Annexure – III (Answer Key Template)

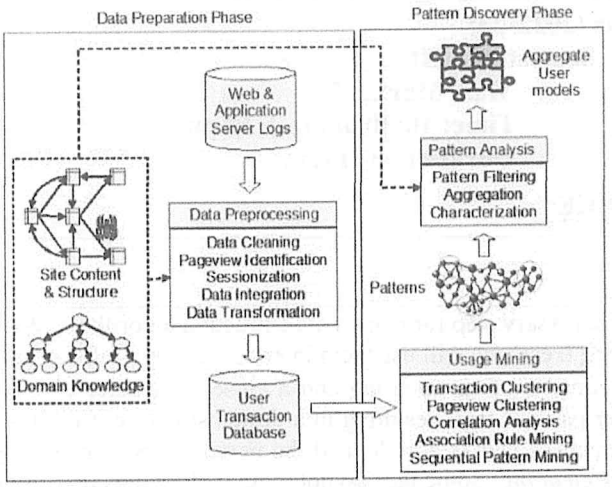
SVKM'S 10:30 am to 1:30 pm
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
Academic Year (2022-23)
Year: B. Tech Semester: VIII

: B. Tech in Computer Engineering **Max. Marks: 75**
Course: Web Intelligence **Time: 10:30 am to 1:30 pm**
05/2023 **Duration: 3 Hours**

ANSWER KEY

DOM (Document Object Model) tree building from input pages is a necessary step for many data extraction algorithms. We describe methods for building **DOM trees**, which are also commonly called **tag trees** (we will use them interchangeably in this chapter).
Using Tags Alone: Most HTML tags work in pairs. Each pair consists of an open tag and a close tag (indicated by < > and respectively). Within each corresponding tag-pair, there can be other pairs of tags, resulting in a nested structure. Building a DOM from a page using its HTML code is thus natural. In the tree, each pair of tags is a node, and the nested pairs of tags within it are children of the node. Two tasks need to be performed: 1. *HTML code cleaning*: Some tags do not require close tags (e.g., <li>, <hr> and <p>) although they have close tags. Hence, additional close tags should be inserted to ensure all tags are balanced. formatted tags also need to be fixed. Such error tags are usually close tags that cross different nested blocks, e.g., <tr> … <td> … < … </td>, which can be hard to fix if multiple levels of nesting exist. There are open source programs that can be used to clean up HT pages. One popular program is called **tidy** (available at http://tidy.sourceforge. net/). 2. *Tree building*: We can follow the nested block the HTML tags in the page to build the DOM tree. It is fairly straightforward. We will not discuss it further. This method works for r pages. However, for some ill-formatted tags, even the tidy program cannot fix. Then, the constructed DOM trees may be wrong, wl makes it difficult for subsequent data extraction. **Using Tags and Visual Cues:** Instead of analyzing the HTML code to fix en rendering or visual information (i.e., the locations on the screen at which tags are rendered) can be used to infer the struct relationship among tags and to construct a DOM tree. This method leads to more robust tree construction due to the high error tolera of the rendering engines of Web browsers (e.g., Internet Explorer). As long as the browser is able to render a page correctly, its tag can be built correctly.

