„Lucian Blaga” University of Sibiu
“Hermann Oberth” Engineering Faculty
Computer Science Department

Web Information Retrieval

First Technical Report

PhD title: “Data Mining for unstructured data”

Author:
Daniel MORARIU, MSc

PhD supervisor:
Professor Lucian VINTAN, PhD

SIBIU, 2005

Contents

1 Introduction

Impressive amounts of information potentially relevant to the user are contained in the web, this information being very chaotic at the moment. The Web is becoming a huge repository of information, and it is built in uncoordinated manner but yet restrictive. The Web is an environment that grows continuously, it is populated and it implies a growth in participants, without having a coordinated manner of building. These characteristics have as a result both pluses and minuses. One of the pluses is the increase in the variety of the content. Information found on the Web is undoubtedly larger than the one available trough news, radio and television. This is due to a considerably large number of web developers. The lack of organization and the heterogeneity of the data represent one of the minuses when trying to find information. Exploring the web is the only way to find relevant information, this method being unfortunately time consuming. Queries that contain words are the simplest method of searching information on the Web – method that usually doesn't return the expected results. Inexperienced users don't know how to formulate complex queries (like using preposition in the queries) and need more time to find information.

The growth of the Web in diversity and dimension gives it an inestimable value that makes it an active repository of information. For the first time we can speak about an environment that has almost as many authors as it has readers. The growing Web content makes it more and more difficult to estimate real values from whole content. Its unsupervised progress makes it contain a large number of redundant information.

The difficulty in finding and organizing relevant information grew exponentially with the growth of information on the Internet and Intranets. Methods for searching relevant information for the user were developed at the same time. Thus, a series of "search engines" was developed. Search engines research grew rapidly in the past years, in areas such as algorithms, strategies and architecture, increasing both effectiveness and quality of results. People want to find relevant information quickly. Searching information on the web can be a frustrating activity when a search engine returns thousands of documents for a given query. When searching, a user inputs a simple keyword query (two or three words usually). The query response is usually a list of pages ranked based on their similarity to the query. Search tools today have the following problems [Sou00]:

- Low precision - due to the irrelevancy of many of the search results – this leading to difficulties in finding relevant information
- Low recall - due to the inability to index all the information available on the web – this leads to difficulties in finding relevant information in un-indexed documents in the search engine database

Most search engines are constantly looking on the net for new web pages and they use the page's summary or some reported keywords, to index the pages in a great database. When a user inputs some specific keywords the searching engine returns the addresses of all documents indexed in the database on those keywords. In order to raise the result's quality, the search engine applies various functions in order to assign relevance to a page (e.g.: page rank, similarity, back link and mixed approaches) and gives precedence to pages with high weight, supposedly indicating greater relevance.

I have structured this PhD report into two main parts (Chapters 2 and 3). Chapter 2 is a tutorial on general data mining included due to its use for further studies in text mining. Than I present text mining as far as changes are concerned in rapport with data mining. I will use this in future studies. At the end of Chapter 2 I present Web content and usage mining in order to obtain the user

profile, my main aim. In Section 2.3 I present a combination of classical literature and up to date research. Chapter 3 is precisely the state of the art on returned search results reorganization from the nowadays search engines. I present here a general perspective of current achievements as well as on going research.

Acknowledgments

First of all I would like to express my sincere gratitude to my PhD supervisor Professor Lucian VINȚAN for his responsible scientific coordination, for providing stimulating discussions and for all his support. I would also like to thank the ones that guided me from the beginning of my PhD studies: mat. Ioana MOISIL, eng. Boldur BĂRBAT, eng. Daniel VOLOVICI, eng. Dorin SIMA and eng. Macarie BREAZU for their valuable generous professional support.

I would also like to thank SIEMENS AG, CT IC MUNCHEN, Germany, especially Vice-President Dr. h. c. mat. Hartmut RAFFLER, for his very useful professional suggestions and for the financial support that he and his company have provided. I want to thank my tutor from SIEMENS, Dr. Volker TRESP, Senior Principal Research Scientist in Neural Computation for the scientific support provided and for his valuable guidance in this wide interesting domain of research. I also want to thank Dr. Kai Yu for useful information in the development of my ideas. Last but not least I want to thank all those who supported me in the preparation of this technical report.

2 Data mining for Hypertext

2.1 Data mining in Databases

Data mining [JaiMic01] refers to extracting or "mining" knowledge from large amounts of data. It is the short term for "knowledge mining from data". Many people treat data mining as a synonymous for another popular used term, *Knowledge Discovery in Databases*, others view data mining simply as an essential step in the process of knowledge discovery in databases. This step is about solving problems by analyzing data presented (usually) in databases and trying to discover such characteristics that can be used to organize massive databases. Thus data mining can be defined as the process of discovering patterns in data and relationships between attributes from data. This process must be automatic or (more usual) semi-automatic. Thus data mining represents the process of discovering interesting knowledge from large amounts of data stored either in databases, data warehouses, or other information repositories. Typically data mining systems have the following major components:

- *Databases, data warehouses, or other information repositories*: Databases are created to perform On-Line Transaction and query Processing (OLTP), covering most of day-to-day operations and being detailed to be easily used for decision making. Data warehouse systems offer help, for the specialists, in data analysis and decision making, collective referred to as On-Line Analytical Processing (OLAP), providing facilities for summarization, aggregation, storing and managing information at different levels of granularity. Information repositories can also be flat files.
- *Databases or data warehouses server* – is responsible for fetching the relevant data, based on the user's data mining request.
- *Knowledge base* – includes the domain knowledge that is used to guide the search or to evaluate the interestingness of resulting patterns. The knowledge can include hierarchical concepts, used to organize attributes or attribute values into different levels of abstraction. The knowledge can be used to assess a pattern's interestingness based on its unexpectedness or additional interestingness constraints or thresholds and metadata.
- *Data mining engine* – is the essential component of the data mining system and ideally consists of a set of functional modules for tasks such as characterization, association, classification, cluster analysis, and evolution and derivation analysis.
- *Pattern evaluation modules* –component typically employing interestingness measures and interacting with the data mining modules so as to focus the search towards interesting patterns. It can used interestingness thresholds to filter out discovered patterns. Alternatively, the pattern evaluation module may be integrated with the mining module, depending on the implementation of the data mining method used. Usually the evaluation of pattern interestingness is put in the mining process so as to confirm only the interesting parents searched.
- *Graphical user interface* – is the module for interaction between users and the data mining systems, allowing the user to interact with the system by specifying a data mining query or task, providing information to help focus the search, and performing exploratory data mining based on the intermediate data mining results. This component allows the user to browse databases and data warehouses schemas or data structures, evaluate mined patterns, and visualize the patterns in different forms.

The process of knowledge discovery in database has more steps, data mining been one of these steps:

- preprocessing data
- data mining
- pattern evaluation
- knowledge presentation

2.1.1 Preprocessing data

This is an important step in the process of knowledge discovery. Today real-world database or repository data are highly susceptible to noise, incomplete and inconsistent data due to their typically huge size. Its aim is to prepare data for analyzing. There are a number of data preprocessing steps:

- data cleaning – to remove noise (a random error or variance in a specific data), and to correct inconsistencies (the same object appears with two different attribute values) in the data (for example a given concept may have different names in different databases, or the same person can be registered as "Bill" in one database, but "William" in another).
- data integration – to merge data from multiple sources into an appropriate form for mining. It combines data from multiple sources into a coherent data store. It refers to the homogeneity of data.
- data selection – to select relevant information from data for analysis
- data transformation – to prepare data for analysis. It contains operations like normalizing data (scaling the values of the attribute so that they fall within a specified range) that can improve the accuracy and efficiency of mining algorithms that involve distance measurements
- data reduction – to reduce data size by aggregation, eliminating redundant features, or clustering

2.1.1.1 Cleaning data

C*leaning data* is the first sub-step and prepares data for processing. Because vast amount of data are involved in the data mining process, these data are usually incomplete, noisy, and inconsistent. Data cleaning routine attempts to fill the missing values, smooth out noise while identifying outliers, and to correct inconsistencies in the data. Outliers are considered values out of range or noise in data. Different methods can be used in the process of filling missing values.

- The "**ignore the samples"** method is not very effective, unless the samples contain several attributes with missing values. This method is especially poor when the percentage of missing values per attribute varies considerably.
- Another method is "**fill the missing value manually",** where the great disadvantage of this method is that it is time consuming and may not be feasible when it is given a large data set with many missing values.
- **"Use a global constant to fill in the missing values"** is the method by which all missing attribute values are replaced with the same constant. This method is simple but it is not recommendable because the mining process may mistakenly think that they form an interesting concept.
- Another method is "**use the attributes mean to fill in the missing value"** where all missing values are replaced with the average of the values for the same attribute in the database.

- **"Use the most probable value to fill in the missing value"** is a good method but it is time consuming because the probable value is determined by the use of regression, inference-based tools using Bayesian formalism, or decision tree induction.

Due to usually noisy data the cleaning data process also uses techniques for smoothing noise in data. The noise is a random error or variance in a specific variable. There are some techniques for smoothing data [JaiMic01]. Usually these methods try to eliminate those values that occur sporadically in the data but without any assurance that those data are not novelty in data. As a result these techniques usually use a variable threshold for eliminating noise at different levels to assure that novelty is not being eliminated. Some of these techniques are:

- **Binning** methods smooth sorted data values by consulting its "neighborhood", that is, the values around it. The sorted values are distributed into a number of "buckets" called *bins*. Because binning methods consult the neighborhood of values, they perform local smoothing. In this smoothing by bin means method, each value in a bin is replaced by the mean value of the bin. In smoothing by bin boundaries, the minimum and maximum values in a given bin are identified as the bin boundaries and each bin value is then replaced by the closest boundary value.
- Another method is **Clustering** where outliers may be detected by clustering. Similar values are organized into groups, or "clusters" and values that fall outside of the set of cluster may be considered outliers.
- **Regression** is a method where data can be smoothened by fitting the data through a function such as the regression. *Linear regression* can be used in finding the best line to fit two variables, so that one variable can be used to predict the other variable. *Multiple linear regressions* are an extension of the linear regression in which more than two variables are involved and the data is fitted in a multidimensional surface. Using regression to find a mathematical equation to fit the data helps smooth out the noise. Usually outliers are identified by using a combination between computer and human inspection. Many methods for data smoothing are also methods for data reduction involving discretization.

Techniques of data preprocessing want to improve the data quality, thereby help to improve the accuracy and efficiency of the subsequent mining process. Preprocessing is an important step in the knowledge discovery process because a quality decision must be based on quality data.

Data recorded in transactions can produce inconsistencies in the data. There may also be inconsistencies due to data integration, where a given attribute can have different names in different databases. Inconsistencies can also be redundancies. Some of these redundancies can be corrected manually or using methods for reducing noise.

