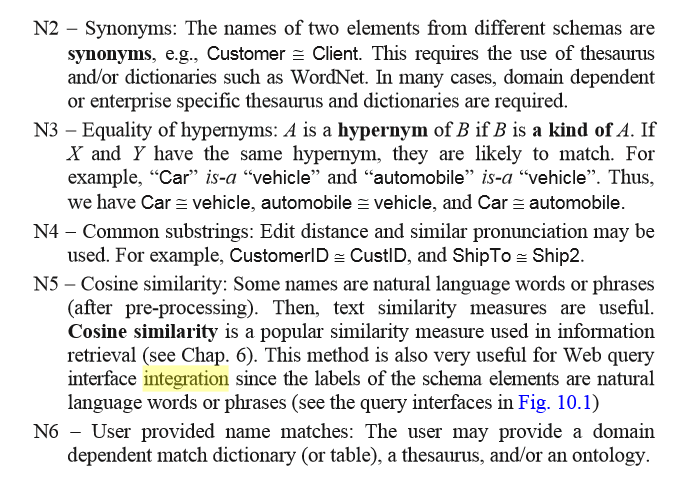
Consider the following schemas:

 m:1 match is similar to 1:m match; m:n match is considerably more complex. An example of an m:n match is to match Cartesian coordinates with polar coordinates. There is little work on such complex matches. Most existing approaches are for 1:1 and 1:m matches.

We now describe some general matching approaches that employ various types of information available in schemas. There are two main types of information in schemas, natural language words and constraints. Thus, there are two main types of approaches to matching.

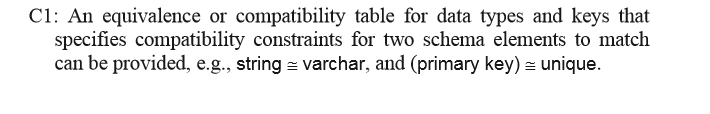
**Linguistic Approaches**

They are used to derive match candidates based on the names, comments or descriptions of schema elements

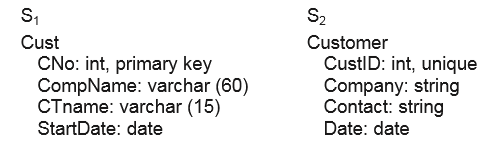


**Constraint Based Approaches**

Constraints such as data types, value ranges, uniqueness, relationship types and cardinalities, etc., can be exploited in determining candidate matches:



Consider the following two schemas:



Constraints can suggest that “CNo” matches “CustID”, and “StartDate” may match “Date”. “CompName” in S1 may match “Company” in S2 or “Contact” in S2. Likewise, “CTname” in S1 may match “Company” or “Contact” in S2. In both cases, the types match. Although in these two cases, we are unable to find a unique match, the approach helps limit the number of match candidates and may be combined with other matchers (e.g., name and instance matchers). For structured schemas, hierarchical relationships such as is-a and part-of relationships may be utilized to help match.

**Domain and Instance-Level Matching**

In this type of matching, value characteristics are exploited to match schema elements For example, the two attribute names may match according to the linguistic similarity, but they may have different domain value characteristics. Then, they may not be the same but homonyms. For example, Location in a real estate sell may mean the address, but could also mean some specific locations, e.g., lakefront property, hillside property, etc. In many applications, data instances are available, which is often the case in the Web database context. In some applications, although the instance information is not available, the domain information of each attribute may be obtained. This is the case for Web query interfaces. Some attributes in the query interface contain a list of possible values (the domain) for the user to choose from. No type information is explicitly given, but it can often be inferred. We note that the set of value instances of an attribute can be treated in the similar way as a domain. Thus, we will only deal with domains below.

Let us look at two types of domains or types of values: simple domains and composite domains. The domain similarity of two attributes, A and B, is the similarity of their domains: dom(A) and dom(B). Definition (Simple Domain): A simple domain is a domain in which each value has only a single component, i.e., the value cannot be decomposed.

A simple domain can be of any type, e.g., year, time, money, area, month, integer, real, string, etc.

**Data Type**: If there is no type specification at the schema level, we identify the data type from the domain values. Even if there is a type specification at the schema level for each attribute, we can still refine the type to find more characteristic patterns. For example, the ISBN number of a book may be specified as a string type in a given schema. However, due to its fixed format, it is easy to generate a characteristic pattern from a set of ISBN numbers, e.g., a regular expression. Other examples include phone numbers, post codes, money, etc. Such specialized patterns are more useful in matching compatible attribute types.

**Semi-automatic approach**: This is done via pattern matching. The pattern for each type may be expressed as a regular expression, which is defined by a human expert. For example, the regular expression for the time type can be defined as “[09]{2}:[09]{2}" or “dd:dd” (d for digit from 0-9) which recognizes time of the form “03:15”. One can use such regular expressions to recognize integer, real, string, month, weekday, date, time, datetime (combination of date and time), etc. To identify the data type, we can simply apply all the regular expression patterns to determine the type. In some cases, the values themselves may contain some information on the type. For example, values that contain “$” or “US$” indicate the monetary type. For all values that we cannot infer their types, we can assume their domains are of string type with an infinite cardinality.

**Automated approach**: Machine learning techniques, e.g., grammar induction, may be used to learn the underlying grammar/pattern of the values of an attribute, and then use the grammar to match attribute values of the other schemas. This method is particularly useful for value of fixed format, e.g., zip codes, phone numbers, zip codes, ISBNs, date entries, or money-related entries, if their regular expressions are not specified by the user.

**The following methods may be used in matching:**

DI 1 – Data types are used as constraints. The method C1 above is applicable here. If the data/domain types of two attributes are not compatible, they should not be matched. We can use a table specifying the degree of compatibility between a set of predefined generic data types, to which data types of schema elements are mapped in order to determine their similarity.

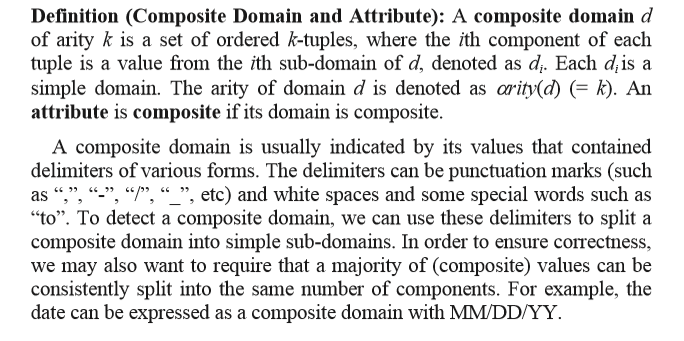
DI 2 – For numerical data, value ranges, averages and variances can be computed to access the level of similarity.

DI 3 – For categorical data, we can extract and compare the set of values in the two domains to check whether the two attributes from different schemas share some common values. For example, if an attribute from S1 contains many “Microsoft” entries and an attribute in S2 also contains some “Microsoft”’s, then we can propose them as a match candidate.

DI 4 – For alphanumeric data, string-lengths and alphabetic/non-alphabetic ratios are also helpful.

DI 5 – For textual data, information retrieval methods such as the cosine measure may be used to compare the similarity of all data values in the two attributes.

DI 6 – Schema element name as value is another match indicator, which characterizes the cases where matches relate some data instances of a schema with a set of elements (attributes) in another schema. For example, in the airfare domain one schema uses “Economy” and “Business” as instances (values) of the attribute “Ticket Class”, while in another interface, “Economy” and “Business” are attributes with the Boolean domain (i.e., “Yes” and “No”). This kind of match can be detected if the words used in one schema as attribute names are among the values of attributes in another schema.



DI 7 – The similarity of a simple domain and a composite domain is determined by comparing the simple domain with each sub-domain of the composite domain. The similarity of composite domains is established by comparing their component sub-domains. We note that splitting a composite domain can be quite difficult in the Web context. For example, without sufficient auxiliary information (e.g., information from other sites) it is not easy to split the following: “Dell desktop PC 1.5GHz 1GB RAM 30GB disk space”.

Sure, let's break down the key points in simpler terms with an example:

**Types of Matching:**

1. **Domain and Instance-Level Matching:**
   * **Idea:** Matching elements based on their names and values.
   * **Example:** Imagine two databases. They both have an attribute named "Location," but in one database, it means an address, while in the other, it could mean specific locations like lakefront or hillside. It's about finding similarities despite potential differences.
2. **Simple and Composite Domains:**
   * **Simple Domain (Example):** Think of simple domains as straightforward categories like years, times, or areas. Each value is simple and can't be broken down.
   * **Composite Domain (Example):** A composite domain is more complex. For instance, "Dell desktop PC 1.5GHz 1GB RAM 30GB disk space" is a composite domain with different components.