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



OR

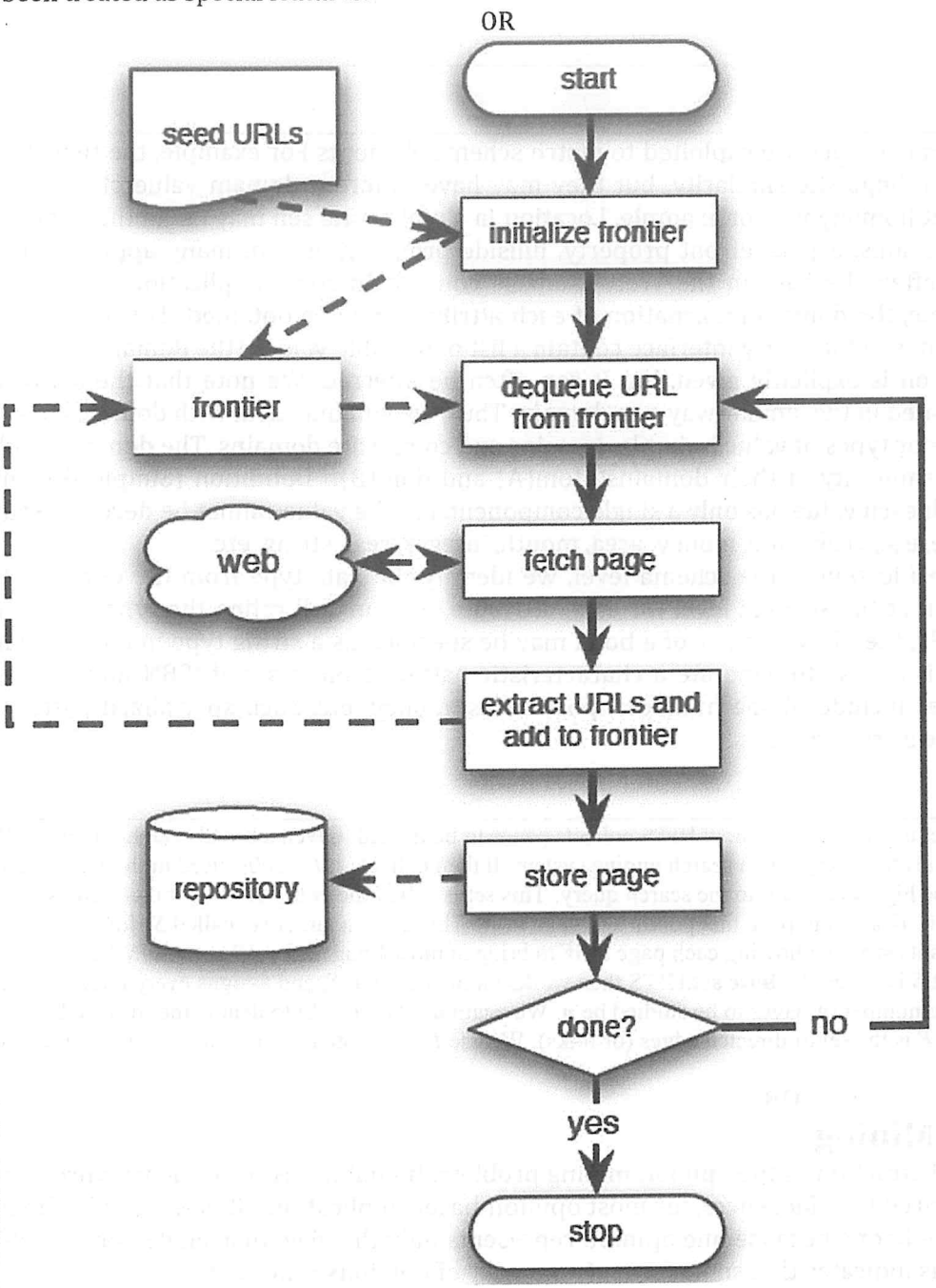
We have a set of users U and a set of items S to be recommended to the users. Each user u in U is defined with a **user profile** that includes various user characteristics or features, such as age, gender, income, marital status, tastes, preferences, needs, etc. In the simplest case, the profile contains only a single (unique) element, such as User ID. Similarly, each item in S is also defined with a set characteristics or features. For example, in a movie recommendation application, where S is a collection of movies,each movie can be represented by its ID, title, genre, director, year of release, leading actors, etc. Again, in the simplest case, a movie may be represente only by a unique Movie ID. In most applications, the spaces of both U and S can be very large. Let p be an utility function that measu the usefulness of item s to user u, i.e., $p:U\times S \rightarrow R$, where R is a totally ordered set (e.g., non-negative integers or real numbers within certain range). The task of a recommender system is twofold: 1. Learn the utility function p. The objective function for learning p can be arbitrary (e.g., user satisfaction or seller profitability) depending on applications.2. Use p to predict the utility value of each item s ($\in S$) to each user u ($\in U$), except those items that already have utility values for u from the input data, and then recommend the top k items to user u.There are two basic approaches to recommendations [1, 11]: • **Content-based recommendations**: The user will be recommended items similar to the ones the user preferred in the past. • **Collaborative filtering** (or **collaborative recommendations**) The user will be recommended items that people with similar tastes and preferences liked in the past. There are also many approaches that combine collaborative and contentbased methods. These methods are called the **hybrid** methods. They typically work in one of t following ways [1, 11]: 1. Implementing collaborative filtering and content-based methods separately and combining their prediction just like ensemble techniques in classification learning (Sect. 3.10) 2. Incorporating some content-based characteristics into a collaborative approach 3. Incorporating some collaborative characteristics into a content-based approach 4. Constructing a general unifying model that incorporates both contentbased and collaborative characteristics All of these approaches have been employed by researchers. See a survey in [1]. Here, we only focus on the two basic approaches.