2.1.1.2 Data integration and transformation

The process of data mining often requests merging data from multiple data stores. Thus in the ***data integration*** steps data from multiple sources are combined into a coherent data store. The source may include multiple databases, data cubes or flat files. Data cube consists of a lattice of cuboids, each corresponding to a different degree of summarization of the given multidimensional data. Partial materialization refers to the selective computation of a subset of the cuboids in the lattice and full materialization refers to the computation of all of the cuboids in the lattice. There are a number of issues to consider during data integration. *Scheme integration* is the method that merges different kinds of input data (e.g. different file formats, different structure names, a. o.). Other important issues can be *Redundancy* where an attribute is considered redundant if it can be "derived" from another attribute. Inconsistencies in attributes dimension or naming can also cause

redundancies in the resulting data set. Redundancies can be detected using correlation analysis, where analysis can measure if one attribute implies the other, based on the available date. The correlation between two attributes can be measured by [JaiMic01]:

$$r_{A,B} = \frac{\sum_{k=1}^{n}(A_k - \overline{A})(B_k - \overline{B})}{(n-1)\sigma_A \sigma_B} \quad , \qquad (2.1)$$

where A and B are the attributes, *n* represents the number of samples, $\overline{A}$ *and* $\overline{B}$ are the respective mean values of A and B, and σ_A *and* σ_B are the standard deviation of A and B computed thus:

$$\sigma_A = \sqrt{\frac{\sum_{k=1}^{n}(A_k - \overline{A})^2}{n-1}} \qquad (2.2)$$

The correlation coefficient ranges from 1 (perfectly correlated), through 0 (no correlation) to -1 (perfectly invert correlated). If the resulting of $r_{A,B}$ are greater then 0, then A and B are positively correlated, and if the value of A increases then the value of B increases. When $r_{A,B}$ tend to 0 then there is no correlation between A and B. The higher the value, the more each attribute implies the other. Hence, a high value may indicate that A (or B) can be removed as a redundancy. If the resulting is equal with 0, then A and B are independent and there is no correlation between them. If the resulting value is less than 0, then A, and B are negatively correlated (when the values of the one attribute increase as the values of the other attribute decrease). When redundancies between attributes are detected, duplication at the sample level is detected. The above formulas fail when $A_k = \overline{A}$ (or / and $B_k = \overline{B}$) $\forall$ k=1, n, when the values are constants, leading to nonsense values (in this particular case the attribute need to be eliminated). Another important issue in data integration is *detection and resolution of data value conflict*. This is due to differences in representation, scaling or encoding. For example some databases can use the British unit measure and also can use American system unit. Careful data integration from multiple sources can help reduce and avoid redundancies and inconsistencies in the resulting data set.

Data transformation is used to transform or consolidate data into forms appropriate for mining. Data transformation can involve some methods like **smoothing** (for noise removal method presented in 2.1.1.1), **aggregation** (for summarizing the data), **generalization** (for replacing the low level data with a high level concept), **normalization** (rescaling attributes to fit in the specified range) and **attribute construction** (construction and adding of new attributes to help the mining process). The attribute is normalized by scaling its values so that they fit to a small specified range.

Normalization is particularly useful for classification algorithms involving, for example, neural networks or distance measurements. There are many methods for data normalization. Method *min-max normalization* performs a linear transformation on the original data. If that min_A and max_A are the minimum and maximum values of an attribute A, this method maps a value *v* of A to *v'* in the range [new_min_A, new_max_A], where new_min_A and new_max_A are the limits of the new range, by computing:

$$v' = \frac{v - \min_A}{\max_A - \min_A}(new\_\max_A - new\_\min_A) + new\_\min_A . \qquad (2.3)$$

This method preserves the relationships between the original data values. The second normalization method is *z-score normalization* where the values for an attribute A are normalized based on the mean and standard deviation of A. Thus a value *v* is normalized to v' by computing:

$$v' = \frac{v - \overline{A}}{\sigma_A}, \qquad (2.4)$$

where $\overline{A}$ and σ_A are the mean and the standard deviation for the attribute A (presented earlier). This method is used when the minimum and the maximum for the attribute A are unknown, or when are outliers dominate. Another method is *normalization by decimal scaling* where the A's attribute values decimal point is moved. The number of decimal points moved depends on the maximum absolute value of A. A value *v* of A is normalized to *v'* by computing:

$$v' = \frac{v}{10^j}, \qquad (2.5)$$

where j is the smallest integer such that max(|v'|)<1. Normalization can change the original data quite a bit.

In **attribute construction**, the new attributes are constructed from the given attributes and added in order to help improve the accuracy and understanding of the structure in the high dimensional data. Attribute construction can help ease the fragmentation problem when decision tree algorithms are used for classification. By combining attributes, attribute construction can discover missing information about the relationship between data attributes that can be useful for knowledge discovery.

2.1.1.3 Data reduction

Complex data analysis and data mining on huge amounts of data may take a very long time, making such analysis impractical or infeasible. Data reduction is a technique that can be applied to obtain a reduced representation of the data set that is much smaller in volume, and closely maintains the integrity of the original data. That is, mining on reduced data set should be more efficient yet produce the same (or almost the same) analytical results. There are some techniques for data reduction (for details see [JaiMic01]):

- **data cube aggregation** – some aggregation operations are applied to the data in the construction of the data cube. Data cubes store multidimensional aggregated information. Each cell holds an aggregate data value, corresponding to the data point in multidimensional space. Concept hierarchies may exist for each attribute, allowing the analysis of data at multiple levels of abstraction. Data cubes provide fast access to pre-computed, summarized data, thereby benefiting on-line analytical processing as well as data mining. The cube created at the lowest level of abstraction is referred to as the *base cuboid.* A cube for the highest level of abstraction is the *apex cuboid.* Data cube created for various levels of abstraction are often referred to as *cuboids*. Because data cube provide fast access to pre-computed data, they should be used when possible to reply to queries regarding aggregated information.
- **dimension reduction** – irrelevant or redundant attributes or dimensions may be detected and removed. Typically, attribute subset selection methods are applied, where the goal is to find a minimum set of attributes such that the resulting probability distribution of the data classes is as close as possible to the original distribution obtained using all attributes. The "best" or "worst" attribute are determined using tests of statistical significance, which assume that the attributes are independent of one another. Many other attribute evaluation measures can be used, such as the information gain and the average entropy (presented in the next paragraph).

- **data compression** – encoding mechanisms are used to reduce or "compress" the data set size. There are two popular methods of lossy data compression (when we can reconstruct only an approximation of the original data, with losses). First there is the *wavelet transformation* method where the linear signal processing techniques are applied to the data vector, and transform it into a numerically different vector of wavelet coefficients. The two vectors, the original and the wavelet, are of the same length, but the wavelet vector can be truncated and can be retained by storing only a small fraction of the strongest coefficients. The second method is the *principal component analysis* where the idea is to search for an orthogonal vector that can best be used to represent the data and the dimension is smaller than the original dimension. The original data are thus projected onto a much smaller space.
- **numerosity reduction** – the data are replaced or estimated by alternative, smaller data representation. These techniques may be parametric (where the model is used to estimate the data, so that typically only the data parameter needs to be stored, instead of the actual data), or nonparametric (where histograms, clustering and samples are used for storing reduced representation of the data).
- **discretization and concept hierarchy generation** – is used to reduce the number of values for a given continuous attribute by dividing the range of the attribute into intervals. Interval labels can then be used to replace actual data values reducing the number of values for an attribute. A concept hierarchy allows the mining of data at multiple levels of abstraction and is a powerful tool for data mining. There are different methods for numeric concept hierarchy generation like binning, histogram analysis, cluster analysis and entropy based discretization (the first three presented earlier).

2.1.1.4 Entropy-Based Discretization

The measure named *Entropy Based Discretization* is a measure commonly used in information theory, that characterizes the (im) purity of an arbitrary collection of samples, being a measure of homogeneity of samples. The entropy can be used to recursively partition the values of a numeric attribute.
Given a collection S of *n* samples grouped in *c* target concepts (classes), the entropy of S relative to the classification is:

$$Ent(S) = -\sum_{i=1}^{c} p_i \log_2(p_i) \qquad (2.6)$$

where p_i is the proportion of S belonging to class *i*. In all calculations involving entropy we define $0*\log_2 0$ to be 0. Note that the entropy is 0 if all members of S belong to the same class, and the entropy is maxim ($\log_2 c$) when the samples are equally distributed to classes. If the samples, in the collection, are unequally distributed in classes the entropy is between 0 and $\log_2 c$.
One interpretation of entropy from information theory is that it specifies the minimum number of bits of information needed to encode the classification of an arbitrary member of S. If p_i is 1, the receiver knows the drawn sample will be positive, so no message needs to be sent, and the entropy is zero. On the other hand, if p_i is 1/c, c/2 bits are required to indicate whether the drawn sample is in what class. For example in the two classes case if p_i is 0.8, then a collection of messages can be encoded using an average less than 1 bit per message by assigning shorter codes to collections of positive samples and longer codes to less likely negative samples. The logarithm is still base 2 (even if we have more classes then 2) because entropy is a measure of the expected encoding length measured in bits. If the target attribute can take *c* possible values, the entropy can be as large as $log_2 c$.

To illustrate this, suppose S is a collection of 12 samples (presented in Table 2.1). The presented collection contains 3 classes denoted "*a*", "*b*", and "*c*". There are 5 samples in class *"a"*, 3 samples in class *"b"* and 4 samples in class *"c"*. We adopt the notation S=[5a, 3b, 4c]. The entropy of S relative to this classification computed using (2.6) is:

$$Entropy[5a,3b,4c] = -\frac{5}{12}\log_2\frac{5}{12} - \frac{3}{12}\log_2\frac{3}{12} - \frac{4}{12}\log_2\frac{4}{12} = 1.5545$$

Attributes(words) / Samples (docs)	$attrib_1$	$attrib_2$	$attrib_3$	$attrib_4$	$attrib_5$	$attrib_6$	classes
v_1	1	5	7	0	1	1	a
v_2	1	1	0	0	7	5	b
v_3	0	0	1	7	1	0	c
v_4	1	3	7	1	0	1	a
v_5	2	1	8	0	0	0	a
v_6	0	0	0	0	10	7	b
v_7	0	0	7	10	7	0	c
v_8	0	0	5	3	8	1	c
v_9	2	2	5	2	0	1	a
v_{10}	0	1	0	1	9	5	b
v_{11}	0	1	4	1	9	2	c
v_{12}	1	4	6	1	0	0	a

Table 2.1 Training examples with term frequency

Entropy represents the expected number of bits needed to encode the class (for us "a", "b", "c") of a randomly drown sample from S (under the optimal, shortest – length code). Therefore, entropy is a measure of the impurity in a collection of training samples. Using entropy an attribute effectiveness measure is defined in classifying the training data. The measure is called *information gain*, and is simply the expected reduction in entropy caused by partitioning the samples according to this attribute. More precisely, the information gain of an attribute relatively to a collection of samples S, is defined as:

$$Gain(S,A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v) \qquad (2.7)$$

where Values(A) is the set of all possible values for attribute A, and S_v is the subset of S for which attribute A has the value v. The first parameter is just the entropy of the original collection S, and the second term is the expected value of the entropy after S is partitioned using attribute A. In other words, the information gain is therefore the expected reduction in entropy caused by knowing the value of attribute A. The information gain is the number of bits saved when encoding the target value of an arbitrary member of S, by knowing the value of attribute A.