**Matching Methods:**

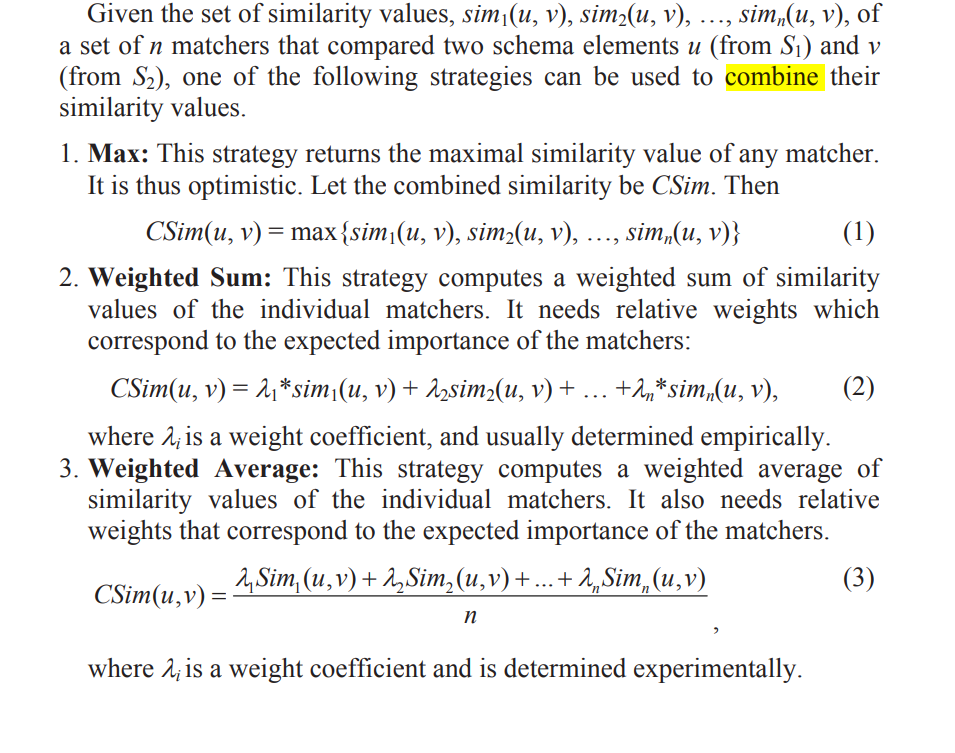
1. **Data Type Matching (DI 1):**
   * **Idea:** Making sure we compare and match only similar types of data.
   * **Example:** If one database uses numbers to represent years and another uses words, they might not match well because they're different data types.
2. **Numerical Data Matching (DI 2):**
   * **Idea:** Comparing numerical values like ranges and averages.
   * **Example:** If one database has prices ranging from $10 to $100, and another has prices from $50 to $150, we could say they are somewhat similar.
3. **Categorical Data Matching (DI 3):**
   * **Idea:** Matching based on categories or groups of values.
   * **Example:** If one database has many entries with "Microsoft," and another also has some "Microsoft" entries, we could consider them as potential matches.
4. **Alphanumeric Data Matching (DI 4):**
   * **Idea:** Considering factors like the length and composition of strings.
   * **Example:** If one database has short codes and another has long codes, they might not match well in terms of length.
5. **Textual Data Matching (DI 5):**
   * **Idea:** Comparing the similarity of text.
   * **Example:** If one database has descriptions like "great product" and another has "excellent item," we could use methods like cosine similarity to find their similarity.
6. **Schema Element Name as Value (DI 6):**
   * **Idea:** Using the names of elements as values for matching.
   * **Example:** If one database has "Economy" and "Business" as values for the attribute "Ticket Class," and another uses "Economy" and "Business" as attributes, we could consider them as matches.
7. **Matching Simple and Composite Domains (DI 7):**
   * **Idea:** Comparing simple domains with each part of a complex domain.
   * **Example:** If one database has a simple domain like "color," and another has a composite domain like "Dell desktop PC," we would need to compare the simple domain ("color") with each part of the composite domain.

**Challenges:**

* **Complex Domains:** Splitting complex domains, like "Dell desktop PC 1.5GHz 1GB RAM 30GB disk space," can be tricky without extra information.

**Combining Similarities**

Let us call a program that assesses the similarity of a pair of elements from two different schemas based on a particular match criterion a matcher. It is typically the case that the more indicators we have the better results we can achieve, because different matchers have their own advantages and also shortcomings. Combining schema-level and instance-level approach will produce better results than each type of approaches alone. This combination can be done in various ways.



Imagine you have two computer programs, Program A and Program B. Each program has its own way of comparing elements from different datasets (schemas). We call these comparison methods "matchers."

Now, you want to combine the results from these matchers to get an overall similarity score for pairs of elements from Program A's schema (S1) and Program B's schema (S2).

Here are three ways to combine these scores:

1. **Max Strategy:**
   * **Idea:** Take the highest similarity score from any matcher.
   * **Example:** If Matcher 1 says the similarity is 0.8, and Matcher 2 says 0.6, then the combined similarity (CSim) using the max strategy would be 0.8.
2. **Weighted Sum Strategy:**
   * **Idea:** Assign weights to each matcher based on its importance. Combine scores using these weights.
   * **Example:** If Matcher 1 has a weight of 0.3 and gives a similarity score of 0.8, and Matcher 2 has a weight of 0.7 and gives a score of 0.6, then the combined similarity (CSim) would be 0.3 \* 0.8 + 0.7 \* 0.6 = 0.72.
3. **Weighted Average Strategy:**
   * **Idea:** Similar to the weighted sum but normalized by the number of matchers.
   * **Example:** If Matcher 1 and Matcher 2 both have weights of 0.5 and give similarity scores of 0.8 and 0.6 respectively, then the combined similarity (CSim) would be (0.5 \* 0.8 + 0.5 \* 0.6) / 2 = 0.7.

**Integration of Web Query Interfaces**

The preceding discussions are generic to database integration and Web data integration. In this and the next sections, we focus on integration in the Web context. The Web consists of the surface Web and the deep Web. The surface Web can be browsed using any Web browser, while the deep Web consists of databases that can only be accessed through parameterized query interfaces. With the rapid expansion of the Web, there are now a huge number of deep web data sources. In almost any domain, one can find a large number of them, which are hosted by e-commerce sites. Each of such sources usually has a keyword based search engine or a query interface that allows the user to fill in some information in order to retrieve the needed data.

1. **Surface Web vs. Deep Web:**
   * **Surface Web (Library Shelves):**
     + Imagine the surface web like the shelves in a library. You can easily walk around, browse, and pick up any book you see.
     + These are the websites you can access with any web browser – straightforward and visible.
   * **Deep Web (Library Catalog):**
     + Now, think of the deep web as the library's catalog. It's not like the shelves; you can't directly pick up books from there.
     + Instead, you need to use the catalog, fill in specific details like the book title, author, or genre, and the catalog guides you to the exact location of the book on the shelves.
     + Similarly, deep web sources require users to input specific information to retrieve the data they need.
2. **Integration Goal (Librarian's Desk):**
   * **Combining Multiple Interfaces:**
     + Picture the librarian's desk as the global interface. You, as a user, only need to interact with the librarian to find books.
     + Instead of going to each shelf (interface) individually, the librarian (system) helps you by searching all the shelves for the information you want.
   * **User-Friendly Process:**
     + You provide your request to the librarian (global interface), and they take care of searching through the catalog (query interfaces) and fetching the relevant books (data) from the shelves.
     + This way, you don't have to manually go to each shelf – the librarian streamlines the process for you.
3. **Query Interface vs. Database Schema (Library Catalog vs. Book Index):**
   * **Traditional Databases (Book Index):**
     + In a traditional database, think of the book index. It provides an organized list of titles, authors, and categories – a schema that helps you understand how the books are categorized.
   * **Web Query Interfaces (Library Catalog):**
     + Now, envision the library catalog as a web query interface. It's not just a list; it allows you to search for specific books based on various criteria like author, genre, or publication date.
     + Web interfaces are more dynamic than traditional database schemas.
   * **Schema Model for Query Interfaces:**
     + To organize the catalog effectively, the librarian creates a schema model – a structure defining how the information is categorized and accessed.
     + Similarly, for web query interfaces, there's a schema model that guides how the information is organized for user queries.

Since query interfaces are different from traditional database schemas, we first define a schema model.

1. Limited use of acronyms and abbreviations: Data displayed in Web pages are for the general public to view and must be easy to understand. Hence, the use of acronyms and abbreviations is limited to those very obvious ones. Enterprise-specific acronyms and abbreviations seldom appear. In the case of a company database, abbreviations are frequently used, which are often hard to understand by human users and difficult to analyze by automated systems. To a certain extent, this feature makes information integration on the Web easier.

2. Limited vocabulary: In the same domain, there are usually a limited number of essential attributes that describe each object. For example, in the book domain, we have the title, author, publisher, ISBN number, etc. For each attribute, there is usually limited ways to express the attribute. The chosen label (describing a data attribute, e.g., “departure city”) needs to be short, and easily understood by the general public. Therefore, there are not many ways to express the same attributes. Limited vocabulary also makes statistical approaches possible.

3. A large number of similar databases: There are often a large number of sites that offer the same services or sell the same products, which result in a large number of query interfaces and make it possible to use statistical methods. This is not the case in a company because the number of related databases is small. Integration of databases from multiple companies seldom happens.

4. Additional structure: The attributes of a Web interface are usually organized in some meaningful ways. For example, related attributes are grouped and put together physically (e.g., “first name” and “last name” are usually next to each other), and there may also be a hierarchical organization of attributes. Such structures also help integration as we will see later. In the case of databases, attributes usually have no structure.

**Characteristics of Web Query Interfaces:**

**1. Limited Use of Acronyms and Abbreviations:**

**Explanation:**

* Web pages are designed for everyone, so they avoid using confusing acronyms.
* Minimizing abbreviations makes it easier for both human users and automated systems to understand the content.

**Example:**

* Instead of using "CEO" (Chief Executive Officer), a web page might say "Head of the Company" for broader understanding.

**2. Limited Vocabulary:**

**Explanation:**

* Each domain (topic or field) has a small set of essential attributes or key terms.
* Attributes are expressed in a short and easily understood way to ensure clarity.
* Having a limited vocabulary allows for statistical analysis because terms are consistent.

**Example:**

* In a movie database, essential attributes could include "Title," "Director," and "Release Year," using straightforward and commonly known terms.

**3. Large Number of Similar Databases:**

**Explanation:**

* Many websites offer similar services or sell the same products, resulting in numerous query interfaces with slight variations.
* Statistical methods can be applied to identify patterns across these similar databases.

**Example:**

* Various e-commerce websites sell smartphones, each with its own query interface. Despite differences, they share common attributes like "Brand," "Model," and "Price," facilitating statistical analysis for integration.

**4. Additional Structure:**

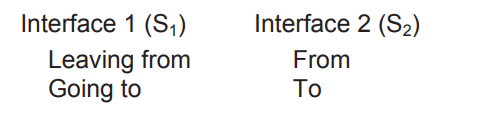
**Explanation:**

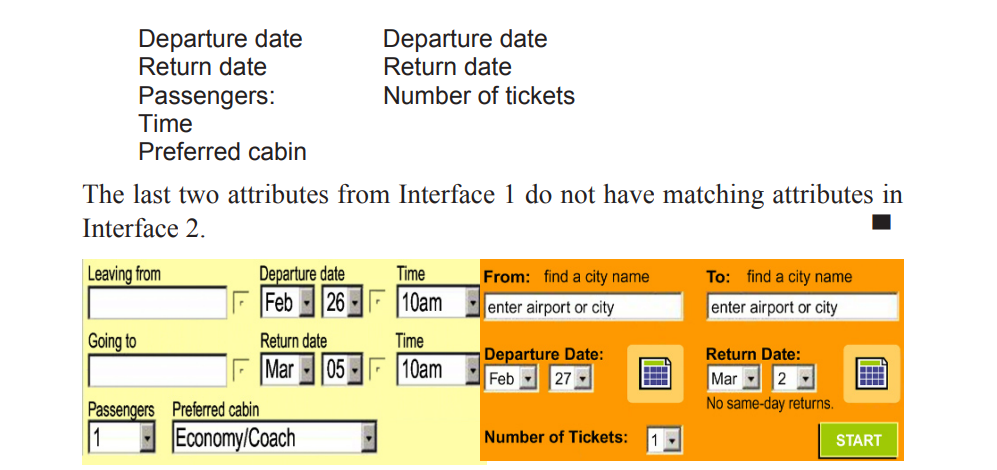
* Web interfaces organize attributes in meaningful ways, grouping related information logically.
* Attributes may follow a hierarchy, enhancing the structure and aiding integration efforts.