Sentiment classification obviously can be formulated as a supervised learning problem with three classes, posit negative, and neutral. Training and testing data used in the existing research are mostly product reviews, which is surprising due to the above assumption. Since each review already has a reviewer-assigned rating (e.g., 1–5 stars), trair and testing data are readily available. For example, a review with 4 or 5 stars is considered a positive review, a review v 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, wl classifies documents into predefined topic classes, e.g., politics, sciences, sports, etc. In topic-based classification, to related words are important. However, in sentiment classification, topic-related words are unimportant. Instead, opir words (also called sentiment words) that indicate positive or negative opinions are important, e.g., great, excell amazing, horrible, bad, worst, etc Terms and their frequency. These features are individual words or word ngrams their frequency counts (they are also commonly used in traditional topic-based text classification). In some cases, w positions may also be considered. The TF-IDF weighting scheme from information retrieval may be applied too. Th 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 h been treated as special features.

OR



Implementation Issues : Fetching
Parsing
Stopword Removal and Stemming
Link Extraction and Canonicalization
Spider Traps
Page Repository
Concurrency

Types : Universal
Focused

Topical

In this type of matching, value characteristics are exploited to match schema elements For example, the two attrib names may match according to the linguistic similarity, but they may have different domain value characterist Then, they may not be the same but homonyms. For example, Location in a real estate sell may mean the address, could also mean some specific locations, e.g., lakefront property, hillside property, etc. In many applications, d instances are available, which is often the case in the Web database context. In some applications, although instance information is not available, the domain information of each attribute may be obtained. This is the case Web query interfaces. Some attributes in the query interface contain a list of possible values (the domain) for the u to choose from. No type information is explicitly given, but it can often be inferred. We note that the set of va 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 simila of two attributes, A and B, is the similarity of their domains: dom(A) and dom(B). Definition (Simple Domain 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 val Even if there is a type specification at the schema level for each attribute, we can still refine the type to find m characteristic patterns. For example, the ISBN number of a book may be specified as a string type in a given sche However, due to its fixed format, it is easy to generate a characteristic pattern from a set of ISBN numbers, e.g regular expression. Other examples include phone numbers, post codes, money, etc. Such specialized patterns more useful in matching compatible attribute types.

Before describing the HITS algorithm, let us first describe how HITS collects pages to be ranked. Given a broad search query, q, HIT collects a set of pages as follows:1. It sends the query q to a search engine system. It then collects t ($t = 200$is used in the HITS paper) highest ranked pages, which assume to be highly relevant to the search query. This set is called the **root** set W. 2. It then grows W by including any page pointed to by a page in W and any page that points to a page in W. This gives a larger set called S.However, this s can be very large. The algorithm restricts its size by allowing each page in W to bring at most k pages ($k = 50$ is used in the HITS paper) pointing to it into S. The set S is called the **base set**.HITS then works on the pages in S, and assigns every page in S an **authority score** and a **hub score**. Let the number of pages to be studied be n. We again use $G = (V, E)$ to denote the (directed) link gr of S. V is the set of pages (or nodes) and E is the set of directed edges (or links). We use $\boldsymbol{L}$ to denote the adjacency matrix of the grapl