Continuing the example above we now attempt to compute the informational gain. Attributes can only take two values: 0 if the sample doesn't contain that attribute and 1 otherwise. We denote the set of samples containing an attribute as $S_{attribi>0}$ and $S_{attribi=0}$ otherwise. For example for $attrib_1$ the 12 samples are divided as follows, 5 of the class *"a"* and 1 of the class *"b"* are in $S_{attrib1>0}$ and 2 from the class *"b"* and 4 from the class *"c"* are in $S_{attrib1=0}$. The informational gain in respect to each attribute is computed below using formula 2.7:

$$Gain(S, attrib_1) = Entropy(S) - \frac{6}{12} Entropy(S_{attrib_1=0}) - \frac{6}{12} Entropy(S_{attrib_1>0})$$
$$= 1{,}5545 - 0{,}5 * 0{,}6498 - 0{,}5 * 0{,}9182 = 0{,}7705$$
$$Gain(S, attrib_2) = Entropy(S) - \frac{8}{12} Entropy(S_{attrib_2>0}) - \frac{4}{12} Entropy(S_{attrib_2=0})$$
$$= 1.5545 - 0.66 * 1.2987 - 0.333 * 0.811278 = 0.418296$$
$$Gain(S, attrib_3) = Entropy(S) - \frac{9}{12} Entropy(S_{attrib_3>0}) - \frac{3}{12} Entropy(S_{attrib_3=0})$$
$$= 1.5545 - 0.75 * 0.991076 - 0.25 * 0.0 = 0.811278$$
$$Gain(S, attrib_4) = Entropy(S) - \frac{8}{12} Entropy(S_{attrib_4>0}) - \frac{4}{12} Entropy(S_{attrib_4=0})$$
$$= 1.5545 - 0.666 * 1.4056 - 0.333 * 1.0 = 0.284159$$
$$Gain(S, attrib_5) = Entropy(S) - \frac{8}{12} Entropy(S_{attrib_5>0}) - \frac{4}{12} Entropy(S_{attrib_5=0})$$
$$= 1.5545 - 0.666 * 1.4056 - 0.333 * 0.0 = 0.617492$$
$$Gain(S, attrib_6) = Entropy(S) - \frac{8}{12} Entropy(S_{attrib_6>0}) - \frac{4}{12} Entropy(S_{attrib_6=0})$$
$$= 1.5545 - 0.666 * 1.436278 - 0.333 * 1.0 = 0.263733$$

Thus if a threshold equal to 0.5 is applied after computing the informational gain we will take into account only $attrib_1$, $attrib_3$, $attrib_5$. Thus entropy based discretization can reduce data size using class information and allows the selection of the best attributes.

2.1.2 Data mining

Data mining is an essential step in process of knowledge discovery data where AI methods are applied in order to extract patterns (rules) from data. In the data mining step the user communicates with the data mining system using a set of *data mining primitives* designed in order to facilitate efficient and fruitful knowledge discovery. Those primitives include the database portion or the data set specifications in which the user is interested and the kinds of knowledge to be mined. A *data mining query language* is designed to incorporate these primitives, allowing users to flexibly interact with the data mining systems. A data mining task can be specified in the form of a data mining query, which is the input of the data mining system.

Data mining can be classified into *descriptive data mining* and *predictive data mining*. Concept description is based on descriptive data mining and describes a given set of task relevant data in a concise and summative manner, presenting interesting general properties of the data. Concept description consists in characterization and comparison or discrimination. There are two general approaches to concept characterization: the data cube on-line analytical processing approach and the attribute oriented induction approach. Both are approaches based on attribute or dimension.

For a data mining system there are some primitives like *task relevant data, the kind of knowledge to be mined, background knowledge, interestingness measures,* and *presentation and visualization of discovered patterns* [JaiMic01,Sou00].

Task relevant data – specifies a portion of the database that is investigated, because usually the user is interested only in a subset of the dataset. The difficult task for this step is to specify the relevant attributes or dimensions involved in the problem. Users have only a rough idea of what the interesting attributes for exploration might be.

The kind of knowledge to be mined – specifies the data mining functions to be performed, such as characterization, discrimination, association, classification, clustering, and evolution analysis. In this step the user can specify and provide more pattern templates that all discovered patterns must match. These templates can be used to guide the discovery process.
Background knowledge – some knowledge about the domain to be mined is specified for guiding the knowledge process of discovery. A powerful form of background knowledge is known as *concept hierarchies*. They allow the discovery of knowledge at multiple levels of abstraction. This is represented like a set of nodes organized in a tree, where each node represents a concept.

Interestingness measure – is used to separate uninteresting pattern from knowledge. This is used to guide the process of mining or to evaluate the discovered patterns. This can reduce substantially the number of patterns, typically only a small fraction of these patterns will actually be interesting to the given user. There are some objective measures of pattern interestingness. Some are based on the structure of patterns and some are based of the statistic underlying them. Generally, each measure is associated with a threshold that can be controlled by the user. One measure is *simplicity*. This can be viewed as a function of the pattern structure, defined in terms of pattern size in bits, or the number of attributes or operations appearing in the pattern. Another measure is *certainty* where each discovered pattern has a measure of certainty associated with it that assesses the validity or "trustworthiness" of the pattern. A measure of certainty for the rules "A=>B", where A and B are sets of items, is confidence:

$$\text{confidence}(A \Rightarrow B) = \frac{\#\_\text{tuples\_containing\_both\_A\_and\_B}}{\#\_\text{tuples\_containing\_A}} \qquad (2.8)$$

Another measure is *utility*. It measures the potential usefulness or interestingness of the pattern. It can be estimated by a utility function, such as support. The support associated to a pattern refers to the percentage of task-relevant data tuples for which the pattern is true:

$$\text{support}(A \Rightarrow B) = \frac{\#\_\text{tuples\_containing\_both\_A\_and\_B}}{\text{total\_\#\_of\_tuples}} \qquad (2.9)$$

The last measure of interestingness presented here is the *novelty*. The novelty bringing patterns are those that contribute with new information or increase performance to the given pattern set. For example data exception may be considered novelty in that it differs from the data expected based on the statistical model or user belief. Another strategy in detecting novelty is to remove redundant patterns (a discovered rule can be implied by another rule that is already in the knowledge based or in the derived rule set).

Presentation and visualization of discovered patterns – is the form in which the discovered patterns are to be displayed. The visualization of discovered patterns in various forms can help users with different backgrounds to identify patterns of interest and to interact or guide the system in future discovery.

There are some algorithms of data analysis grouped in four categories. They are grouped according to the idea of the algorithm used for mining: *mining association rules, classification, prediction,* and *clustering*.

2.1.3 Mining association rules

Mining association rules consists of first finding frequent item-set (set of items such as A or B, satisfying a minimum support threshold, or percentage of task-relevant tuples), from which strong association rules in the form of A =>B are generated. These rules also satisfy a minimum confidence threshold. Association rules can be classified into several categories based on different criteria. The association rules can be classified by the type of values handled in the rule into Boolean or quantitative. The association rules can be classified by dimension into single dimension or multidimensional. The association rules can be classified by the level of abstraction involved in the rule, into single level or multilevel (where different attributed are at different levels of abstraction). For mining association rules there are some classical algorithms like *Apriori, Frequent Pattern Growth, Multilevel Association Rules,* and *Constraint Based Rule Mining* (for details see [JaiMic01]).

The *Apriori* algorithm explores the level-wise mining an apriori property (all nonempty subsets of a frequent item-set must also be frequent). The name of the algorithm is based on the fact that the algorithm uses prior knowledge and is based on the iterative approach known as a level-wise search. In this approach every k-itemsets are used to explore (k+1)-itemsets. At first we compute the set of 1-itemsets, denoted L_1 that contains the frequency of each item from the set. Using L_1 we create the 2-itemsets frequency set that contains all pairs of itemsets that occur together in the set. L_2 is used to find L_3, and so on, until no more frequent k-itemsets can be found. For finding each L_k set we need a full scan of the database. The Apriori algorithm has a basic property that says that all nonempty subsets of frequent itemsets must also be frequent. As follows I will present an example for the Apriori algorithm.
For example consider the next transactional database:

TID	List of items
T100	I1,I2,I5
T200	I2,I4
T300	I2,I3
T400	I1,I2,I4
T500	I1,I3
T600	I2,I3
T700	I1,I3
T800	I1,I2,I3,I5
T900	I1,I2,I3

Table 2.2 Transactional database

In this table we have 9 transactions and 5 different items. The steps for the Apriori algorithm are:

1. For the first iteration each item is a member of the candidate set of 1-itemsets denoted C_1. The algorithm scans all transactions in order to count the number of occurrences for each item.
2. Let 2 be the threshold value that represents the minimal transaction support count. The set L_1 of frequent 1-itemsets is created taking only the candidates that satisfy the threshold.
3. To discover the set of frequent 2-itemsets, L_2, the algorithm uses $L_1 \bowtie L_1$ to generate candidates set denoted C_2. $L_1 \bowtie L_1$ is equivalent to L_1 x L_1 and requires the two joining itemsets to share items and the C_2 consists of $\binom{|L_1|}{2}$ 2-itemsets.

4. The transactions are scanned and the support count of each candidate itemset from C_2 is updated.
5. The set of frequent itemset, L_2, is then determined from set C_2 considering only those candidates that satisfy the threshold.
6. To discover the set of frequent 3-itemsets, L_3, the algorithm uses $L_2 \bowtie L_2$ to generate the new candidates set C_3. The set L_3 is then determined from C_3 considering only those candidates that satisfy the threshold.
7. The algorithm uses $C_4 = L_3 \bowtie L_3$ to generate a candidate set of 4-itemsets. All new candidates are pruned since its subset is not frequent. Thus L_4 is an empty set, and the algorithm terminates, having found all the frequent itemsets.

In the next figure we present the evolution of the Apriori algorithm for the given transactional database.

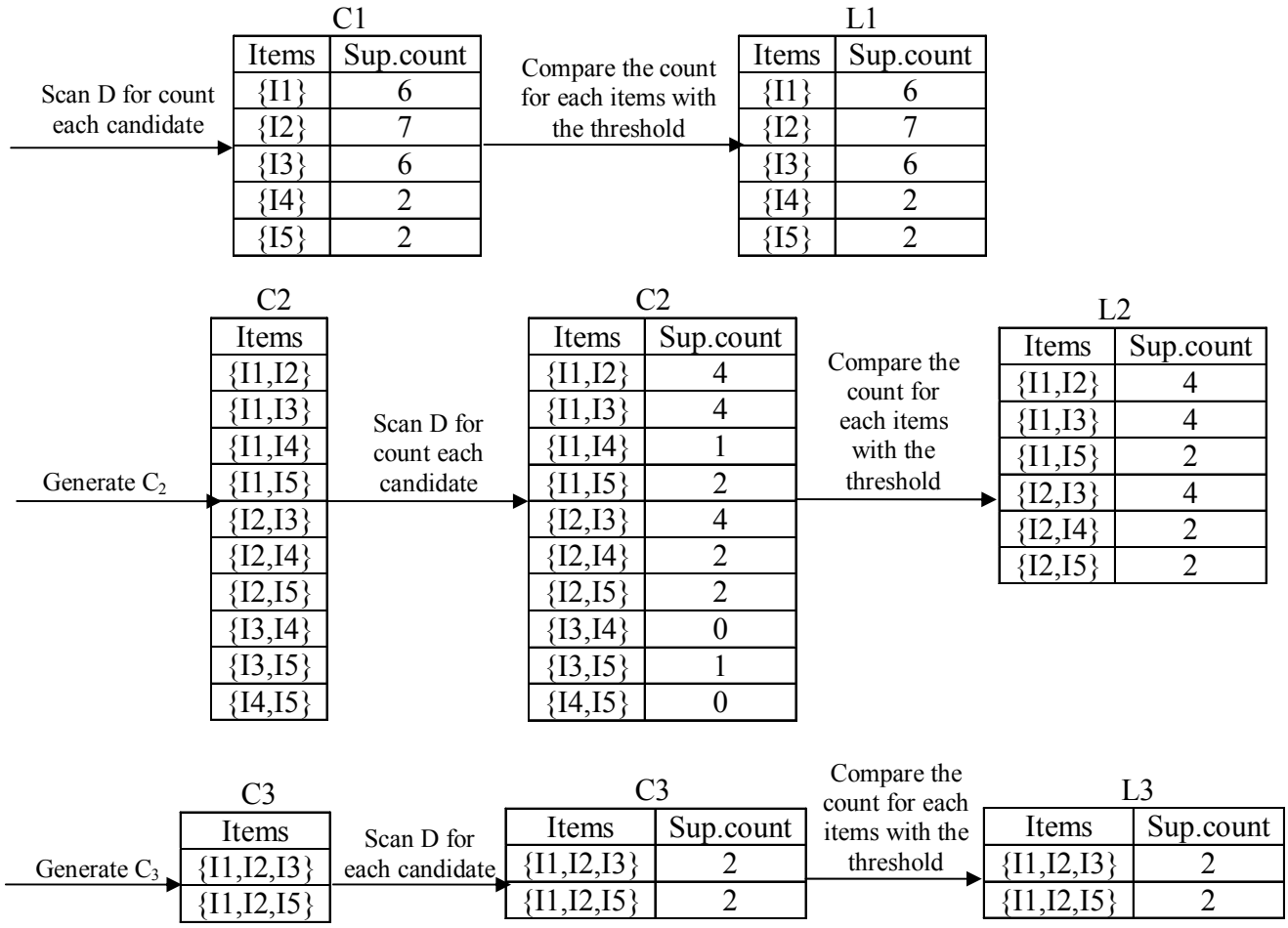

Figure 2.1 Generation of candidate itemsets and frequent itemsets.