**Example:**

* In an online travel website, attributes related to flights may be organized under "Departure" and "Arrival," creating a clear structure for users. This structure facilitates integration because similar attributes are grouped together.

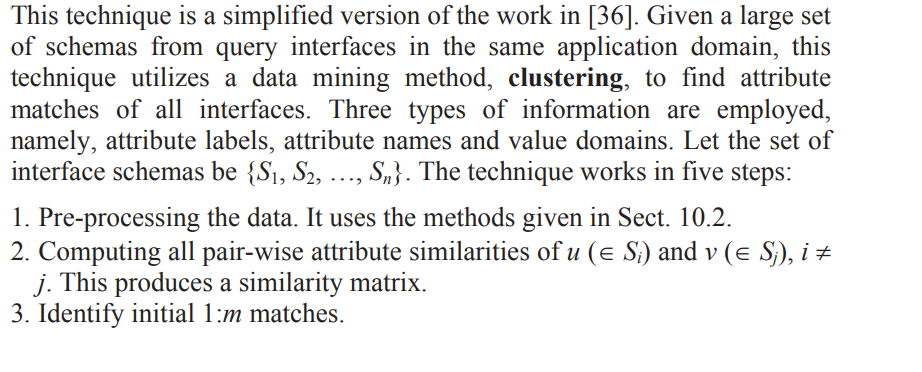
Due to these differences, schema matching of query interfaces can exploit new methods. For example, data mining techniques can be employed as we will see in the next few sub-sections. Traditional schema matching approaches in the database context are usually based on pair-wise matching. Similar to schema integration, query interface integration also requires mapping of corresponding attributes of all the query interfaces.

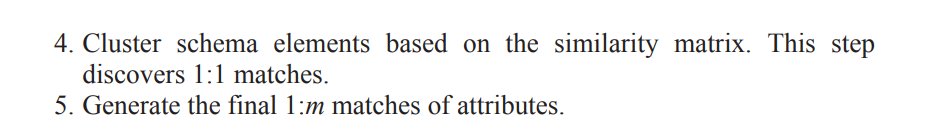




The problem of generating the mapping is basically the problem of identifying synonyms in the application domain. However, it is important to note that the synonyms here are domain dependent. A general-purpose semantic lexicon such as WordNet or any thesaurus is not sufficient for the identification of most domain-specific synonyms. For example, it is difficult to infer from WordNet or any thesaurus that “Passengers” is synonymous to “Number of tickets” in the context of airline ticket reservation. Domain-specific lexicons are not generally available as they are expensive to build. In this section, we discuss three query interface matching techniques. We also describe a method for building a global interface.

**A Clustering Based Approach**





**Shopping for Attributes**

1. **Preparing for Shopping (Pre-processing):**
   * Before you start shopping, you make a list of items you need. Similarly, in the digital world, the system prepares a list of attributes from different websites.
2. **Finding Similar Items (Computing Pair-Wise Similarities):**
   * Imagine you're looking for similar items in different stores. The system compares attributes like product labels, names, and types (e.g., color or size) to find similarities.
   * If you're shopping for shoes, it checks how similar the labels and names are, and whether they share similar characteristics like being red or having a particular size.
3. **Grouping Items Together (Identifying Initial 1:m Matches):**
   * When shopping, you notice that some items go well together, like shoes and shoelaces. Similarly, the system identifies initial matches based on the layout and proximity of attributes.
   * For instance, if there are attributes that are physically close and have similar labels, they might be grouped together.
4. **Organizing Your Shopping Cart (Cluster Schema Elements):**
   * Think of your shopping cart as a cluster. The system organizes similar items into clusters using a hierarchical approach, like grouping all shoes together.
   * This helps in finding 1:1 matches – for instance, pairing specific shoes with their corresponding shoelaces.
5. **Completing Your Shopping List (Generate Final 1:m Matches):**
   * After you've filled your cart, you might realize you need matching accessories. Similarly, the system looks for additional 1:m matches, considering the relationships discovered during clustering.
   * If shoes are related to socks, and socks are related to a specific brand, the system connects the dots to complete the shopping list.

**A Correlation Based Approach**

This technique also makes use of a large number of interfaces. The approach is based on co-occurrences of schema attributes and the following observations:

1. In an interface, some attributes may be grouped together to form a bigger concept. For example, “first name” and “last name” compose the name of a person. This is called the grouping relationship, denoted by a set, e.g., {first name, last name}. Attributes in such a group often cooccur in schemas, i.e., they are positively correlated.

2. An attribute group rarely co-occurs in schemas with their synonym attribute groups. For example, “first name” and “last name” rarely cooccur with name in the same query interface. Thus, {first name, last name} and {name} are negatively correlated. They represent 2:1 match. Note that a group may contain only one attribute.

1. **Preparing for the Meal (Pre-processing):**
   * Before going to a restaurant, you check the menu to decide what you want to order. Similarly, the system prepares a list of items from different restaurant menus.
2. **Grouping Related Dishes (Group Discovery):**
   * In a restaurant, you notice that some dishes naturally go together, like a burger and fries. Similarly, the system looks for groups of attributes that often appear together in different menus.
   * For instance, it identifies that "first name" and "last name" are often grouped together as part of a person's name.
3. **Checking Compatibility Between Menus (Match Discovery):**
   * Imagine you find that certain groups of dishes in one restaurant rarely appear with similar groups in another. Similarly, the system checks if attribute groups positively or negatively correlate between different interfaces.
   * For example, it notices that "first name" and "last name" rarely appear with a broader category like "name" in the same query interface, indicating a potential match.
4. **Selecting the Best Combos (Matching Selection):**
   * In the restaurant scenario, if you find two combos with overlapping items, you want the one with the most negative correlation to avoid redundancy. Similarly, the system selects matches that are most negatively correlated to avoid conflicts.
   * It uses a scoring system based on the negative correlation values to rank matches and resolve conflicts. The goal is to find the most confident and consistent matches.

**An Instance Based Approach**

It matches query interfaces and also the query results. It assumes that:

1. a global schema (GS) for the application domain is given, which represents the key attributes of the domain, and

2. a number of sample data instances under the domain global schema are also available.

1. **Library Schema (Global Schema):**
   * Imagine you have a master catalog (global schema) listing key details like book titles, authors, publishers, ISBNs, publication dates, and formats.
2. **Library Search System (Query Interface):**
   * When you go to the library, you use a search system (query interface) where you input information like the book title, author, publisher, keywords, or ISBN to find a book.
3. **Book Search Results (Result Schema):**
   * When you search for a book, the system returns a list of books (result schema) matching your query, with details like title, author, publisher, and ISBN.

### 4. Key Observation - Reappearing Words:

When you search for something, like "Harry Potter," the words from your search often show up a lot in the details of the results. For example, if you search by the author's name with "Harry Potter," you might not get as many matches as when you search by the book title.

### 5. Creating an Occurrence Matrix:

To keep track of how often these words show up, the system takes real examples (actual books) and puts their details into different parts of the search tool. It looks at how many times the words you searched for show up in the results, depending on which parts of the search tool were used.

### 6. Counting Occurrences - 3D Matrix:

All this information gets organized into a 3D matrix. Imagine a big table where each cell adds up how often the words in your search appear when you use different parts of the search tool – like searching by author, title, etc.

### 7. Matching Schemas Based on Mutual Information:

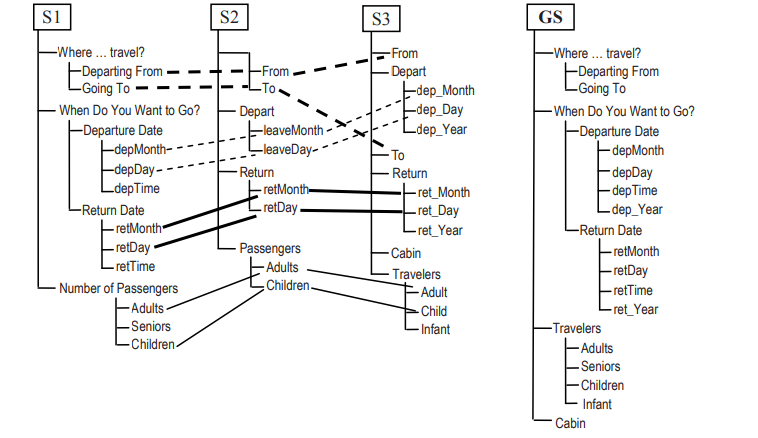
Now, the system looks at how much the different parts of the search tool depend on each other. It calculates something called "mutual information." The higher the mutual information, the stronger the connection between parts. It then picks the pairs (combinations) that have the strongest connection, creating matches between the different parts of the search tool and the overall database structure.

**Constructing a Unified Global Query Interface**

Once a set of query interfaces in the same domain is matched, we can automatically construct a well-designed global query interface that contains all the (or the most significant) distinct attributes of all source interfaces. To build a “good” global interface, three requirements are identified in

**Structural Appropriateness and the Merge Algorithm**

Structural appropriateness means to satisfy grouping constraints and ancestor-descendant relationship constraints of the attributes in individual interfaces. These constraints guide the merging algorithm to produce the global interface, which has one attribute for each cluster



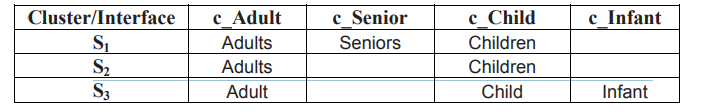
Grouping Constraints: Recall that semantically related attributes within an interface are usually grouped together. Grouping constraints require that these attributes should also appear together in the global interface.

As the global interface has an attribute for each cluster, the problem is to partition all the clusters into semantically meaningful subsets (or groups), which are employed to organize attributes in the global interface. For instance, the following sets of clusters are produced, {c\_deptCity, c\_destCity}, {c\_deptYear, c\_deptTime, c\_deptDay, c\_depMonth}, and {c\_Senior, c\_Adult, c\_Child, c\_Infant}, where c\_X is a cluster representing X (e.g., c\_deptCity and c\_destCity are clusters representing departure cities and destination cities, respectively).