OR

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 complex and intimidating unstructured text. Moreover, for most opinion-based applications, it is essential to analy collection of opinions rather than only one because one opinion represents only the view of a single person, whic 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 d such as reviews, tweets, and social media posts. The main problem with opinion mining is the subjectivity of langu 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 w using idiomatic expressions, sarcasm, irony, and other linguistic devices that can be difficult to interpret. Secc context plays a crucial role in determining the sentiment of a text. For example, the same sentence can be positiv one context and negative in another. Third, there is a high degree of variability in people's opinions, making it diffi 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 r linguistic devices are being developed all the time. As such, researchers in this field must continually refine th 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 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 innovati out of the box thinking - the risk of a filter bubble, where automated content analysis combined with behaviou analysis leads to a very effective but ultimately deviating selection of relevant opinions and content, so that the use 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 a government, but not by citizens. In other words, government have the means today to monitor public opinion in w that are not available to the average citizens. While content production and publication has democratized, cont analysis has not.
- The integration of opinion with behaviour and implicit data, in order to validate and provide further analysis i the data beyond opinion expressed
- The continuous need for better usability and user-friendliness of the tools, which are currently usable mainly data analysts

Problem Definitions We use the following review segment on iPhone to introduce the problem (an id numb 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) ' voice quality was clear too. (5) However, my mother was mad with me as I did not tell her before I bought it. (6) : 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 several opinions in this review.

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

We are now ready to study the first approach to data extraction, namely **wrapper induction**, which is based on supervised learning. A wrapper induction system learns data extraction rules from a set of labeled training examples. Labeling is usually done manually, whi simply involves marking the data items in the training pages/examples that the user wants to extract. The learned rules are then applie to extract target data from other pages with the same mark-up encoding or the same template. The algorithm discussed in this section based on the Stalker system [28]. Related work includes WIEN [19], Softmealy [17], WL_2 [10], the systems in [18] and [43], etc. The next section describes a different learning approach, which is based on the IDE system given in [38]. Stalker models the Web data as nested relations. Let us model the restaurant page in Fig. 9.5. It has four addresses in four different cities. The type tree of the data is given in Fig. 9.6 (the country code is omitted). For each type, we also added an intuitive label. The wrapper uses a tree structure base on this to facilitate extraction rule learning and data extraction.

OR

The wrapper induction method discussed in the previous section requires a set of labeled examples to learn extraction rules. Active learning may be applied to identify informative examples for labeling to reduce the manual labeling effort. In this section, we introdu an instance-based learning approach to wrapper building, which has been implemented by a company and is in commercial use. This

method does not learn extraction rules. Instead, it extracts target items in a new instance/page by comparing their prefix and suffix tol strings with those of the corresponding items in the labeled examples. At the beginning, the user needs to label only a single example, which is then used to identify target items from unlabeled examples. If some item in an unlabeled example cannot be identified, it is sent for labeling, which is *active learning* but with no additional mechanism. Thus, in this approach the user labels onl minimum number of training examples.

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 en does not mean that the author has positive opinions on all aspects of the entity. Likewise, a negative opinionated docum does not mean that the author dislikes everything. In a typical opinionated document, the author writes both positive an 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 wh but about its picture quality. However, the sentence "I love this camera" evaluates the camera as a whole, i.e., the GENER 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 sentim 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.

The fundamental problem is schema matching, which takes two (or more) database schemas to produce a mapp between elements (or attributes) of the two (or more) schemas that correspond semantically to each other. ' objective is to merge the schemas into a single global schema. This problem arises in building a global database t comprises several distinct but related databases. One application scenario in a company is that each department its database about customers and products that are related to the operations of the department. Each databas typically designed independently and possibly by different people to optimize database operations required by functions of the department. This results in different database schemas in different departments. However, consolidate the data about customers or company operations across the organization in order to have a m complete understanding of its customers and to better serve them, integration of databases is needed. The integrat problem is clearly also important on the Web as we discussed above, where the task is to integrate data from multi sites.

Schema matching is challenging for many reasons. First of all, schemas of identical concepts may have structural : naming differences. Schemas may model similar but not identical contents, and may use different data models. T 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 matc between two schemas because some semantic information that determines the matches between two schemas r not be formally specified or even documented. Thus, any automatic algorithm can only generate candidate matc that the user needs to verify, i.e., accept, reject or change. Furthermore, the user should also be allowed to spe 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.

S_1	S_2
Cust	Customer
CNo	CustID
CompName	Company
FirstName	Contact
LastName	Phone

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

Cust.CNo ≅ Customer.CustID
Cust.CompName ≅ Customer.Company
{Cust.FirstName, Cust.LastName} ≅ Customer.Contact

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 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 e attribute are provided. No schema is available. Such cases occur quite frequently on the Web, where we need to ma corresponding columns of the hidden schemas.
3. Integrated matching of schema, domain and instance data: In this type of match, both schemas and instance d (possibly domain information) are available. The match algorithm can exploit clues from all of them to perfc matching.