Once the frequent itemsets from the transaction have been found, it is straightforward to generate strong association rules from them (that satisfy both minimum support and minimum confidence presented in 2.1.2) and take in consideration only the frequent itemsets that satisfy minimum confidence threshold. This can be done using the formulas for confidence and support presented earlier (2.8 and 2.9). For each frequent itemsets are generated all nonempty subsets, and for every nonempty subset are generated rules that have the minimum confidence threshold. Since the rules are generated from frequent itemsets, each one automatically satisfies the minimum support. As an

example suppose the data contain the frequent itemset L={I1, I2, I5}. The nonempty subsets of L are {I1,I2}, {I1,I5}, {I2,I5}, {I1}, {I2}, and {I5}. The resulting association rules are:

$I1 \wedge I2 \Rightarrow I5, \quad confidence = 2/4 = 50\%$

$I1 \wedge I5 \Rightarrow I2, \quad confidence = 2/2 = 100\%$

$I2 \wedge I5 \Rightarrow I1, \quad confidence = 2/2 = 100\%$

$I1 \Rightarrow I2 \wedge I5, \quad confidence = 2/6 = 33\%$

$I2 \Rightarrow I1 \wedge I5, \quad confidence = 2/7 = 29\%$

$I5 \Rightarrow I1 \wedge I2, \quad confidence = 2/2 = 100\%$

If the minimum confidence threshold is, say, 70%, then only the second, third, and last rules above are output, since these are the only generated rules that are also strong.

A great disadvantage of this algorithm is that it needs to scan all transactions from the database to generate each candidate, and for large databases this is time consuming. For this algorithm there are different variations for reducing the time, like partitioning the data (mining on each partition and then combining the results), and sampling the data (mining on a subset of data). The *frequent pattern growth* (*FP-growth)* is a method of mining frequent item-sets without candidate generation. It constructs a highly compact data structure to compute the original transaction database.

In the Frequent Pattern growth (FP-growth) algorithm we adopt a *divide and conquer* strategy: compress the database representing frequent items into a frequent pattern tree (FP-tree) retaining the itemset association information. Then divide such a compressed database into a set of conditional databases (a special kind of projected database), each associated with one frequent item.

To illustrate this, consider the transactional database presented in Table 2.2. Scan the database like in the Apriori algorithm which counts the set of frequent items (1-itemsets) and their support counts (frequencies). Let 2 be the minimum support count (the threshold). The obtained set is sorted descending by support count, resulting a set denoted L = [I2:7, I1:6, I3:6, I4:2, I5:2]. Using this set the FT-tree is constructed as follows. We create the root of the tree, labeled with “null”. We scan the database again. The items in each transaction are processed in the order that they appear in L and a branch is created for each transaction. We first take the transaction “T100: I1, I2, I5”. We put the items in L order {I2, I1, I5} and leads to the construction of the first branch of the tree with three nodes {(I2:1), (I1:1), (I5:1)}. The node I2 is linked as a child of the “root”, the node I1 is linked to I2 and the node I5 is linked to I1. We take the next transaction T200 that contains the items I2 and I4 in L order, which would result in a branch where I2 is linked to the root and I4 is linked to I2. However, this branch would share a common prefix {I2}, with the existing path for T100. Therefore, we increment the count of the I2 node by 1, and create a new node {I4:1}, which is linked as a child of I2. When considering the branch to be added for a transaction, the count of each node along a common prefix is incremented by 1, and nodes for the items following the prefix are created and linked accordingly. In this manner all transactions from the database are analyzed. The FR-tree is summarized in Figure 2.2.

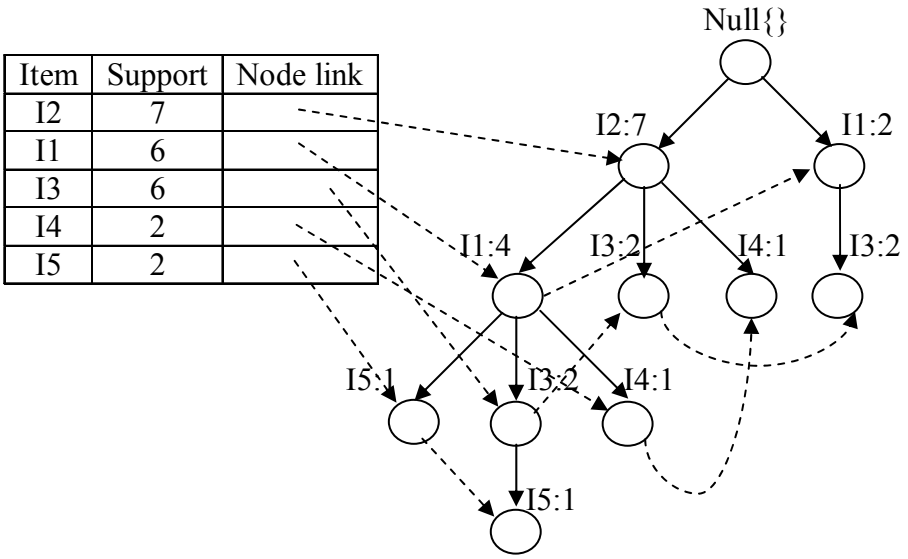

Figure 2.2 A FP-tree

In the next step the patterns are extracted from the FP-Tree as follows. Let's first consider I5 which is the last item in L, rather than the first. I5 occurs in two branches of the FP-tree. The paths formed by these branches are {(I2 I1 I5:1), (I2 I1 I3 I5:1)}. Considering I5 as a suffix, we have two prefix paths which form its conditional pattern base. Its conditional FP-tree contains only a single path [I2:2, I1:2]; I3 is not included because its support count of 1 is less than the threshold. The single path generates all the combinations of frequent patterns: [I2 I5:2, I1 I5:2, I2 I1 I5:2]. This is done for all items. The process is summarized in Table 2.3.

item	conditional pattern base	conditional FP-tree	frequent patterns generated
I5	{(I2 I1:1), (I2 I1 I3:1)}	{I2:2, I1:2}	{(I2 I5:2), (I1 I5:2), (I2 I1 I5:2)}
I4	{(I2 I1:1), (I2:1)}	{I2:2}	{(I2 I4:2)}
I3	{(I2 I1:2), (I2:2), (I1:2)}	{(I2:4, I1:2), (I1:2)}	{(I2 I3:4), (I1 I3:4), (I2 I1 I3:2)}
I1	{(I2:4)}	{I2:4}	{(I2 I1:4)}

Table 2.3 Mining the FP-tree by creating conditional (sub)pattern bases

The Fp-growth method transforms the problem of finding long frequent patterns to looking for shorter ones recursively and then concatenating the suffix. This method is less time consuming comparing with the Apriori algorithm for huge databases.

After finding the frequent itemsets from the transactional database, it is straightforward to generate strong association rules from them (that satisfy both the minimum support and the minimum confidence) (like in Apriori example presented above).

Multilevel association rules can mine using several strategies. These are based on how the minimum support threshold is defined at each level of abstraction. *Constraint based rules* mining allows users to focus the search by providing meta-rules and additional mining constraints.

2.1.4 Classification and prediction

Classification and prediction are two forms of supervised data analysis that can be used to extract models describing important data classes or to predict future data trends. While classification

predicts categorical labels, the prediction models continuous valued functions. Data classification is a two-step process. In the first step a model is built, based on predetermined set of data classes or concepts, by analyzing database samples described by the attributes. Each sample is assumed to belong to a predefined class. Since the class label of each training sample is provided this step is also known as supervised learning. The learned model is represented as classification rules, decision trees, et al. In the second step the model is used for classification. First the predictive accuracy of the model is estimated. The accuracy of a model on a given test set is the percentage of test set samples that are correctly classified by the model. If the accuracy of the model is considered acceptable, the model can be used to classify future data samples for which the class label is not known. Prediction can be viewed as the construction and usage of a model to assess the class of an unlabeled sample, or to assess the value or value ranges of an attribute that a given sample is likely to have. A commonly accepted view in data mining is to refer to the use of prediction to predicted class labels as *classification,* and the use of prediction to predict continuous values as *prediction*. There are five criteria for the evaluation of classification and prediction methods like *predictive accuracy* (refers to the ability of the model to correctly predict the class label of new or previously unseen data), *computational speed* (refers to the computational costs involved in generating and using the model), *robustness* (the ability of the model to make correct prediction given noisy data or data with missing values), *scalability* (refers to the ability to construct the model efficiently given large amounts of data) and *interpretability* (refers to the level of understanding and insight that is provided by the model). There are some common algorithms used for data classification and prediction [Ian00].

ID3 and C4.5 are the two greedy algorithms for the induction of the decision trees. Each algorithm uses theoretically an information measure to select the attribute tested for each non-leaf node in the tree. Pruning algorithms attempt to improve accuracy by removing tree branches reflecting noise in the data. Decision trees can be easily converted to classification IF-THEN rules.

Naive Bayesian classification and Bayesian belief networks - are based on Bayes theorem of posterior probability. Like in naive Bayesian classification (which assume class conditional independence), Bayesian belief networks allow class conditional independencies to be defined between subset of variables.

Backpropagation [Mit97] is a neural network algorithm for classification using supervised learning that employs a method of gradient descent. It searches for a set of weights that can model the data so as to minimize the mean squares distance between the network's class prediction and the actual class label of data samples. Rules may be extracted from trained neural networks in order to help improve the interpretability of the learned network.

Nearest neighbor classifier and case based reasoning classifier are methods based on instances of classification in that they store all of the training samples in the pattern space. In genetic algorithms, populations of rules "evolve" via operations of crossover and mutation until all rules within a population satisfy classes that are not distinguishable based on the available attributes. Fuzzy set approaches replace "brittle" threshold cutoffs for continuous valued attributes with degree of membership functions.

Linear, nonlinear, and generalized linear models of regression can be used only for prediction. In linear regression, data are modeled using a straight line. Linear regression models a random variable, Y (called a response variable), as a linear function on another random variable, X (called a predictor variable). Many nonlinear problems can be converted to linear problems by performing transformations on the predictor variables.

Stratified *k* fold cross-validation is a recommended method for estimating classifier accuracy. Bagging and boosting methods can be used to increase overall classification accuracy by learning and combining a series of individual classifiers. Sensitivity, specificity, and precision are useful alternatives to the accuracy measure, particularly when the main class of interest is in the minority.

2.1.5 Clustering

Clustering is the unsupervised process of grouping the data into classes (or clusters) so that objects within a class (cluster) have high similarity, but are very dissimilar in comparison with the objects from other classes (clusters). Dissimilarity is assessed based on the attribute values describing the objects. Cluster analysis can be used as a standalone data mining tool to gain insight into the data distribution, or serve as a preprocessing step for other data mining algorithms in the detection of clusters. Clustering is a dynamic field of research in data mining. Many clustering algorithms have been developed. These can be categorized into *partitioning methods, hierarchical methods, density based methods, grid-based methods and model-based methods* [JaiMic01].