**Lexical Appropriateness**

After the interfaces are merged, the attributes in the integrated interface need to be labeled so that the labels of the attributes within a group are consistent and the labels of the internal nodes are consistent with respect to themselves and to the leaf nodes .

For instance, “Adults”, “Seniors” and “Children” are all plurals, whereas “Leaving” and “Returning” are gerunds. Ideally, the groups within the global interface should have the same uniformity property. Since the attributes may be from different interfaces, a group of attributes within the unified interface might not correspond to any group in a single interface, which makes it hard to assign consistent labels. To deal with the problem, a strategy called intersect-and-union is used, which finds groups with non-empty intersection from different interfaces and then unions them.



Notice that by combining the labels given by S1 and S2 a consistent naming assignment, namely, “Seniors”, “Adults” and “Children”, can be achieved because the two sets share labels (i.e., “Adults” and “Children”) that are consistent with the labels in both sets. This strategy can be iteratively applied until a label is assigned to each attribute in the group. To deal with minor variations, more relaxed rules for combining attribute labels can be used, e.g., requiring that the set of tokens of the labels to be equal after removal of stopwords (e.g., “Number of Adults” has the same set of tokens as “Adults Number”, i.e. {Number, Adults}) and stemming. If a consistent solution for the entire group cannot be found, consistent solutions for subsets of attributes are constructed.

**Instance Appropriateness**

Finally, we discuss how to determine the domain for each attribute in the global schema (interface). A domain has two aspects: the type and the set of values. To determine the domain type of a global attribute, compatibility rules are needed . For instance, if all attributes in a cluster have a finite (infinite) domain then the global attribute will have a finite (infinite) domain. If in the cluster there are both finite and infinite domains, then the domain of the global attribute will be hybrid (i.e., users can either select from a list of pre-compiled values or fill in a new value). As a case in point, the “Adults” attribute on the global interface derived from the two interface will have a finite domain, whereas the attribute “Going to” will have a hybrid domain.

**The Problem of Opinion Mining**

In this first section, we define an abstraction of the opinion mining problem. It enables us to see a structure from the complex and intimidating unstructured text. Moreover, for most opinion-based applications, it is essential to analyze a collection of opinions rather than only one because one opinion represents only the view of a single person, which is usually not sufficient for action. This indicates that some form of summary of opinions is needed.

-Opinion mining, also known as sentiment analysis, is the process of extracting subjective information from text data, such as reviews, tweets, and social media posts. The main problem with opinion mining is the subjectivity of language, which makes it difficult to accurately classify text data as positive, negative, or neutral.

-There are several challenges associated with opinion mining. First, people express opinions in many different ways, using idiomatic expressions, sarcasm, irony, and other linguistic devices that can be difficult to interpret. Second, context plays a crucial role in determining the sentiment of a text. For example, the same sentence can be positive in one context and negative in another. Third, there is a high degree of variability in people's opinions, making it difficult to establish a standard for what constitutes positive, negative, or neutral sentiment.

-Despite these advances, opinion mining remains a challenging problem, as language is constantly evolving and new linguistic devices are being developed all the time. As such, researchers in this field must continually refine their methods and models to keep pace with the ever-changing landscape of language and opinion.

- The detection of spam and fake reviews, mainly through the identification of duplicates, the comparison of qualitative with summary reviews, the detection of outliers, and the reputation of the reviewer

- The limits of collaborative filtering, which tends to identify most popular concepts and to overlook most innovative / out of the box thinking - the risk of a filter bubble, where automated content analysis combined with behavioural analysis leads to a very effective but ultimately deviating selection of relevant opinions and content, so that the user is not aware of content which is somehow different from his expectations

- The asymmetry in availability of opinion mining software, which can currently be afforded only by organisations and government, but not by citizens. In other words, government have the means today to monitor public opinion in ways that are not available to the average citizens. While content production and publication has democratized, content analysis has not.

- The integration of opinion with behaviour and implicit data, in order to validate and provide further analysis into the data beyond opinion expressed

- The continuous need for better usability and user-friendliness of the tools, which are currently usable mainly by data analysts

**Problem Definitions We use the following review segment on iPhone to introduce the problem (an id number is associated with each sentence for easy reference):**

“(1) I bought an iPhone a few days ago. (2) It was such a nice phone. (3) The touch screen was really cool. (4) The voice quality was clear too. (5) However, my mother was mad with me as I did not tell her before I bought it. (6) She also thought the phone was too expensive, and wanted me to return it to the shop. … ”

The question is: what we want to mine or extract from this review? The first thing that we notice is that there are several opinions in this review.

Sentences (2), (3), and (4) express some positive opinions, while sentences (5) and (6) express negative opinions or emotions. Then we also notice that the opinions all have some targets. The target of the opinion in sentence (2) is the iPhone as a whole, and the targets of the opinions in sentences (3) and (4) are “touch screen” and “voice quality” of the iPhone, respectively. The target of the opinion in sentence (6) is the price of the iPhone, but the target of the opinion/emotion in sentence (5) is “me”, not iPhone. Finally, we may also notice the holders of opinions. The holder of the opinions in sentences (2), (3), and (4) is the author of the review (“I”), but in sentences (5) and (6) it is “my mother.” With this example in mind, we now formally define the opinion mining problem. We start with the opinion target.

**Document sentiment classification**

Document sentiment classification is the task of analyzing a piece of text, such as a document or a tweet, and determining the overall sentiment expressed within it. This is typically done by assigning a label to the text that reflects its sentiment, such as positive, negative, or neutral.

**Classification Based on Supervised Learning**

Sentiment classification obviously can be formulated as a supervised learning problem with three classes, positive, negative, and neutral. Training and testing data used in the existing research are mostly product reviews, which is not surprising due to the above assumption. Since each review already has a reviewer-assigned rating (e.g., 1–5 stars), training and testing data are readily available. For example, a review with 4 or 5 stars is considered a positive review, a review with 1 or 2 stars is considered a negative review and a review with 3 stars is considered a neutral review.

Sentiment classification is similar to but also somewhat different from classic topic-based text classification, which classifies documents into predefined topic classes, e.g., politics, sciences, sports, etc. In topic-based classification, topic-related words are important. However, in sentiment classification, topic-related words are unimportant. Instead, opinion words (also called sentiment words) that indicate positive or negative opinions are important, e.g., great, excellent, amazing, horrible, bad, worst, etc

Terms and their frequency. These features are individual words or word ngrams and their frequency counts (they are also commonly used in traditional topic-based text classification). In some cases, word positions may also be considered. The TF-IDF weighting scheme from information retrieval may be applied too. These features have been shown quite effective in sentiment classification.

Part of speech. It was found in many researches that adjectives are important indicators of opinions. Thus, adjectives have been treated as special features.

**Opinion words and phrases**. Opinion words are words that are commonly used to express positive or negative sentiments. For example, beautiful, wonderful, good, and amazing are positive opinion words, and bad, poor, and terrible are negative opinion words.

**Rules of opinions**. Although opinion words and phrases are important, there are also many other expressions that contain no opinion words or phrases but indicate opinions or sentiments.

**Syntactic dependency**. Words dependency-based features generated from parsing or dependency trees are also tried by several researchers.

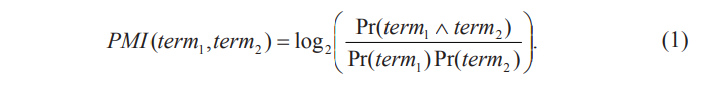
**Classification Based on Unsupervised Learning**

It is not hard to imagine that opinion words and phrases are the dominating indicators for sentiment classification. Thus, using unsupervised learning based on such words and phrases would be quite natural.It performs classification based on some fixed syntactic phrases that are likely to be used to express opinions. The algorithm makes use of a natural language processing technique called part-of-speech (POS) tagging. The part-of-speech of a word is a linguistic category that is defined by its syntactic or morphological behaviour. Common POS categories in English grammar are: noun, verb, adjective, adverb, pronoun, preposition, conjunction, and interjection. Then, there are many categories which arise from different forms of these categories. For example, a verb can be a verb in its base form, in its past tense, etc

Step 1: It extracts phrases containing adjectives or adverbs as adjectives and adverbs are good indicators of opinions. However, although an isolated adjective may indicate opinion, there may be insufficient context to determine its opinion orientation

For example, the adjective “unpredictable” may have a negative orientation in an automotive review, in a phrase such as “unpredictable steering,” but it could have a positive orientation in a movie review, in a phrase such as “unpredictable plot.” Therefore, the algorithm extracts two consecutive words, where one member of the pair is an adjective or adverb, and the other is a context word.

Step 2: It estimates the semantic orientation of the extracted phrases using the pointwise mutual information (PMI) measure given in Equation (1):

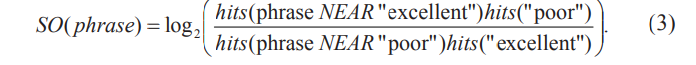


Here, Pr(term1 ∧ term2) is the co-occurrence probability of term1 and term2, and Pr(term1)Pr(term2) gives the probability that the two terms co-occur if they are statistically independent. The ratio between Pr(term1 ∧ term2) and Pr(term1)Pr(term2) is thus a measure of the degree of statistical dependence between them. The log of this ratio is the amount of information that we acquire about the presence of one of the words when we observe the other.

The semantic/opinion orientation (SO) of a phrase is computed based on its association with the positive reference word “excellent” and its association with the negative reference word “poor”:

SO(phrase) = PMI(phrase, “excellent”) − PMI(phrase, “poor”).

(2) The probabilities are calculated by issuing queries to a search engine and collecting the number of hits. For each search query, a search engine usually gives the number of relevant documents to the query, which is the number of hits. Thus, by searching the two terms together and separately, we can estimate the probabilities in Equation (1). Turney, the author of, used the AltaVista search engine because it has a NEAR operator, which constrains the search to documents that contain the words within ten words of one another in either order. Let hits(query) be the number of hits returned. Equation (2) can be rewritten as:



To avoid division by zero, 0.01 is added to the hits.

Step 3: Given a review, the algorithm computes the average SO of all phrases in the review and classifies the review as recommended if the average SO is positive, not recommended otherwise. Final classification accuracies on reviews from various domains range from 84% for automobile reviews to 66% for movie reviews.

**Sentence Subjectivity and Sentiment Classification**

Naturally the same document-level sentiment classification techniques can also be applied to individual sentences. The task of classifying a sentence as subjective or objective is often called subjectivity classification. The resulting subjective sentences are also classified as expressing positive or negative opinions, which is called sentence-level sentiment classification.