Schema-Level Matching

A schema level matching algorithm relies on information about schema elements, such as name, description, data t and relationship types (such as part-of, is-a, etc.), constraints and schema structures. Before introducing sc matching methods using such information, let us introduce the notion of match cardinality, which describes number of elements in one schema that match the number of elements in the other schema. In general, given t schemas, S1 and S2, within a single match in the match relation one or more elements of S1 can match one or m 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 correspond 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:

S_1	S_2
Cust	Customer
CustomID	CustID
Name	FirstName
Phone	LastName

We can find the following 1:1 and 1:*m* matches:

1:1	CustomID	CustID
1:*m*	Name	FirstName, LastName

m:1 match is similar to 1:m match; m:n ma is considerably more complex. An example of an m:n match is to match Cartesian coordinates with polar coordina 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 schen There are two main types of information in schemas, natural language words and constraints. Thus, there are t 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

N2 – Synonyms: The names of two elements from different schemas are **synonyms**, e.g., Customer ≅ Client. This requires the use of thesaurus and/or dictionaries such as WordNet. In many cases, domain dependent or enterprise specific thesaurus and dictionaries are required.

N3 – Equality of hypernyms: *A* is a **hypernym** of *B* if *B* is **a kind of** *A*. If *X* and *Y* have the same hypernym, they are likely to match. For example, "Car" *is-a* "vehicle" and "automobile" *is-a* "vehicle". Thus, we have Car ≅ vehicle, automobile ≅ vehicle, and Car ≅ automobile.

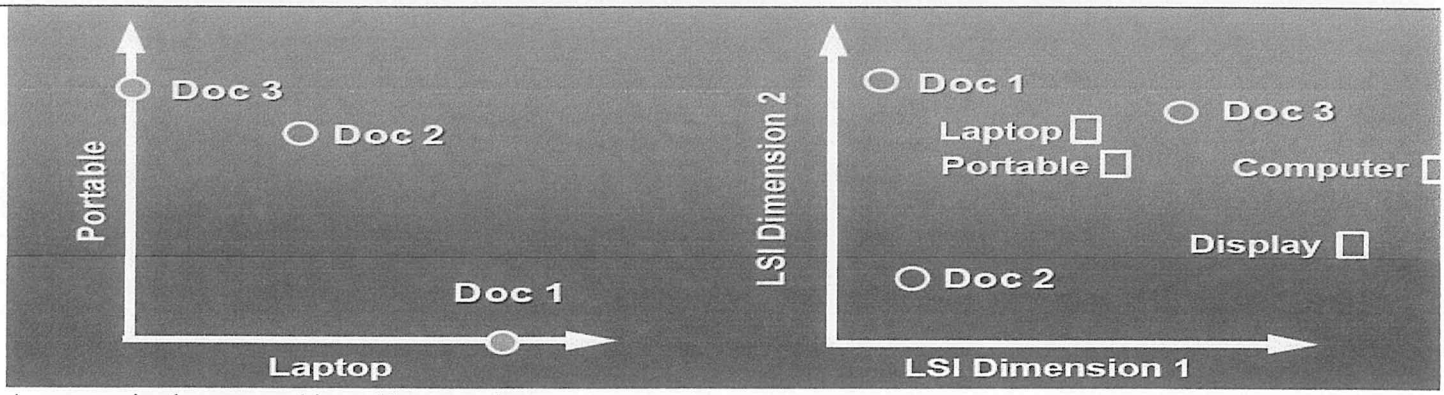
N4 – Common substrings: Edit distance and similar pronunciation may be used. For example, CustomerID ≅ CustID, and ShipTo ≅ Ship2.