A *partitioning method* first creates an initial set of *k* partitions, where *k* is the arbitrary number of partitions to be constructed; then it uses an iterative relocation technique that attempts to improve the partitioning by moving objects from one group to another. On the other hand the *hierarchical method* creates a hierarchical decomposition of given set of data. The method can be classified as being either agglomerative (button-up) or divisive (top-down), based on how the hierarchical decomposition is formed. To compensate for the rigidity of merge or split, the quality of hierarchical agglomeration can be improved by analyzing object linkages at each hierarchical partition or by integrating other clustering techniques, such as iterative relocation. A *density based method* clusters objects based on the notion of density. It either grows clusters according to the density of the neighborhood objects or according to some density function. A *grid based method* first quantifies the object space into a finite number of cells that form a grid structure, and then performs clustering on the grid structure. A *model based method* creates a model for each of the clusters and finds the best fit of the data to that model. Typically model based methods involve statistical approaches or neural network approaches (such as competitive learning and self organizing feature maps). For more details related to there clustering methods see [JaiMic01,Ian00]

2.2 Text mining

As we presented by now data mining is about looking for patterns in a database that is considered to be structured data. In reality a substantial portion of the available information is stored in text, which consists of large collections of documents from various sources, such as news articles, research papers, books, digital libraries, e-mail messages and web pages. Data stored in text format is considered semi-structured data in that they are neither completely unstructured nor completely structured, because a document may contain a few structured fields such as title, authors, publication, data, et al. Unfortunately these fields usually are not filled in, the majority of people don't loose time to complete these fields. Some researchers suggest that the information needs to be organized during the creation time, using some planning rules. This is pointless because most people will not respect them. This is considered a characteristic of unordered egalitarianism of Internet [Der00]. Any attempt to apply the same organizing rules will determine the users to leave. The result for the time being is that most information needs to be organized after it was generated, and searching and organizing tools need to work together.

From enormous quantity of information, only a small fraction will be relevant to the user. Thus, users need tools to be able to compare different documents, rank the importance and relevance of the documents, or find patterns and trends across multiple documents. Without knowing what could be in the documents, it is difficult to formulate effective queries for analyzing and extracting useful information from data. Thus, text mining has become an increasingly popular and essential theme in data mining.

2.2.1 Text data analyzing and information retrieval

Information retrieval (IR) is a field developed in parallel with database systems. Information retrieval is concerned with the organization and retrieval of information from a large number of text-based documents. A typical information retrieval problem is to locate relevant documents based on user input, such as keywords or example documents. Usually information retrieval systems include on-line library catalog systems and on-line document management systems. Since information retrieval and database systems each handle different kinds of data, there are some database system problems that are usually not present in information retrieval systems such as concurrency control, recovery, transaction and management. There are also some common information retrieval problems that are usually not encountered in traditional database systems, such as unstructured documents, approximate search based on keywords and the notion of relevance.

2.2.1.1 Basic measures for text retrieval

May [Relevant] be the set of documents relevant to a query and [Retrieved] be the set of documents retrieved. The set of documents that are both relevant and retrieved is denoted by $[\text{Relevant}] \cap [\text{Retrieved}]$. There are tow basic measures for assessing the quality of text retrieval:

- **Precision**: is the percentage of retrieved documents that are in fact relevant to a query. It is defined as follows:

$$precision = \frac{|\{\text{Relevant}\} \cap \{\text{Retrieved}\}|}{|\{\text{Retrieved}\}|} \quad (2.10)$$

- **Recall**: is the percentage of documents that are relevant to the query and were in fact retrieved.

$$recall = \frac{|\{\text{Relevant}\} \cap \{\text{Retrieved}\}|}{|\{\text{Relevant}\}|} \quad (2.11)$$

2.2.1.2 Keyword-Based and Similarity-Based retrieval

Most information retrieval systems support *keyword-based* and *similarity-based* retrieval. In keyword-based information retrieval, a document is represented by a string, which can be identified by a set of keywords. A user provides a keyword or an expression formed out of a set of keywords, such as “car and repair shop”. A good information retrieval system needs to consider synonyms when answering such query. This is a simple model that can encounter two difficulties: (1) the *synonyms* problem, keywords may not appear in the document, even though the document

is closely related to the keywords; (2) the *polysemy* problem: the same keyword may mean different things in different contexts.

The information retrieval system based on similarity finds similar documents based on a set of common keywords. The output for this system is based on the degree of relevance measured based on using keywords closeness and the relative frequency of the keywords. In some cases it is difficult to give a precise measure of the relevance between keyword sets. The information retrieval system often associates a stop list with the set of documents. A stop list is a set of words that are deemed "irrelevant" and can vary when the document set varies. Another problem that appears is *stemming.* A group of different words may share the same word stem. A text retrieval system needs to identify groups of words where the words in a group are small syntactic variants of one another, and collect only the common word stem per group.

In information retrieval systems the document is modeled based on the frequency of the words of the document. Starting with a set of *d* documents and *t* terms, we can model each document as a vector υ in *t* dimensional space $\mathbb{R}^t$. The coordinates of υ are numbers that measure the association of the terms with respect to the given document. It is generally defined as 0 if the document does not contain the term, and nonzero otherwise. υ[i] can indicate the frequency of the term in the document. Similar documents are expected to have similar relative term frequency, and we can measure the similarity among a set of documents or between a document and a query. There are many metrics for measuring the document similarity. One is cosine similarity defined as:

$$sim(v_1,v_2) = \frac{v_1 \cdot v_2}{\|v_1\|\|v_2\|} \qquad (2.12)$$

where $v_1 \cdot v_2$ is the standard dot products defined as $\sum_{i=1}^{t} v_{1i} v_{2i}$ and $\|v_1\|$ is defined as $\|v_1\| = \sqrt{v_1 \cdot v_1}$.

To illustrate this, suppose S is a collection of 12 samples like that presented in Table 2.1. Each row represents a document vector, and each entry (attribute) represents the frequency of the word in the document. The similarities of three vectors, 2 from same class (v_1, v_2) and one from another class (v_4), using formulas presented early are:

$$\|v_1\| = \sqrt{1^2 + 5^2 + 7^2 + 0^2 + 1^2 + 1^2} = 8{,}7749$$

$$\|v_2\| = \sqrt{1^2 + 1^2 + 0^2 + 0^2 + 7^2 + 5^2} = 8{,}7177$$

$$\|v_4\| = \sqrt{1^2 + 3^2 + 7^2 + 1^2 + 0^2 + 1^2} = 7{,}8102$$

$$\langle v_1 \cdot v_2 \rangle = 1 \cdot 1 + 5 \cdot 1 + 7 \cdot 0 + 0 \cdot 0 + 1 \cdot 7 + 1 \cdot 5 = 18$$

$$\langle v_1 \cdot v_4 \rangle = 1 \cdot 1 + 5 \cdot 3 + 7 \cdot 7 + 0 \cdot 1 + 1 \cdot 0 + 1 \cdot 1 = 66$$

The similarities between documents according to 2.12 are:

$$sim(v_1,v_2) = \frac{18}{8.7749 * 8.7177} = 0.2353$$

$$sim(v_1,v_4) = \frac{66}{8.7749 * 7.8102} = 0.9630$$

The similarity ranges from 1 (perfectly similar), to 0 (dissimilar). Great values of similarity represent a small angle between vectors and therefore the vectors (the documents) are similar. Resulting thus the documents v_1 and v_2 are considered to be similar and v_1 and v_4 are considerate dissimilar.

Each document from the set can be represented as a vector. Using similarity we can construct a similarity-based index, and when a query occurs, it is represented like a vector. This vector is used to search for their nearest neighbors in a document collection. The problem in this representation is that usually the dimension of the set of documents and the number of terms is quite large and leads to the problem of inefficient computation. On the other hand high dimensionality leads to a very sparse vector and increases the difficulty to detecting relationships between terms. A method called *latent semantic indexing* was developed to reduce the size of the frequency table for analysis.

2.2.1.3 Latent semantic indexing

It is used for reducing the size of frequency matrix. This method uses *singular value decomposition (SVD)*, a well-known technique in matrix theory. Given a t × d term frequency matrix representing *t* terms and *d* documents, the SVD method removes columns to reduce the matrix to size k × d, where *k* is usually taken to be around a few hundred for large document collections. To minimize the amount of information loss, only the least significant parts of the frequency matrix are omitted. The latent semantic indexing method consists of the following basic steps:

1. Create a term frequency matrix, *frecvency_matrix*. Compute singular value decomposition of *frequency_matrix* by splitting the matrix into three smaller matrices, *U, S, V*, where U and V are orthogonal matrices ($U^TU = I$ and $V^TV = I$), and S is a diagonal matrix of singular values. Matrix U have size *t* x *k*, the V matrix have size *k* x *d*, and matrix S is of size *k* ×*k*.
2. For each document *d*, replace its original document vector by a new one that excludes the terms eliminated during SVD
3. Store the set of all vectors, and create indexes for them using advanced multidimensional indexing techniques.

Most matrixes that represent a set of documents are sparse matrixes, having most of elements equal to 0. A better representation in the memory of those matrixes is to keep only the values there are greater than 0 and the position of those values for each line of the matrix. This allows us to store the *frequency_matrix* without lose (for details see [cs.utk]).

There are other techniques for text retrieval like invert index or signature files. The *invert index* technique is an index structure that maintains two hash indexed or B+ tree indexed tables: *document_table* and *term_table*. The document_table contains a set of document records, each containing two fields: *doc_id* and *posting_list* (list of terms) that occur in the document, sorted according to some relevance measure. The *term_table* contains a set of term records, each containing two fields: *term_id* and *posting_list* (specifies a list of document identifiers in which the term appears).

The *signature file* is a file that stores a signature record for each document in the database. Each signature has a fixed size of *b* bits representing terms. Each bit of a document signature is initialized to 0. A bit is set to 1 if the term it represents appears in the document. A signature S_1 matches another signature S_2 if each bit that is set in signature S_2 is also set in S_1.

2.2.2 Keyword-based association

Association analysis first preprocesses the text data by parsing, stemming, removing stop words and so on, and then evokes association mining algorithms. In a document database, each document can be viewed as a transaction, while a set of keywords in the document can be considered as a set of items in the transaction. That is, the database format is:
{document_id, a_set_of_keywords}.

The problem of keyword association mining in document databases is thereby mapped to item association mining in transaction database, where many interesting methods have been developed (like Apriori). Notice that a set of frequencies occurring consecutively or closely located keywords may form a *term* or a *phrase*. This process can help detect *component associations*, that is, domain-dependent terms or phrases, such as {Stanford, University}, or *noncompound associations*, such as {dollars, shares, exchange}. Mining based on these associations is referred to as "*term level association mining*" (as opposed to mining on individual words). Term recognition and term level association mining have two advantages in text analysis: (1) terms and phrases are automatically tagged so that there is no need for human effort in tagging documents, and (2) the number of meaningless results is greatly reduced, as is the execution time of mining algorithms. Mining term level can be used to find associations among a set of detected terms and keywords. It is interesting to find associations between pairs of keywords or terms from a given set of keywords or phrases or find the maximal set of terms occurring together.