Problem Definition: Given a sentence s, two sub-tasks are performed:

1. Subjectivity classification. Determine whether s is a subjective sentence or an objective sentence

2. Sentence-level sentiment classification. If s is subjective, determine whether it expresses a positive, negative, or neutral opinion.

1. **Purpose of Sentence-level Classification**:
   * Sentence-level classification aims to identify opinions expressed in text. It serves two main purposes:
     + Filtering out sentences containing opinions from those that do not.
     + Determining whether the opinions expressed are positive or negative.
2. **Classification Problems**:
   * There are two main classification problems:
     + Subjectivity classification: Identifying whether a sentence expresses a subjective opinion.
     + Sentiment classification: Determining whether the subjective opinion expressed is positive or negative.
3. **Traditional Learning Methods**:
   * Supervised learning methods, such as the naïve Bayesian classifier, are commonly used for both subjectivity and sentiment classification.
4. **Bootstrapping Approach**:
   * An algorithm automates the labeling of training data by using high precision classifiers to identify subjective and objective sentences.
   * These classifiers use lexical items (words or n-grams) as clues to determine subjectivity.
   * Extracted sentences are added to the training data to learn patterns, which are then used to identify more subjective and objective sentences in subsequent iterations.
5. **Pattern Learning**:
   * Syntactic templates are provided to guide the learning of patterns.
   * Examples include passive and active verbs, nouns, auxiliary verbs, etc.
6. **Assumption of Sentence-level Sentiment Classification**:
   * Assumes that each sentence expresses a single opinion from a single opinion holder.
   * This assumption is suitable for simple sentences but may not hold for compound or complex sentences.
7. **Examples of Mixed Opinions**:
   * Compound or complex sentences may express multiple opinions.
   * Example: "The picture quality of this camera is amazing and so is the battery life, but the viewfinder is too small for such a great camera."
     + Positive opinions: Picture quality, battery life, camera as a whole.
     + Negative opinion: Viewfinder.
8. **Advanced Approaches**:
   * Studies use supervised learning and methods like log-likelihood ratio to classify sentiments.
   * Consider contextual sentiment influencers such as negation and contrary words.
9. **Importance of Identifying Opinions in Both Subjective and Objective Sentences**:
   * Not all subjective sentences contain opinions, and some objective sentences may imply opinions.
   * To effectively mine opinions, both types of sentences need to be considered.

**Opinion Lexicon Expansion**

In the preceding sections, we mentioned that opinion words are employed in many sentiment classification tasks. We now discuss how such words are generated. In the research literature, opinion words are also known as polar words, opinion-bearing words, and sentiment words. Positive opinion words are used to express some desired states while negative opinion words are used to express some undesired states. Examples of positive opinion words are beautiful, wonderful, good, and amazing. Examples of negative opinion words are bad, poor, and terrible. Apart from individual words, there are also opinion phrases and idioms, e.g., cost someone an arm and a leg. Collectively, they are called the opinion lexicon. They are instrumental for opinion mining for obvious reasons.

Opinion words can, in fact, be divided into two types, the base type and the comparative type. All the examples above are of the base type. Opinion words of the comparative type are used to express comparative and superlative opinions. Examples of such words are better, worse, best, worst, etc., which are comparative and superlative forms of their base adjectives or adverbs, e.g., good and bad. Unlike opinion words of the base type, the words of the comparative type do not express a direct opinion on an entity but a comparative opinion on more than one entity, e.g., “Car-x is better than Car-y.” This sentence tells us something quite interesting. It does not express an opinion that any of the two cars is good or bad. It just says that compared to Car-y, Car-x is better, and compared to Car-x, Car-y is worse. Thus, although we still can assign a comparative word as positive or negative based on whether it represents a desirable or undesirable state, we may not use it in the same way as an opinion word of the base type.

Dictionary-based approach: One of the simple techniques in this approach is based on bootstrapping using a small set of seed opinion words and an online dictionary, e.g., WordNet . The strategy is to first collect a small set of opinion words manually with known orientations and then to grow this set by searching in the WordNet for their synonyms and antonyms. The newly found words are added to the seed list. The next iteration starts. The iterative process stops when no more new words are found. This approach is used in [37, 55]. After the process completes, manual inspection can be carried out to remove and/or correct errors. Researchers have also used additional information (e.g., glosses) in WordNet and additional techniques (e.g., machine learning) to generate better lists.The dictionary-based approach and the opinion words collected from it have a major shortcoming. The approach is unable to find opinion words with domain and context-specific orientations, which is quite common. For example, for a speaker phone, if it is quiet, it is usually negative. However, for a car, if it is quiet, it is positive. The corpus-based approach can help deal with this problem.

### Corpus-based Approach and Sentiment Consistency:

#### **Step 1: Seed Opinion Adjectives**

We start with a small list of seed opinion adjectives:

* Positive: good, excellent, wonderful
* Negative: bad, terrible, awful

#### **Step 2: Identifying Opinion Words**

We use these seed words along with linguistic constraints to identify additional opinion words in a large corpus. One key constraint is the association between adjectives connected by conjunctions like "and" and "but." For example:

* "This car is beautiful and spacious." Here, "beautiful" and "spacious" are likely to share the same sentiment (positive).
* "This car is beautiful but difficult to drive." In this case, "beautiful" and "difficult" are likely to have different sentiments.

#### **Step 3: Building Sentiment Links**

We create links between adjectives based on their sentiment consistency. Adjectives with the same sentiment form same-orientation links, while those with different sentiments form different-orientation links. These links create a graph representing the sentiment relationships between adjectives.

#### **Step 4: Clustering**

We perform clustering on the graph to group adjectives into two sets: positive and negative.

#### **Example:**

Consider the following sentences:

1. "This movie is entertaining and engaging."
   * "Entertaining" and "engaging" are likely to have the same positive sentiment.
2. "The movie was entertaining but predictable."
   * "Entertaining" and "predictable" are likely to have different sentiments.

**Aspect-Based Opinion Mining**

Although classifying opinionated texts at the document level or at the sentence level is useful in many cases, it does not provide the necessary detail needed for many other applications. A positive opinionated document about a particular entity does not mean that the author has positive opinions on all aspects of the entity. Likewise, a negative opinionated document does not mean that the author dislikes everything. In a typical opinionated document, the author writes both positive and negative aspects of the entity, although the general sentiment on the entity may be positive or negative. Document and sentence sentiment classification does not provide such information.

1. Aspect extraction: Extract aspects that have been evaluated. For example, in the sentence, “The picture quality of this camera is amazing,” the aspect is “picture quality” of the entity represented by “this camera.”

Note that “this camera” does not indicate the GENERAL aspect because the evaluation is not about the camera as a whole, but about its picture quality. However, the sentence “I love this camera” evaluates the camera as a whole, i.e., the GENERAL aspect of the entity represented by “this camera.” Bear in mind whenever we talk about an aspect, we must know which entity it belongs to. In our discussion below, we often omit the entity just for simplicity of presentation.

2. Aspect sentiment classification: Determine whether the opinions on different aspects are positive, negative, or neutral. In the first example above, the opinion on the “picture quality” aspect is positive, and in the second example, the opinion on the GENERAL aspect is also positive.

**Aspect Sentiment Classification**

Aspect sentiment classification is the task of determining the orientation of opinions expressed on each aspect in a sentence. This involves identifying whether the opinions expressed about specific aspects are positive, negative, or neutral. While sentence-level and clause-level sentiment classification methods are useful for this task, they face challenges such as handling mixed opinions and opinions requiring phrase-level analysis. Here, we describe a lexicon-based approach to aspect sentiment classification, which has shown promising performance.

**1. Mark opinion words and phrases:** The first step involves identifying and marking opinion words and phrases in the sentence. Each positive word is assigned a score of +1, while each negative word is assigned a score of -1. For example, in the sentence "The picture quality of this camera is not great, but the battery life is long," "great" is marked as [+1].

**Example:** Sentence: "The camera quality is amazing, but the battery drains quickly."  
Opinion Marking: "The camera quality is amazing[+1], but the battery drains quickly[-1]."

**2. Handle opinion shifters:** Opinion shifters, such as negation words, modal auxiliary verbs, and presuppositional items, can alter opinion orientations. Negation words like "not" change the polarity of opinions. For instance, "The camera is not great" changes the opinion from positive to negative.

**Example:** Sentence: "The camera is not great, but it's functional."  
Opinion Handling: "The camera is not great[-1], but it's functional[+1]."

**3. Handle but-clauses:** Sentences containing contrasting conjunctions like "but" require special handling. The opinion orientations before and after "but" are opposite to each other if the opinion on one side cannot be determined.

**Example:** Sentence: "The camera quality is not great, but the battery life is impressive."  
Opinion Handling: "The camera quality is not great[-1], but the battery life is impressive[+1]."

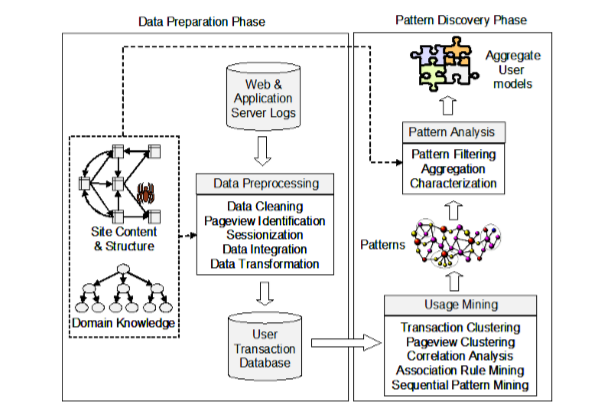
**4. Aggregating opinions:** Finally, an opinion aggregation function is applied to determine the final orientation of opinions on each aspect in the sentence. This function considers the distance between aspect and opinion words, assigning higher weights to words closer to the aspect.

**Example:** Sentence: "The camera quality is excellent, but the battery life is disappointing."  
Opinion Aggregation: For aspect "camera quality," the positive opinion word "excellent" has a higher weight than the negative opinion word "disappointing" for aspect "battery life."

This lexicon-based approach effectively handles aspect sentiment classification by considering opinion words, opinion shifters, but-clauses, and aggregating opinions based on their proximity to aspects. While it performs well in many cases, it may not cover all types of expressions conveying opinions, necessitating further refinement for comprehensive sentiment analysis.