N5 – Cosine similarity: Some names are natural language words or phrases (after pre-processing). Then, text similarity measures are useful. **Cosine similarity** is a popular similarity measure used in information retrieval (see Chap. 6). This method is also very useful for Web query interface integration since the labels of the schema elements are natural language words or phrases (see the query interfaces in Fig. 10.1)

N6 – User provided name matches: The user may provide a domain dependent match dictionary (or table), a thesaurus, and/or an ontology.

a) PageRank is a static ranking of Web pages in the sense that a PageRank value is computed for each page off-line and it does depend on search queries. Since PageRank is based on the measure of prestige in social networks, the PageRank value of eacl page can be regarded as its prestige.We now derive the PageRank formula. Let us first state some main concepts again in the Web context.**In-links** of page *i*: These are the hyperlinks that point to page *i* from other pages. Usually, hyperlinks from the same site are not considered.**Out-links** of page *i*: These are the hyperlinks that point out to other pages from page *i*. Usually, links to pages of the same site are not considered.From the perspective of prestige, we use the following to derive the PageRank algorithm.1. A hyperlink from a page pointing to another page is an implicit conveyance of authority to the target page. Th the more in-links that a page *i* receives, the more prestige the page *i* has. 2. Pages that point to page *i* also have their own prestige sco A page with a higher prestige score pointing to *i* is more important than a page with a lower prestige score pointing to *i*. In other worc page is important if it is pointed to by other important pages.

b) Each row and column of A gets mapped into the k-dimensional LSI space, by the SVD.
Claim – this is not only the mapping with the best (Frobenius error) approximation to A, but in fact improves retrieval.



A query q is also mapped into this space, by

$$q_k = q^T U_k \Sigma_k^{-1}$$

Query NOT a sparse vector

c) In its simplest form, the inverted index of a document collection is basically a data structure that attaches each distinctive term witl list of all documents that contains the term. Thus, in retrieval, it takes constant time to find the documents that contains a query term. Finding documents containing multiple query terms is also easy as we will see later.Given a set of documents, $D = \{d_1, d_2, \ldots, d_N\}$, ea document has a unique identifier (ID). An inverted index consists of two parts: a vocabulary V, containing all the distinct terms in the document set, and for each distinct term t_i an **inverted list** of postings. Each **posting** stores the ID (denoted by id_j) of the document d_j contains term t_i and other pieces of information about term t_i in document d_j. Depending on the need of the retrieval or ranking algorit different pieces of information may be included. For example, to support phrase and proximity search, a posting for a term t_i usually consists of the following, $<id_j, f_{ij}, [o_1, o_2, \ldots, o_{|f_{ij}|}]>$ where id_j is the ID of document d_j that contains the term t_i, f_{ij} is the frequency count of t_i in d_j, and o_k are the offsets (or positions) of term t_i in d_j. Postings of a term are sorted in increasing order based o the id_j's and so are the offsets in each posting {Web, mining, useful, applications, usage, structure, studies, hyperlink}

d)

Finding information on the Web has become a critical ingredient of everyday personal, educational, and business life. An informatio seeker communicates with a search engine by typing a query, which in its simplest form, consists of a few keywords or terms, with th aim of finding Web pages whose content matches these terms. Search **query logs** are files that record queries submitted by users alon with the results returned by the search engine and the results that have been clicked by the user. The formulation of a query is very important since it must convey the exact need of the user, meaning that the words in the query should match all and only the documents being sought. However, studies that have analyzed search logs found that users' queries suffered from two limitations. First, queries tend to be short, in fact too short to correctly convey the **user's intent**. Second, the distribution of the conte of the queries does not correspond to that of the documents, which means that it is difficult for users to express their information nee such a way that it could describe the documents that they are seeking. This gap between queries and document contents is due to man reasons, including the ambiguity of some terms that have multiple meanings, as well as the existence of different words that possess t same meaning. Thus it has become imperative to go beyond the single query submitted by the user in order to discover the true intention of the user, also known as the **information need**, with the final aim of providing the accurate information being sought. Recent research on search query log mining has addressed this problem by using data mining techniques that bridge the three areas of Web usage mining, text mining, and, in some cases, Web structure mining. Search engines would not survive without a suitable source of revenue. This revenue has mainly come from the **sponsored search** business model. For this reason, recent researc has also addressed the problem of discovering the user's intent, for an aim that is different from serving only the correct results during search. Instead, their aim has been to target sponsored search advertisements (often called **Ads** in short) more accuratel the right user at the right time. Thus, the accurate identification of the user's information need is a prerequisite to both aims: (1) providing the user with the correct list of results to their query and (2) displaying the **sponsored Ads** that most accurately match the user's preferences.