2.2.3 Document classification analysis

There are an increasing number of online documents and an automated document classification is an important task. It is essential to be able to automatically organize such documents into classes so as to facilitate document retrieval and analysis. One possible general procedure for this classification is to take a set of pre-classified documents and consider them as the training set. The training set is then analyzed in order to derive a classification scheme. Such a classification scheme often needs to be refined with a testing process. After that this scheme can be used for classification of other on-line documents. The classification analysis decides which attribute-value pairs set has the greatest discriminating power in determining the classes. An effective method for document classification is to explore association-based classification, which classifies documents based on a set of associations and frequently occurring text patterns. Such an association-based classification method proceeds as follows: (1) keywords and terms can be extracted by information retrieval and simple association analysis techniques; (2) concept hierarchies of keywords and terms can be obtained using available term classes, or relying on expert knowledge or some keyword classification systems. Documents in the training set can also be classified into class hierarchies. A term-association mining method can then be applied to discover sets of associated terms that can be used to maximally distinguish one class of documents from another. This produces a set of association rules for each document class. Such classification rules can be ordered - based on their occurrence frequency and discriminative power - and used to classify new documents.

Usually the key phrases do not come from a fixed vocabulary, but are likely to be phrases that occur in the text of the document itself. Maybe only a smaller training set is available – after all, assigning phrases to documents involves expensive intellectual labor. This technique is inapplicable because it can only form models for the key phrases that are used in the training data. But instead the key phrases could be chosen from the text itself. Given a document, rudimentary

lexical techniques based on punctuation and common words could be used to extract a set of candidate key phrases. Then, features could be computed for each phrase, like how often it appears in that document, how close to the beginning of the document it first occurs, how often it has been used as a key phrase in other training documents, whether it occurs in the title, abstract or section headings, whether it occurs in the title of the paper cited in the reference list, and so on.

2.3 Web mining

Web is a huge service which offers a lot of information [LawGil99]. The web also contains a rich and dynamic collection of hyperlink information and web page access and usage information, providing rich sources for data mining. The mining of web pages is so different from mining of text documents. The web is too huge and is still growing rapidly, the information stored on the web is continuously updated and the web pages contain far more authoring style and content variations than any set of books or other traditional text based documents. Another problem for web mining is that the web serves a diversity of user communities, and the users may have very different backgrounds, interests, and usage purposes. When users want to find something on the web only a small fraction of the information on the web is relevant or useful to the current user. These challenges have promoted research into efficient discovery and use of resources on the Internet. There are many index-based Web engines that search the web, index pages, build and store huge keyword-based indexes. These indexes help locate sets of Web pages containing certain keywords. Thus an experienced user can be able to quickly locate documents by providing a set of keywords and phrases. As we previously pointed out, this type of search engines based on keywords suffers from several deficiencies. One is because the topic can contain a huge number of document entries returned by the search engine. Many of these pages can have low relevance with what the user wants, or can contain a poor quality. Another deficiency is that many relevant pages for the user's search don't contain the keywords defining term. This is referring to as the polysemy problem, presented earlier in the text mining section. The web mining is the most challenging task that searches for web access patterns, web structures and the regularity and dynamics of web contents.

2.3.1 Automatic classification on Web documents

In the automatic classification of Web documents, each document is assigned to a class label from a set of predefined topic categories, based on a set of examples of pre-classified documents. For example Yahoo taxonomy from the net and its associated documents can be used as training and test sets in order to construct a Web document classification scheme and this scheme can be used after that for classifying new Web documents by assigning categories from the same taxonomy. This method might be useful for supervised learning classification.

The method for classifying documents using words from the document can be used for classifying Web documents. This scheme for classifying based on terms obtains good results for web documents classifying. Because hyperlink contain a solution of good quality from the semantic point of view in comparison with the topic of the page, it is better to use this semantic information in order to achieve even better accuracy than pure keywords based classification. However, since the hyperlink surrounding a document may be quite noisy, naive use of terms in a documents hyperlink neighborhood can even degrade accuracy of classifying.

2.3.2 Web mining categories

Data mining consist of three steps: *preprocessing data, pattern discovery and pattern analysis*. Similarly, web mining has three parts which can contain the above three steps:

- Web content mining – text mining for the web page content (the real data from the Web pages) and metadata given by tags in the html files.
- Web structure mining - data which describes the organization of the content and of the site
- Web usage mining – data that describes the pattern of link usage on Web pages.

Alternatively, the structure mining of the Web can be treated like pieces of web's content mining, for this, mining the web can instead be simply classified in content mining and usage mining.

Web mining is the application of data mining techniques to the content, structure, and usage of web resources. This can help to discover global, as well as local structure within and between web pages. Like other data mining applications, web mining can profit from given structure of data, but it can also be applied to semi-structured or unstructured data like free-form text. This means that web mining is an invaluable help in the transformation from human understandable content to machine understandable one.

2.3.2.1 Web content mining

Web content mining is a form of text mining. The primary resource of the web is the individual page. Web content mining can take advantage of the semi or structured nature of web pages text. For this there are more techniques for information retrieval from text documents, like methods for indexing a text that were developed to work with unstructured (semi-structured) documents. Some traditional techniques for information retrieval became inadequate for an extensive amount of data. Without knowing what is in the document it is difficult to formulate querying for analyzing and extracting interesting information. The user needs tools for comparing different documents. Some important tools are rank and relevance of the document or finding patterns and direction for more documents. Web content mining describes the discovery of useful information from the Web contents/data/documents. Some of the web content data are hidden data, which cannot be indexed. These data are either generated dynamically as a result of queries and reside in the database or are private. The web already contains many kinds and types of data such as textual, image, audio, video, metadata as well as hyperlinks. Thus we could consider multimedia data mining as an instance of web content mining. The web content data consists of unstructured data such as free texts, semi-structured data such as HTML documents, and more structured data such as data in the tables or databases generated by HTML. According to [RayHen00], we could differentiate the research done in web content mining from two different points of view: *information retrieval* and *database* views.

The goal of web content mining from the information retrieval view is mainly to assist or to improve the information finding or filtering through the users inferred or solicited profiles. Most of the applications use sets of words (in literature referred as bags of words) to represent unstructured documents. The bag of words or vector representation takes single words found in the training set as features. This representation ignores the sequences in which the words occur and is based on the statistics about single words isolation. The feature could be Boolean (a word either occurs or not in a document), or frequency based (frequency of the word in a document). The features could be reduced further by applying some other feature selection techniques, such as information gain,

mutual information, cross entropy, odds ratio [MlaGro99], χ^2 statistic or Term strength [YanPed97].

Other preprocessing includes Latent Semantic Indexing as we briefly presented in paragraph Latent semantic indexing [DeeDum90] [Hof99]. The preprocessing variations are useful for reducing feature set size. In general they are highly effective over different domains for text categorization. Other feature representations are also possible such as: using information about word position in the document, using n-grams representation (word sequences of length up to n), using phrases, using document concept categories, using terms, using hypernyms, etc. Currently the term text mining has been used to describe different applications such as text categorization, text clustering, empirical computational linguistic tasks, exploratory data analysis, finding patterns in text databases, finding sequential patterns in texts, and association discovery. Information retrieval for semi-structured documents is due to the additional structural (HTML) information in the hypertext documents. Actually all of the works surveyed use the hyperlinks structure between the documents for document representation. The methods that are used are common data mining methods.

The database techniques on the web are related to the problems of managing and querying the information on the web. This was presented in the data mining section.

2.3.2.2 Web structure mining

Web structure mining usually operates on the hyperlink structure of web pages. The primary web resource that is being mined is the set of pages, ranging from a single web site to the web as a whole. Web structure mining explores the additional information that is contained in the structure of hypertext. An important application area is the identification of the relative relevance of different pages that appear equally pertinent when analyzed with respect to their content in isolation. The web structure mining can be used to find authoritative web pages and identify hubs (presented summary further in section 2.3.2.2.3) or see details in [Cha03]. According to Cooley, Mobasher and Srivastava [CooMob99] there are five types of web pages:

- *head* pages that are entry points to a site
- *navigation pages that contain many links* and pour content in information
- *content* pages that contain a small number of links but are rich in information
- *look-up* pages that have many incoming links, few outgoing ones and no significant content, such as pages used to provide a definition or acronym expansion
- *personal* pages that have diverse characteristics and no significant traffic

Web structure mining tries to discover the model underlying the link structure of the web. The model is based on the topology of the hyperlinks with or without the description of the links. The model can be used to categorize web pages and is useful to generate information such as the similarity and the relationship between different web sites. Web structure mining could be used to discover authoritative sites regarding the subject and overview sites regarding the subject, sites that point to many authoritative pages (hubs). This line of research is inspired by the study of social networks and citations analysis. With social network analysis we could discover specified types of pages (such hubs, authorities) based on the incoming and out coming links. Web structure mining uses the hyperlinks structure of the web to apply social network analysis to model the underlying links structure of the web itself. Some research uses the network analysis to model the network of artificial intelligence research. They use the name of the entry data found in the close proximity of any public web pages such as the hyperlinks from home pages, co-authorship and citation of pages, exchange of information between individuals found in net-news archives, and

organization charts. Some algorithms have been proposed to model the web topology such as *Hyperlink Induced Topic Search*, and *PageRank*. These models are mainly applied as a method for computing the quality rank or relevance of each Web page.

2.3.2.2.1 Mining the Web Links

This is done in order to identify the authoritative Web pages mining relevant and high quality pages related to a given topic. The Web consists not only of pages, but also of hyperlinks pointing from one page to another. These hyperlinks contain an enormous amount of latent human annotation that can help us to automatically infer the notation of authoritative. When an author of a Web page creates a hyperlink pointing to other Web pages, this can be considered the author's endorsement of the other page. The collective endorsement of a given page by different authors on the Web may indicate the importance of the page and may naturally lead to the discovery of authoritative Web pages. Therefore, the tremendous amount of Web linkage information provides rich information about the relevance, the quality, and the structure of the Web's contents, and thus is a rich source for Web mining. The problem is that not every hyperlink represents the endorsement we seek. Some links are creating for other purposes, such as for navigation or for paid advertisement. Yet overall, if the majority of hyperlinks are for endorsement, the collective opinion will still dominate. Other problem is commercial or competitive interests, one authority will seldom have its Web page point to its rival authorities in the same field.

2.3.2.2.2 Page Rank

Page rank is used to discover the most "important" pages on the Web. In PageRank, each page on the Web has a measure of prestige that is independent of any information need or query. Roughly speaking, the prestige of a page is proportional to the sum of the prestige scores of pages linking to it. In this method all measures are defined recursively: the prestige of a node depends on the prestige of other nodes, and the measure of being a good hub depends on how good neighboring nodes are as authorities (and vice versa). This procedure involves computing eigenvectors for the adjacency matrix, or a matrix derived thereof, of the web or a suitably relevant subgraph of the web.
Let's assume for the moment that the web graph is strongly connected – that is, from any node u there is a directed path to a node v (ergodic graph). If one wanders on the Web for infinite time, following a random link out of each page with probability *1-p* and jumps to a random Web page with probability p, then different pages will be visited at different rates; popular pages with many in-links (link to that page) will tend to be visited more often. This measure of popularity is called PageRank, defined recursively as:

$$PageRank(v) = \frac{p}{N} + (1-p)\sum_{u \to v} \frac{PageRank(u)}{OutDegree(u)}, \qquad (2.13)$$

where '$\to$' means "link to" and N is the total number of nodes in the Web graph and *OutDegree(u)* is the numbers of out-links from page u (links to other pages). The Google search engine simulates such a random walk on the web graph in order to estimate PageRank, which is used as a score in pre-computed independent of the query. Hence Google can be potentially as fast as any relevance ranking search engine. Note that the PageRank is independent of any query or textual content. When a query is submitted, a text index is used to first make a selection of possible response pages. Then an undisclosed ranking scheme that combines PageRank with textual match is used to produce a final ordering of response URLs (Uniform Resource Locator). The strongest criticism of

PageRank is that it defines prestige via a single random walk uninfluenced by a specific query or by a current user's profile. A related criticism is of the artificial decoupling between relevance and quality, and the ad hoc manner in which the two are brought together at query time, for the sake of efficiency.