**Web Usage Mining**

Web usage mining refers to the automatic discovery and analysis of patterns in clickstreams, user transactions and other associated data collected or generated as a result of user interactions with Web resources on one or more Web sites . The goal is to capture, model, and analyze the behavioral patterns and profiles of users interacting with a Web site. The discovered patterns are usually represented as collections of pages, objects, or resources that are frequently accessed or used by groups of users with common needs or interests. Following the standard data mining process , the overall Web usage mining process can be divided into three inter-dependent stages: data collection and pre-processing, pattern discovery, and pattern analysis. In the pre-processing stage, the clickstream data is cleaned and partitioned into a set of user transactions representing the activities of each user during different visits to the site. Other sources of knowledge such as the site content or structure, as well as semantic domain knowledge from site ontologies (such as product catalogs or concept hierarchies), may also be used in pre-processing or to enhance user transaction data. In the pattern discovery stage, statistical, database, and machine learning operations are performed to obtain hidden patterns reflecting the typical behavior of users, as well as summary statistics on Web resources, sessions, and users. In the final stage of the process, the discovered patterns and statistics are further processed, filtered, possibly resulting in aggregate user models that can be used as input to applications such as recommendation engines, visualization tools, and Web analytics and report generation tools. In the remainder of this chapter, we first provide a detailed examination of Web usage mining as a process, and discuss the relevant concepts and techniques commonly used in all the stages mentioned above. We then focus on the important problem of recommendation. It is followed by another special case of Web usage mining targeting search query logs, and known as query log mining (QLM). To complete our discussion of mining Web usage data, we finally introduce the new field of Ad click mining.



**Aspect-Based Opinion Mining**

Although classifying opinionated texts at the document level or at the sentence level is useful in many cases, it does not provide the necessary detail needed for many other applications. A positive opinionated document about a particular entity does not mean that the author has positive opinions on all aspects of the entity. Likewise, a negative opinionated document does not mean that the author dislikes everything. In a typical opinionated document, the author writes both positive and negative aspects of the entity, although the general sentiment on the entity may be positive or negative. Document and sentence sentiment classification does not provide such information. To obtain these details, we need to go to the aspect level.

1. Aspect extraction: Extract aspects that have been evaluated. For example, in the sentence, “The picture quality of this camera is amazing,” the aspect is “picture quality” of the entity represented by “this camera.”
2. Note that “this camera” does not indicate the GENERAL aspect because the evaluation is not about the camera as a whole, but about its picture quality. However, the sentence “I love this camera” evaluates the camera as a whole, i.e., the GENERAL aspect of the entity represented by “this camera.” Bear in mind whenever we talk about an aspect, we must know which entity it belongs to. In our discussion below, we often omit the entity just for simplicity of presentation. 2. Aspect sentiment classification: Determine whether the opinions on different aspects are positive, negative, or neutral. In the first example above, the opinion on the “picture quality” aspect is positive, and in the second example, the opinion on the GENERAL aspect is also positive.

**Discovery and Analysis of Web Usage Patterns**

The types and levels of analysis, performed on the integrated usage data, depend on the ultimate goals of the analyst and the desired outcomes. In this section we describe some of the most common types of pattern discovery and analysis techniques employed in the Web usage mining domain and discuss some of their applications.

The statistical analysis of pre-processed session data constitutes the most common form of analysis. In this case, data is aggregated by predetermined units such as days, sessions, visitors, or domains. Standard statistical techniques can be used on this data to gain knowledge about visitor behavior. This is the approach taken by most commercial tools available for Web log analysis. Reports based on this type of analysis may include information about most frequently accessed pages, average view time of a page, average length of a path through a site, common entry and exit points, and other aggregate measures. Despite a lack of depth in this type of analysis, the resulting knowledge can be potentially useful for improving the system performance, and providing support for marketing decisions. Furthermore, commercial Web analytics tools are increasingly incorporating a variety of data mining algorithms resulting in more sophisticated site and customer metrics.

Cluster Analysis and Visitor Segmentation

Clustering is a data mining technique that groups together a set of items having similar characteristics. In the usage domain, there are two kinds of interesting clusters that can be discovered: user clusters and page clusters. Clustering of user records (sessions or transactions) is one of the most commonly used analysis tasks in Web usage mining and Web analytics. Clustering of users tends to establish groups of users exhibiting similar browsing patterns. Such knowledge is especially useful for inferring user demographics in order to perform market segmentation in e-commerce applications or provide personalized Web content to the users with similar interests. Further analysis of user groups based on their demographic attributes (e.g., age, gender, income level, etc.) may lead to the discovery of valuable business intelligence.

Association and Correlation Analysis

Association rule discovery and statistical correlation analysis can find groups of items or pages that are commonly accessed or purchased together. This, in turn, enables Web sites to organize the site content more efficiently, or to provide effective cross-sale product recommendations. Most common approaches to association discovery are based on the Apriori algorithm. This algorithm finds groups of items (pageviews appearing in the preprocessed log) occurring frequently together in many transactions (i.e., satisfying a user specified minimum support threshold). Such groups of items are referred to as frequent itemsets. Association rules which satisfy a minimum confidence threshold are then generated from the frequent itemsets.

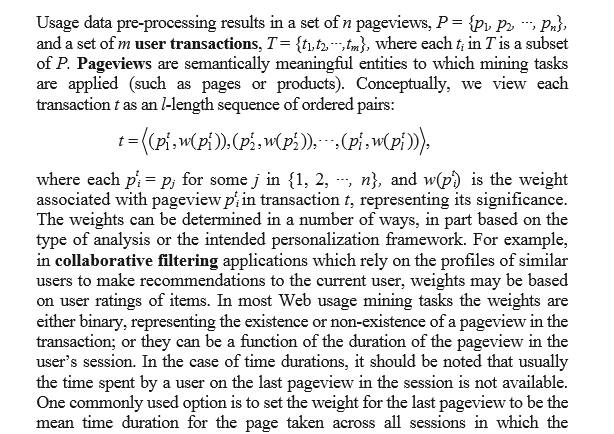
Analysis of Sequential and Navigational Patterns

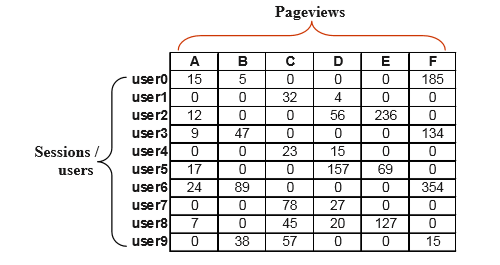
The technique of sequential pattern mining attempts to find inter-session patterns such that the presence of a set of items is followed by another item in a time-ordered set of sessions or episodes. By using this approach, Web marketers can predict future visit patterns which will be helpful in placing advertisements aimed at certain user groups. Other types of temporal analysis that can be performed on sequential patterns include trend analysis, change point detection, or similarity analysis. In the context of Web usage data, sequential pattern mining can be used to capture frequent navigational paths among user trails.

**Data Modelling for Web Usage Mining**

Usage data pre-processing results in a set of n pageviews, P = {p1, p2, ···, pn}, and a set of m user transactions, T = {t1,t2,···,tm}, where each ti in T is a subset of P. Pageviews are semantically meaningful entities to which mining tasks are applied (such as pages or products). Conceptually, we view each transaction t as an l-length sequence of ordered pairs: ,))(,(,)),(,()),(,( 2211 t l t l tttt p wppwppwpt " where each pti = pj for some j in {1, 2, ···, n}, and w(pt i) is the weight associated with pageview pti in transaction t, representing its significance. The weights can be determined in a number of ways, in part based on the type of analysis or the intended personalization framework. For example, in collaborative filtering applications which rely on the profiles of similar users to make recommendations to the current user, weights may be based on user ratings of items. In most Web usage mining tasks the weights are either binary, representing the existence or non-existence of a pageview in the transaction; or they can be a function of the duration of the pageview in the user’s session.

**Data Modeling for Web Usage Mining**





An example of a hypothetical u In this example, the weights for each pageview is the amount of time (e.g., in seconds) that a particular user spent on the pageview. In practice, these weights must be normalized to account for variances in viewing times by different users. It should also be noted that the weights may be composite or aggregate values in cases where the pageview represents a collection or sequence of pages and not a single page.

Given a set of transactions in the user-pageview matrix as described above, a variety of unsupervised learning techniques can be applied to obtain patterns. These techniques such as clustering of transactions (or sessions) can lead to the discovery of important user or visitor segments. Other techniques such as item (e.g., pageview) clustering and association or sequential pattern mining can find important relationships among items based on the navigational patterns of users in the site. As noted earlier, it is also possible to integrate other sources of knowledge, such as semantic information from the content of Web pages with the Web usage mining process. Generally, the textual features from the content of Web pages represent the underlying semantics of the site.

**Discovery and Analysis of Web Usage Patterns**

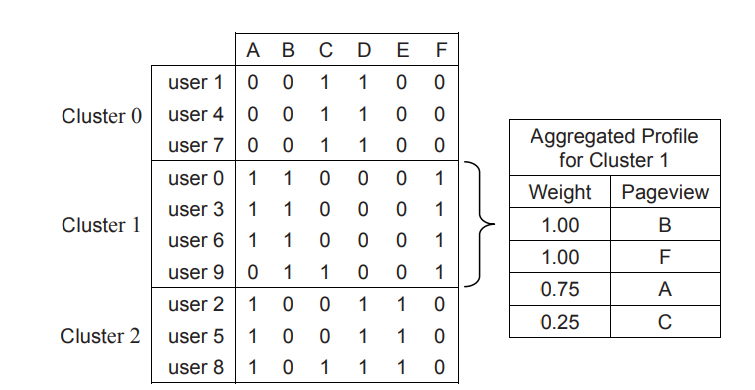
The types and levels of analysis, performed on the integrated usage data, depend on the ultimate goals of the analyst and the desired outcomes. In this section we describe some of the most common types of pattern discovery and analysis techniques employed in the Web usage mining domain and discuss some of their applications.

Session and Visitor Analysis

The statistical analysis of pre-processed session data constitutes the most common form of analysis. In this case, data is aggregated by predetermined units such as days, sessions, visitors, or domains. Standard statistical techniques can be used on this data to gain knowledge about visitor behavior. This is the approach taken by most commercial tools available for Web log analysis. Reports based on this type of analysis may include information about most frequently accessed pages, average view time of a page, average length of a path through a site, common entry and exit points, and other aggregate measures. Despite a lack of depth in this type of analysis, the resulting knowledge can be potentially useful for improving the system performance, and providing support for marketing decisions. Furthermore, commercial Web analytics tools are increasingly incorporating a variety of data mining algorithms resulting in more sophisticated site and customer metrics.

Cluster Analysis and Visitor Segmentation

Clustering is a data mining technique that groups together a set of items having similar characteristics. In the usage domain, there are two kinds of interesting clusters that can be discovered: user clusters and page clusters. Clustering of user records (sessions or transactions) is one of the most commonly used analysis tasks in Web usage mining and Web analytics. Clustering of users tends to establish groups of users exhibiting similar browsing patterns. Such knowledge is especially useful for inferring user demographics in order to perform market segmentation in e-commerce applications or provide personalized Web content to the users with similar interests. Further analysis of user groups based on their demographic attributes (e.g., age, gender, income level, etc.) may lead to the discovery of valuable business intelligence. Usage-based clustering has also been used to create Web-based “user communities” reflecting similar interests of groups of users and to learn user models that can be used to provide dynamic recommendations in Web personalization application.



**Recommender Systems and Collaborative Filtering**

Recommender systems are widely used on the Web for recommending products and services to users. Most e-commerce Web sites have such systems. These systems serve two important functions. First, they help users deal with the information overload by giving them personalized recommendations. For example, given thousands of movies, a recommender system selects and recommends some movies to each user that he/she will most likely enjoy watching. Second, they help businesses make more profits. Due to the problem rich nature and abundance of applications, numerous research papers have been published on recommender systems in the fields of computer science, information systems, marketing, and management science

1. **Users and Items:**
   * We have a group of users and a set of items to recommend to them.
   * Each user has a profile with characteristics like age, gender, preferences, etc.
   * Similarly, each item has its own set of features like title, genre, director (in the case of movies), etc.
2. **Utility Function (p):**
   * This function measures how useful an item is to a user.
   * It's represented as p:U×S → R, where U is the set of users, S is the set of items, and R is a totally ordered set (like integers or real numbers).
   * The goal is to learn this utility function.
3. **Objectives of Recommender Systems:**
   * **Learning the Utility Function:** Depending on the application, the goal could be to maximize user satisfaction or seller profitability.
   * **Using the Utility Function to Recommend Items:** Once the utility function is learned, the system predicts the usefulness of each item to each user and recommends the top items to them.
4. **Prediction Tasks:**
   * **Rating Prediction:** Predicting the rating a user might give to an unseen item (e.g., rating a movie).
   * **Item Prediction:** Predicting a ranked list of items a user might buy or use.
5. **Approaches to Recommendations:**
   * **Content-Based Recommendations:** Recommending items similar to ones the user has liked in the past.
   * **Collaborative Filtering:** Recommending items that users with similar tastes have liked.
6. **Hybrid Methods:**
   * These combine content-based and collaborative filtering approaches.
   * They can work by:
     + Combining predictions from both approaches.
     + Incorporating content-based characteristics into a collaborative approach, and vice versa.
     + Constructing a unified model that includes both approaches.

These approaches are widely used in recommendation systems to help users discover new items they might like based on their preferences and behavior.

1. **Content-based recommendations:**
   * This method recommends items based on their attributes and similarities to what the user has liked before.
   * For example, if you've enjoyed action movies in the past, a content-based recommendation system might suggest other action movies with similar themes, actors, or directors.
   * It focuses on the characteristics of the items themselves and tries to find items that match your past preferences.
2. **Collaborative filtering (or collaborative recommendations):**
   * This approach recommends items based on the preferences of other users who have similar tastes to you.
   * Instead of looking at the attributes of the items, it looks at the behavior and preferences of other users.
   * For instance, if many users who liked action movies also enjoyed certain thriller movies, the system might recommend those thrillers to you because it thinks you'll also enjoy them.
   * It relies on finding patterns in the behavior of a group of users to make recommendations to individuals.

* **Association Rules for Recommendation:**
  + When we talk about using association rules for recommendation, it means we're using patterns in user behavior to suggest items they might like.
  + Imagine each user's interactions with items (like clicking on a webpage or buying a product) as a transaction.
  + We can mine association rules from these transactions to predict what users might like next.
  + For instance, if many users who bought item X also bought item Y, we can use this pattern to recommend item Y to someone who bought item X.
  + The left side of an association rule represents what's already known (like item X), and the right side represents what's predicted (like item Y).
  + These rules are helpful in suggesting items to users based on what other users with similar behavior have done in the past.
* **Challenges and Solutions:**
  + One challenge is sparse data, where users only interact with a small fraction of available items. This makes it hard to find common patterns.
  + To tackle this, researchers have proposed methods like dimensionality reduction and collaborative filtering.
  + Dimensionality reduction techniques aim to simplify the data while preserving important information, but they might miss out on some useful items.
  + Collaborative filtering involves finding users with similar interests and making recommendations based on what they liked.
  + Another challenge is the search process, which can be time-consuming. Efficient data structures like trie-based graphs help speed up the recommendation process.
  + To ensure rare but important items are included in recommendations, some methods allow users to specify different support levels for different items. This way, we can capture patterns involving less common items.

Top of Form

* **Collaborative Filtering (CF):**
  + Collaborative Filtering predicts what a user might like based on the preferences of other users with similar tastes.
  + It uses past interactions of users with items (like ratings or purchases) to make recommendations.
  + CF doesn't focus on attributes of users or items but solely on their interactions.
* **k-Nearest Neighbor (kNN) Approach:**
  + kNN directly uses the entire user-item database without building a model beforehand.
  + It's divided into two phases: Neighborhood Formation and Recommendation.
  + **Neighborhood Formation Phase:**
    - Compares the activity of a target user with the historical records of other users to find the top k users with similar tastes.
    - Similarity can be based on ratings of items, access to similar content, or purchase of similar items.
    - Pearson’s correlation coefficient is commonly used to measure similarity between users.
  + **Recommendation Phase:**
    - Once similar users (neighbors) are identified, it predicts the preference of the target user for a specific item.
    - Prediction is based on weighted aggregation of ratings from similar users.
    - Formula for predicting the rating of an item for the target user is derived based on similarity and ratings of similar users.
    - High-rated items are recommended to the user based on predictions.
* **Challenges and Solutions:**
  + One problem is scalability, as comparing the target user to all user records in real-time can be slow.
  + Item-based collaborative filtering is a variation that pre-computes item-to-item similarity values, making it more scalable.
  + Dimensionality reduction techniques like Principal Component Analysis or Singular Value Decomposition are used to handle the high dimensionality of user-item matrices, making computations more feasible.

**Query Log Mining**

Query logs contain a historic record of the activities of the users of a search engine and, as such, contain a trove of hidden knowledge about the wisdom of the crowds. Query log data is considered a kind of Web usage data, thus making Query Log Mining (QLM) fall under the umbrella of Web usage mining. However, we will see that it is common to complement the query usage data with content and structure data, making some techniques from this emerging field embrace more than one aspect of Web mining, i.e., usage (queries, search results, and clicked documents), content (actual text contents of the queries, search results, and clicked documents), and structure (hyperlinks between search results and clicked documents).

Query logs contain a historic record of the activities of the users of a search engine and, as such, contain a trove of hidden knowledge about the wisdom of the crowds. Query log data is considered a kind of Web usage data, thus making Query Log Mining (QLM) fall under the umbrella of Web usage mining. However, we will see that it is common to complement the query usage data with content and structure data, making some techniques from this emerging field embrace more than one aspect of Web mining, i.e., usage (queries, search results, and clicked documents), content (actual text contents of the queries, search results, and clicked documents), and structure (hyperlinks between search results and clicked documents).

A typical search engine query log file consists of requests submitted by users, with one request per line, typically in the order of their arrival, as in the case of most Web server access logs. Hence, requests by the same user may end up dispersed among requests by other users. A typical request contains the following fields:

* Timestamp: It indicates when the query is submitted.
* IP address: This is usually the IP address of the client computer from which the query has originated. However, this is not the case for users who connect through ISPs or other proxies.
* User agent: It identifies the type and version number of the client browser.
* CookieID: This is a unique string that can identify a user more accurately than simply using the IP address since it can avoid the proxy and dynamic IP problems.
* Query: This is the text string submitted by the user, which consists of one or more query terms (also called keywords).
* Result list (also called ranking): the URLs returned by the search engine as the answer to the submitted query.
* Clicked URL: A subset of the above URLs that have been clicked by the user.

Query Log Mining Methods

Researchers have used all kinds of data mining methods for query log mining based on application needs. Supervised methods basically try to provide, based on the past and current query activities, a prediction in the form of a numerical score, a probability or a prediction in the form of a label out of a finite set of possibilities, e.g., whether an Ad is relevant to a search query or not , to which topic a given query belongs, whether a Web page is spam or not , or whether a given result page is more relevant than another result page .

On the other hand, unsupervised methods do not have the requirement of available prior labels. They rely solely on the hidden or latent information within the past query activity data to discover clusters of query activities or to assess semantic relatedness between different queries or different Web pages based on past searching activities.

**Supervised Learning Techniques** Since supervised QLM methods have a predictive or labeling aim, they have used a variety of supervised learning methods, e.g., query classification, and a new type of learning, called learning to rank.

**Unsupervised Learning Techniques** In general unsupervised methods have found their use for finding related queries, frequently asked queries, or query clusters. Most of these methods can be used to provide query recommendation or modification. Due to the rich data types, various sophisticated similarity or distance measures have been proposed for clustering and other methods.

**Collaborative Filtering: Using Association Rules**

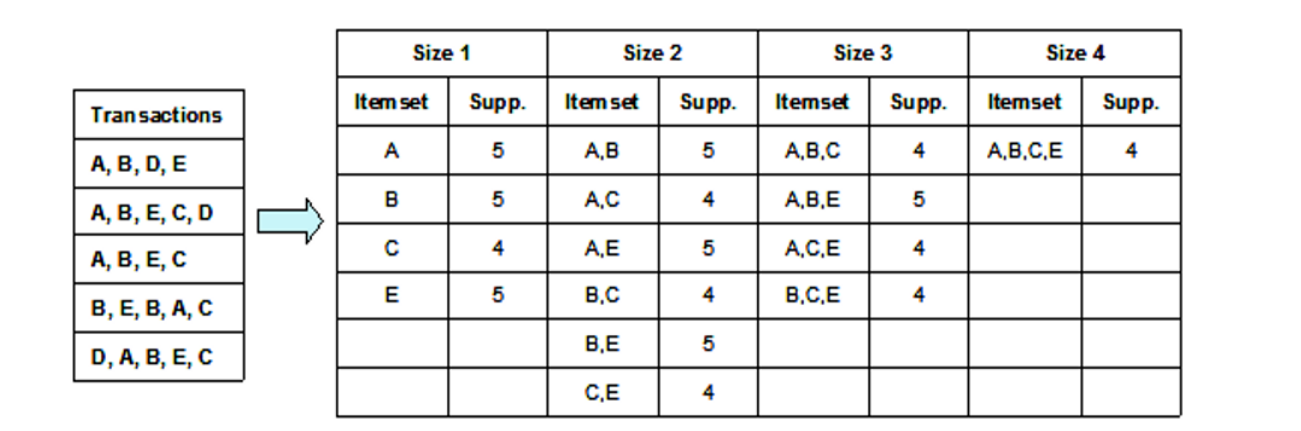
Recommender systems make information filtering for user by predicting user’s preference to items. Collaborative filtering is the most popular technique in implementing a recommender system. Association rule mining is a powerful data mining method to search for interesting relationships between items by finding the items frequently appeared together in a transaction database.