To illustrate this, suppose the Web has only four web pages denoted $Page_1$, $Page_2$, $Page_3$ and $Page_4$. The links among these pages are shown in Figure 2.3.

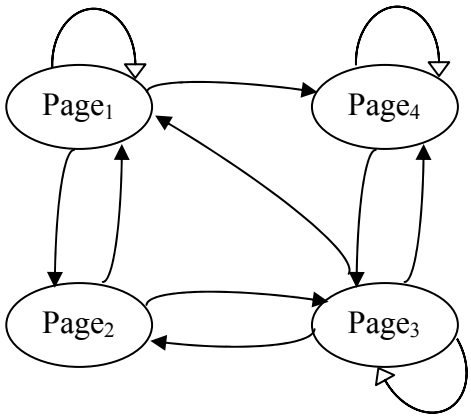

Figure 2.3 An abridge Web graph

Let $[p_1,p_2,p_3,p_4]$ be the importance vector for those pages, in that order. We can now create a stochastic matrix (sum of each column is 1) of the web where each page *i* corresponds to a row *i* and column *i* of the matrix. If the page *i* has n successors (links), then the ij^{th} entry is 1/n if the page *i* is one of these *n* successors of page *j* and 0 otherwise. The resulting matrix is:

$$\begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{bmatrix} = \begin{bmatrix} 1/3 & 1/2 & 1/4 & 0 \\ 1/3 & 0 & 1/4 & 0 \\ 0 & 1/2 & 1/4 & 1/2 \\ 1/3 & 0 & 1/4 & 1/2 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{bmatrix}$$

The estimate after three iterations is obtained as follow. First we initialize each page with same unit of importance (let it be 1). At each round, each page shares whatever importance it has among its successors, and receives new importance from its predecessors. Eventually, the importance of each page reaches a limit, which happens to be its component in the principal eigenvector of this matrix. That importance is also the probability that a web surfer, starting at a random page, and following random links from each page will be at the page in question after a long series of links. The first three iterations give the following estimates:

$$\begin{aligned} p_1 &= 1 \quad 1.08 \quad 0.962 \quad 0.9415 \\ p_2 &= 1 \quad 0.58 \quad 0.672 \quad 0.6055 \\ p_3 &= 1 \quad 1.25 \quad 1.142 \quad 1.1857 \\ p_4 &= 1 \quad 1.08 \quad 1.212 \quad 1.2275 \end{aligned}$$

Note that we can never get absolute values of $[p_1,p_2,p_3,p_4]$, just their rations, since the initial assumption that they were each 1 was arbitrary. Since the matrix is stochastic, the above relaxation process converges to the principal eigenvector.

2.3.2.2.3 Hubs

A hub is one page or a set of Web pages that provides selections of links to authorities. Hub pages may not be prominent themselves, or there may be few links pointing to them; however, they provide links to a collection of prominent sites on a common topic. Such pages could be lists of recommended links on individual home pages, such as recommended reference sites from a course home page, or professionally assembled resource list on commercial sites. In general, a good hub is a page that points to many good authorities; a good authority is a page pointed to by many good hubs. To find the authoritative pages there is an algorithm about how to use hubs, called HITS (Hyperlink-Induced Topic Search) which is presented below.

First it collects a starting set of, say, 200 pages from an index-based search engine. These pages form the root set. Since many of these pages are presumably relevant to the searched topic, some of them should contain links to most of the prominent authorities. Therefore, the root set can be expanded into a base set which includes all of the pages that the root-set pages link to, and all of the pages that link to a page in the root set, up to a designates size cutoff, such as 1000 to 5000 pages (to be included in the base set). After that a weight-propagation phase is initiated. This is an iterative process that determines numerical estimators of a hub and authority weights. Here the links between two pages with the same Web domain (e.g. sharing the same first level in their URLs) often serves as a navigation function and thus does not confer authority, such links are excluded from the weight-propagation analysis.

We first associate a nonnegative authority weight a_p and a nonnegative hub weight h_p with each page p in the base set, and initialize all a and h values with a uniform constant. The weight is normalized and an invariant is maintained that the squares of all weights sum to 1. The authority and hub weights are updated based on the following equations:

$$a_p = \sum_{(q \text{ such that } q \to p)} h_q \qquad h_p = \sum_{(q \text{ such that } q \leftarrow p)} a_q \tag{2.14}$$

where "$\to$" means "link to".
The first equation implies that if a page is pointed to by many good hubs, its authority weight should increase. The second equation implies that if a page is pointing to many good authorities, its hub weight should increase. We write these equations in matrix form as follows. Let us number the pages {1, 2… n} and define their adjacency matrix A to be an $n \times n$ matrix where A(i,j) is 1 if page i links to page j, or 0 otherwise. Similarly, we define the authority weight vector $a=(a_1, a_2, \ldots, a_n)$ and the hub weight vector $h=(h_1, h_2, \ldots, h_n)$. Thus we have:

$$h = A \cdot a \tag{2.15}$$

$$a = A^T \cdot h$$

where A^T is the transposition of matrix A. Unfolding these two equations k times, we have

$$h = A \cdot a = AA^T h = (AA^T)h = (AA^T)^2 h = \ldots = (AA^T)^k h \tag{2.16}$$

$$a = A^T \cdot h = A^T Aa = (A^T A)a = (A^T A)^2 a = \ldots = (A^T A)^k a$$

According to linear algebra, these two sequences of iterations, when normalized, converge to the main eigenvectors of AA^T and A^TA, respectively. This also proves that the authority and hub weights are intrinsic features of linked pages collected, and are not influenced by the initial weight settings.

The Hyperlink Induced Topic Search (HITS) algorithm generates a list of pages with large hub weights and also of pages with large authority weights for a given search topic. Many experiments

have shown that Hyperlink Induced Topic Search provides surprisingly good search results for a wide range of queries. Some problems occur when hubs contain multiple topics, because this algorithm ignores textual contexts. Or also cause "topic hijacking" when many pages from a single Web site point to the same single popular site, giving the site too large a share of the authority weight. Such problems can be overcome by replacing the sum from the first equations with weighted sums, scaling down the weights of multiple links from within the same site. This is done using anchor text (the text surrounding hyperlink definition in web pages).

To illustrate this we use the example presented above in Figure 2.3. The relevant matrixes are:

$$A = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \quad A^T = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix} \quad AA^T = \begin{bmatrix} 2 & 1 & 3 & 1 \\ 1 & 2 & 2 & 1 \\ 3 & 2 & 4 & 2 \\ 1 & 1 & 2 & 2 \end{bmatrix} \quad A^T A = \begin{bmatrix} 3 & 2 & 2 & 2 \\ 2 & 2 & 1 & 2 \\ 2 & 1 & 3 & 2 \\ 2 & 2 & 2 & 2 \end{bmatrix}$$

Assume that the vector $\vec{h} = [h_1, h_2, h_3, h_4]$ and $\vec{a} = [a_1, a_2, a_3, a_4]$ are each initially with [1,1,1,1] the first three iterations using the formulas for $\vec{a}$ and $\vec{h}$ are:

$$\begin{array}{llllll} a_1 & = & 1 & 9 & 73 & 589 \\ a_2 & = & 1 & 7 & 56 & 451 \\ a_3 & = & 1 & 8 & 65 & 509 \\ a_4 & = & 1 & 8 & 64 & 516 \end{array} \qquad \begin{array}{llllll} h_1 & = & 1 & 7 & 59 & 485 \\ h_2 & = & 1 & 6 & 53 & 390 \\ h_3 & = & 1 & 11 & 89 & 7333 \\ h_4 & = & 1 & 6 & 47 & 384 \end{array}$$

For instance, the vector a, properly scaled, will converge to a vector where $a_3 \approx a_4$, and each of those are greater than a_2 and lees than a_1.

2.3.2.3 Web usage mining

Web usage mining [SriCoo00] mines the data derived from users' interaction with the web. The web usage data includes the data from web server access logs, proxi server logs, browser logs, user profiles, registration data, user sessions or transactions, cookies, user queries, bookmark data, mouse clicks and scrolls, and any other data as the results of interactions. Web usage mining focused on techniques that could predict user behavior while the user interacts with the web, mining the log files (IP address), cookies or path analysis. In web usage mining resources that are mined are records of the requests made by visitors to a web site, most often collected in web server logs. The content and structure of web pages, and in particular those of one web site, reflect the intentions of those who have authored and designed those pages, and their underlying information architecture. The idea is to find a relationship that can be induced by usage where no particular structure was designed. Mining the visitors to that site, however, one may find that most of those users who were interested in product A were also interested in product B. Here "*interested*" may be measured by request for product description pages. The idea is to identify association rules between products (web pages in this case). When a new user shows interest in product "A", he will receive a recommendation for product "B".

The web usage mining process could be classified into two commonly used approaches. The first approach maps the usage data of the web server into relational tables before an adapted data mining technique is performed. The second approach uses the log data directly by using special pre-processing techniques. As this is true for typical data mining applications, here the issues of data quality and preprocessing are also very important. The typical problem is distinguishing

among unique users and server sessions. In general typical data mining methods could be used to mine the log data after the data has been preprocessed to the desired form.

The applications of web usage mining could be classified into two main categories: learning a user profile or user modeling in adaptive interfaces and learning user navigation patterns. Web usage mining would be interested in techniques that could learn necessary information about the needs and preferences. These are used to model the user profile and combine it with web content mining. On the other hand, information providers would be interested in techniques that could improve the effectiveness of the information on their web sites by adapting the web site design or by biasing the user's behavior towards satisfying the goals of the site. In other words, they are interested in learning user navigation patterns. Then the learned knowledge could be used for applications such as personalization (at a web site level), system improvement, site modification, business intelligence and usage characterization.

2.3.2.3.1 Mining log files

This type of mining processes log records for extracting user's accessing pattern to web pages. By analyzing and exploring regularity web log records one can identify potential customers for e-commerce, increase quality and deliver information services to end users and increase performance of server system from the web. Web server usually records every login (weblog), for each access to web pages. This contains URL, original IP from user log and request. This weblog provides rich information about the dynamic of web. This is important to develop new sophisticated techniques for mining the weblog.
To develop techniques for mining the web, we can consider the following. Firstly, it is very encouraging and exciting to imagine various applicable possibilities of weblog file analysis. It is important to know if the success of each application depends on how much knowledge is discovered from the data log. Data logs need to be preprocessed and transformed in order to recover and analyze significant and useful information. Secondly, besides available URL's, time and user's IP, the information about web pages content is important. A multidimensional visualization can be built using weblog and multidimensional analysis. On-Line Analytical Processing (OLAP) can be performed to find the most accessed N pages, the latest most frequently accessed and anything else to help us discover relevant data. Thirdly, data mining can be used in weblog records to find association patterns or sequential and oriented access to web. Analyzing detailed weblogs offers us the opportunity to take the necessary measures to obtain additional information from users. This additional information can include sequences about web pages visited by the user from the buffer. Data weblog provides us information about what user groups access what page groups. Weblog can be grouped with web contents mining and link structure mining to help us mine the web, classify web documents, and build a multi-level structure of basic information.