Using association rules for recommendation is quite natural as the items purchased by each user can naturally be treated as a transaction. Association rules can then be mined from the transactions of all users for prediction or classification. The left-hand side of a rule can be used to predict the right-hand-side of the rule. The approach has been used by several researchers and also commonly used in industry. For example, in the collaborative filtering context used association rules in the context of a top-N recommender system for e-commerce. The preferences of the target user are matched against the items on the lefthand side (or antecedent) X of each rule (e.g., X → Y), and the items right-hand side of the matching rules are sorted according to the confidence values. Then the top N items from this list are recommended to the target user. Note that in most applications, the set X contains one or more items, but the set Y contains only a single predicted item [recall that in general Y can contain any number of items as in the case of association rule mining.

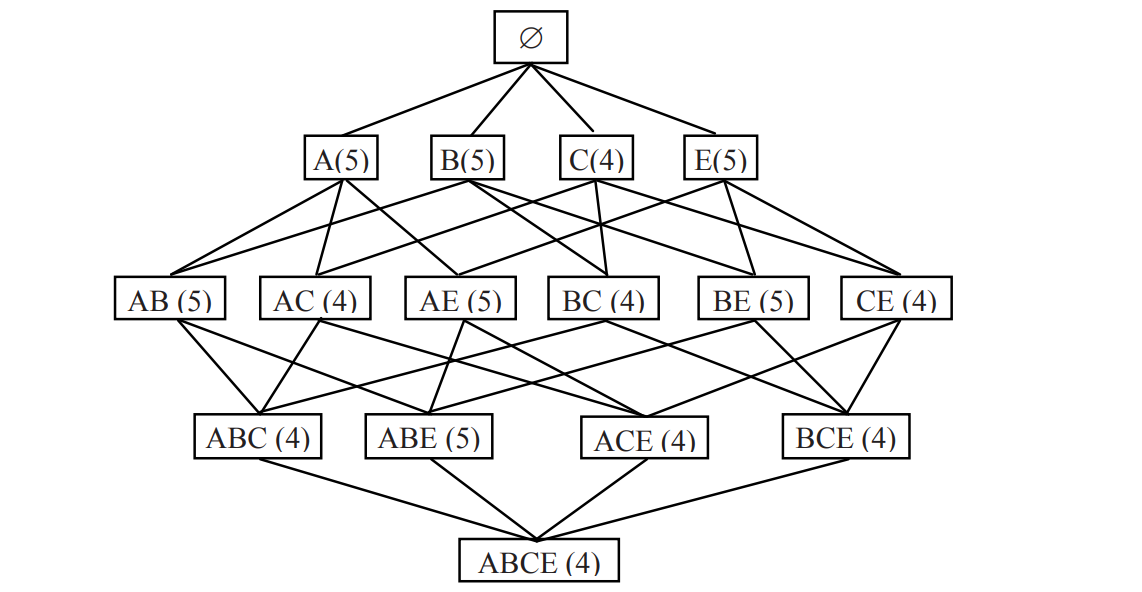
One problem for association rule recommendation systems is that a system has difficulty to give recommendations when the dataset is sparse, which is often the case in collaborative filtering applications. The reason for this sparsity is that any given user visits (or rates) only a very small fraction of the available items, and thus, it is often difficult to find a sufficient number of common items in multiple user profiles.

As an example, suppose that in a hypothetical Web site with user transaction data depicted in the left table. Using a minimum support (minsup) threshold of 4 (i.e., 80%), the Apriori algorithm discovers the frequent itemsets given in the right table. For each itemset, the support is also given.

A recommendation engine based on this framework matches the current user session window with the previously discovered frequent itemsets to find candidate items (pages) for recommendation. Given an active session window w and a group of frequent itemsets, the algorithm considers all the frequent itemsets of size |w| + 1 containing the current session window by performing a depth-first search of the Frequent Itemset Graph to level |w|. The recommendation value of each candidate is based on the confidence of the corresponding association rule whose consequent is the singleton containing the page to be recommended. If a match is found, then the children of the matching node n containing w are used to generate candidate recommendations. In practice, the window w can be incrementally decreased until a match is found with an itemset. For example, given user active session window , the recommendation algorithm, using the graph.

****

finds items A and C as candidate recommendations. The recommendation scores of item A and C are 1 and 4/5, corresponding to the confidences of the rules, B, E → A and B, E → C, respectively.

****

**Online advertising**

Online advertising is a multi-billion dollar industry with big financial stakes and advanced computational challenges, thus making it sit at the crossroads between the fields of economics and computational sciences, attracting efforts from economics, marketing, statistical modeling, information retrieval, data mining, and machine learning. The key to successful online advertising is to monetize (or make profits) more and better by learning from the data, which has given rise to a new discipline, called computational advertising.

Online advertising typically operates in the following steps:

1. A browser requests a Web page from a content provider.

2. Ads are picked by an Ad Network (e.g., Yahoo, Google, or Bing).

3. Ads are displayed on the requested Web page.

Online advertising occurs in three main settings:

**Display advertising**: Ads are displayed on a Web page, in the form of graphical banners or other visual or multimedia components, typically to increase brand awareness and without considering the content of the Web page. They are usually shown on Web sites with high visit rates, such as Web portals and popular news sites. Also, these Ads are typically targeted at particular user demographics, such as “females between 35 and 45 years old.” The computational problem typically reduces to estimating the optimal allocation of Ads to demographics subject to supply (number of available impressions) and demand (desirability of each demographic segment for the advertiser) constraints.

**Content match-based advertising**: Ads whose content matches the content of the requested Web page are selected by the Ad network and displayed on the page. Typically the Ads are of a textual type or have associated text descriptions to enable the content matching. An example of this model is the Google AdSense. Clearly, this setting allows targeted advertising. The matching algorithm can be based on information retrieval methods or text classification methods.

**Sponsored search advertising**: A searcher submits a query to a search engine. The search engine picks the Ads to display by matching the query and/or the search results to a set of sponsored Ads and then displays these Ads on top of and/or to the right of the non-sponsored (organic) results. Again, the matching algorithm can be based on information retrieval methods or text classification methods.

Suppose we show an Ad N times on the same spot, then the revenue, R, under each revenue model would be as follows :

Under CPM: R = N \* CPM

Under CPC: R = N \* CTR \* CPC

Under CPA: R = N \* CTR \* Conv.Rate \* CPA,

where the clickthrough rate, CTR, is the probability of a click given an impression and the conversion rate, Conv.Rate, is the probability of a user conversion on the advertiser’s landing page, given a click. Thus, we see that an accurate estimation of CTR is paramount to estimating Ad revenue.

The prevalent revenue model for sponsored search Ad listings is “CostPer-Click” (CPC) where the advertiser pays only if the Ad advertiser targets special keyword markets by bidding on search queries. An advertising campaign usually consists of many Ad groups. Each Ad group consists of a set of phrases or keywords that the advertiser bids on. Each Ad group is also associated with a creative made up of a short title, a longer description, and a display URL. The URL points to the landing page where the user ends up after clicking the Ad. To show Ads, an advertiser can choose to use exact matching between search queries and the Ad group keywords, or advanced matching which, in addition to exact match, also matches Ads to related queries.

Three-step approach to sponsored search:

Step 1: Finding relevant Ads for a query Step

2: Estimating the click through rate (CTR) for the retrieved Ads and ranking them Step

3: Displaying a subset of the top ranked Ads on the search page

Step 1, finding relevant Ads to a query, looks like an information retrieval (IR) problem. However, the collection of Web documents indexed by a search engine is much larger than the collection of Ad groups. Also, matching Ads to queries is much more flexible than matching documents to queries. This is because the way that Ads may be related to queries is much broader and subtle than how documents relate to queries. Moreover, determining relevant candidate Ads for infrequent queries is very important for sponsored search because the power law of queries makes it so that infrequent queries together make up the bulk of all submitted queries (the long tail), and there is consequently a tremendous financial stake in benefiting from this long tail. Despite these differences, IR-based Ad relevance estimation methods exist. However, machine learning or data mining based methods that take into account click-feedbacks are gaining more attention. They typically perform the tasks in step 1 and step 2 simultaneously by directly estimating the CTR value for each Ad. Below, we briefly explain each family of methods.

IR-based methods: An information retrieval (IR) based method typically uses one of the three different approaches [2]: Vector space models: Match a query to an Ad using the cosine similarity between their vector space representations. Probabilistic models: Predict, for each (ad, query) pair, the probability that the ad is relevant to the query, e.g., using the Okapi BM25 model. Language models: Assume that Ads and queries are generated by statistical models (e.g., multinomial models) of how words are used in the language, and based on the models, translate query and Ad generation probabilities into relevance scores.

Click mining-based methods: Taking into account the past click history presents enormous advantages because it can adapt to dynamic settings and because the click history data is available at low cost and in large quantities. The task of this family of methods is to estimate the CTR, i.e.,

CTR = Pr(click| query, ad, user)

Below, we briefly explain three approaches for click mining-based methods [2]:

• Feature-based modeling: It represents both the queries and the Ads using features such as bag of words, phrases, or topics, and for the Ad, optional additional presentation features such as size, etc. Then the CTR of an Ad can be estimated using machine learning methods, for instance logistic regression.

• Similarity-based collaborative filtering: It predicts the CTR of an Ad i in response to a new query q based on the past CTRs for past Ads, as observed in the click history data, and on the similarity between the new Ad and the past Ads

• Matrix factorization: It predicts the CTR of an Ad i in response to a new query q based on a factorization model of the past CTRs for past Ads, as observed in the click history data.