An interesting problem from the producer's point of view is trying to replace nowadays Internet advertising with a new one, addressed only to users interested in these commercials and with low costs. This onset is already applied in advertising where the specific catalogue is created centered by a specific field that is sent only to interested people. For example if a user buys a software package from a company, he will receive new proposals from that company afterwards. Thus the problem for the entrepreneurs is to identify interested customer for their products. The products can be in these cases a very general item (a new job, software, merchandise). There are some ways to approach this problem. One way is to search on the Internet to find user's presentation pages and analyze those pages from the semantic point of view to identify if the user that could be interested in that product. In case that the user could be interested he can be informed by sending

the commercial for that product. However this approach is limited because in this moment there are not so many people that have complete personal pages on the net, and usually these pages are not updated. Another approach is trying to identify users based on what they do on the web (inspecting queries, cookies, and server log files). Thus analyzing queries can give us information about categories of interest, at the moment, for the user. Analyzing server log files can give us the information about the IP of the users, pages visited on the server, and interests in those pages (by analyzing the time spent on the pages). This information can be mined to identify users that are now interested in the product.

3 Resource discovery systems

Even though interface and web content designers [AndBuz04] included user behavior and some skills in some search engines users may still have difficulties performing web searches:

- The user is unable to formulate the right query and restrict the result set. Using phrases with many words often produces no result. Users then prefer to specify only one or two words, which generates large sets of results.
- The user interface can be difficult to use or inaccessible for the unskilled or disabled user.
- The ranking function is applied statically; the user is not able to select the criteria most appropriated to him. Some options are present in advances searches, but are rarely applied by users.
- The information on the Internet is rarely structured and organized for fast retrieval by search engines. Web "designers" don't apply meta-tags such as description of keywords correctly and don't use meaningful filenames, titles, link descriptions and alternative texts. In addition, an inappropriate use of metadata produces the phenomena called "search engine spam", aimed at deviating search engine results. For this reason most search engine ignore or only partially use metadata.

Besides the results of search engines, another important aspect is the usefulness of search tools. This aspect is often neglected. It is important to make this very accessible and usable by anyone, regardless of their physical condition or environments. Accessibility guarantees use to all, understandable and navigable content. Usability renders a more efficient and satisfactory Internet navigation. Many factories compose the user interface, like presented in [AndBuz04]:

- **Arrangement of components**. This point is very relevant because value-enhancing features are more "visible" when positioned in an area rapidly encountered by eye movement and do not require page scrolling. For example, the refinement functions of Google [Goo], which allows searching into results, it is not very obvious thanks to its position and font (size and color): it is found at the end of the results page, so inexperienced users may not benefit from them.
- **Number of elements**. Simplicity helps unskilled users navigate the interface easily. Web directories are organized according to categories of goods and services offered. Depending on the type of search it may be more appropriate to use a search engine or directory. On the other hand, their interface is quite full and can create confusion in an unskilled user who wishes to formulate a search query.
- **Expressive power**. A graphical representation can communicate certain kinds of information much more rapidly and effectively than other methods.
- **Functions**. A user typically performs a simple search and specifies one or more words, obtaining a large set of results. Further criteria selection can be specified to restrict search in the results. Preference and commands, although very powerful, are rarely used, even by skilled persons.
- **Clustering of results** allows users to explore results grouped by categories, in this way users can navigate a single branch of results more efficiently.

In the system presented earlier the search engine indexes web pages into a very large database, and returns as response all matching links found in the database for a user query. This is a viable solution and it is used by most search engines, but it is not such a good solution because it needs, from the user's point of view, more time and more searching effort. This is due to the fact that the results to the query are hundreds or thousands of pages and it becomes very difficult for the user to

search and find relevant information in the enormous amounts of information available. Another disadvantage for this type of search is that the results are ordered using page ranking that is applied statically, without any influence from the current user, and these criteria usually may not reflect the needs of the user. This situation occurs when a new, interesting and relevant document for the current user is not placed in among the most interesting documents because it has a poor page rating. This situation makes it too difficult for the user to find it, or too time consuming for the user to find it. Another disadvantage is due to the fact that these types of search don't provide any information about the influence the keyword has in the query results and how does this keyword influence the joining of documents.

All search engines have some capability to customize search, something like "Advanced search", "Preferences". For example the possibility to select the language for the searched pages, to specify exactly the words that need to be in the documents and the words that should not be in the documents, documents format and the date when it was created. The disadvantage is that it has a rigid structure and has a fixed number of possibilities.

To solve the problems presented earlier this field of research has grown rapidly in areas such as algorithms, strategies and architecture. Hundreds of ideas were tried, some were implemented and some are only prototypes. Some of these ideas are:

1. Organizing documents in predetermined categories. Usually these are named ***web directories*** and have a static structure that is very difficult to update. (e.g. yahoo [Yah], aol [Aol], and excite [Exc])
2. An upgrade of the existing search engines was tried by improving the way the results are presented. This was done by different methods of organizing the results:
   - *hierarchical representation* obtained by performing one or two levels on-fly clustering of the results returned by a classical search engine (e.g. webcrawler [WebCr] for one-level representation and vivisimo [VivS] for two-level representation)
   - *graphical representation* by a sequence of interactive maps of the results obtained from the classical search engine (e.g. kartoo [Kart], and surfwax [SuW])
3. Programs (agents) that allow ***monitoring a specified page*** from web and report when changes occur (e.g. DICA, Syskill & Weber and others).
4. Programs (agents) that silently ***observe the user's browser behavior*** during the day and then use these training examples to learn user's profiles, and during the night searches for new pages that are likely to fit the learnt profile.
5. Search engines that help users by suggesting synonyms or idioms of the words presented in the query and try to develop a more refined query to obtain better results. Literature calls this "***Query refinement***".
6. Creating programs (agents) that work between the results of the search engine and the user by filtering the results presented by the former using the ***user profile*** learnt from the latter.

3.1 Web directories

We can easily see that hyperlinks don't create a topic-oriented structure on the web. To solve this problem and group the documents in some way there were created topic-structured directories which are called topic directories. Some of these are *Open Directory Project* [dmoz] and *Yahoo* [Yah]. This structure has been constructed through human effort, and resulted in a giant taxonomy of topic directories. Each directory contains a collection of hyperlinks relevant to the topic that are often the most popular or authoritative sites related to this specific topic. One may model such tree-structured hierarchies which are considering the relations between topics and their generality

such as specific topics or general topic. Following this idea a page is assigned to the best fitting topic for the content of that page. Although topic taxonomies are a special case of semi-structured data, they are important and frequent enough to be mentioned separately. For example, the *dmoz* [dmoz] directory that categorizes Web sites is created manually by about 52 thousand editors, according to Kummamuru [KumLot04], and unfortunately covers less than 5% of the Web.

These directories are created by users and have a fixed structure and the new documents are fitted in this structure. This is a good idea because the structure of these directories is very eloquent and intuitive for human users but it is very difficult to implement. Another disadvantage is that these types contain a limited number of pages in directories because the process of assigning new pages is difficult. This type of server accepts requests from users to put new pages in its structure. These pages can be put only into the existing hierarchical structure. This is based on the users registering a page address and a short description of this page and the server catalogues and puts this page somewhere in its hierarchical structure. Users can search in those directories or can open specific categories from this structure and they can see what is in this structure.

The search engines use agents to find new web sites or web sites that were modified. Web directories will list a site only if this site was registered by the webmaster. Web directories are generally manually created by analyzing the received addresses of sites and than evaluates and catalogues these sites in their structure. Because the descriptions have a limited dimension, in almost all cases these contain only a short description of the content.

In comparison with agents of the search engines that analyze the title, META tags, head and content of the site, web directories evaluate only the title, short description and categories proposed for a site and don't take into account the content of the site and its META tags.

Web directories are divided in categories and subcategories, ordered alphabetically for an easy scan. However the update of new sites is too difficult and needs much time for evaluating the structure and categorize it.

Many sites are based on web directories. Some special sites, very used, are Yahoo (but this also has a search engine) and "search.aol" [Aol] that has many categories and subcategories and many pages indexed in those category. Another interesting site is surfmind [surfM], even if it has only a small number of registered documents (2610 documents) it has a different type of representation of the search results. Thus it can have a list with new sites and a list with most popular sites, or a list with all categories. For a specified category you can see all the articles and a short description for each document. You can also open the article directly in its own web page with differed dimensions of representation without being necessary to open a new web page.

3.2 Representing of search results

On important problem for almost all search engines is the presentation of the results. Usually when search engines return results they return more pages that contain links to the results. Some of these links are interesting for the user and some are not. It is very difficult for the user to rapidly distinguish between the useful and the irrelevant pages. Another disadvantage is that the search engine returns a ranked list of web pages as response to the user's search request. These web pages with different topics and different aspects are mixed together in the returned list. The user has to shift through a long list to locate pages of interest. This method is highly inefficient since the number of retrieved search results can be in the thousands for typical queries. Most users just view

the top results and therefore might miss relevant information. Moreover, the criteria used for ranking may not reflect the needs of the user. It is not specified in the returned list why this document is relevant to the user's query and what is the relationship between the returned documents. Because web directory creation, as presented earlier, is very difficult and very time consuming from a human point of view, it is better to create and populate categories dynamically. After receiving results from the search engines, another program (agent) tries to analyze all the results and tries to classify them in dynamically generated categories. Then it presents this result to the user. Organizing web search results into a hierarchy of topics and subtopics facilitates browsing the collection and locating more easily the interesting results. Almost all sites from this category do this automatically or "on-the-fly". These systems contain two main components: a text classifier that categorizes web pages and a user interface that presents the web pages within the category structure and allows the user to manipulate the structure view. These systems differ in the algorithms used for text classification and the types of user interfaces.

As far as the results are represented there are two main methods: hierarchical representation and graphical representation.

3.2.1 Hierarchical representation of search result

There are systems that create a hierarchical structure for classifying a large, heterogeneous collection of web content. Organizing search results in this format allows users to have a general image of the document categories and focuses on items in categories of interests rather than having to browse through all the results sequentially. This is one of the most used methods for representing the results of the search engine. In this category there are some sites already implemented (e.g. Vivisimo [VivS], WebCrawler [WebCr]) and some are still in research because of the problem of text categorization is very difficult and very time consuming.

S. Dumais et al in [DumChe00] present an algorithm for automatic representation of the search engine results in two-level categories. This method wants to supplement human effort in creating structured knowledge for web directories. The basic idea in their work is to use classification techniques to automatically organize search results into existing hierarchical structures. The model for classification was learned offline using a training set of human-labeled documents and web categories. For classification the authors work with just short summary descriptions of the web pages. This short description is generated automatically for each page and is used for evaluating the classification algorithm. The summaries consist of title, keyword and the description tag if it exists or the first 40 words from the body otherwise. With this technique the authors create a binary vector for each page that indicates whether or not a term appears in the page. For the training and tasting part of text classification the authors use Support Vector Machine [SchSmo02] algorithm because it is very fast and effective for text classification problems. The authors used pre-classified web pages, taken from "LookSmart"[look] web directory, for training. Because SVM algorithm solves the problem for two class classification, for multi-class classification the authors implement one versus the rest algorithm. For each test example, the authors compute the probability of being in each of the 13 first level categories and then in each of the 150 second level categories. To solve this problem the authors use a hierarchical decomposition of a classification problem that allows an efficient learning and representation data. For the training part the authors used a set of pre-classified web pages that can be classified in one ore more classes. Once the categories are learned, the results from any user query can be classified. At query time, each page summary returned by the search engine is compared to the 13 first level category models. A page is placed into one or more categories, if it exceeds a pre-determined threshold for category

membership. Pages are classified into second-level categories only for the first level categories that they belong to. This is done using the same procedure.
In another article Hao Chen et. al. [CheDum00] present the application implemented using the algorithm presented earlier and studies the influence of representing the results of the search engine hierarchically versus typically ranked list interface. This study is done using predefined categories from LookSmart web directory. In the training part the authors make an offline text classification with representative labeled samples of Web pages. At query time, new search results are quickly classified on the fly into the learned category structure.

The categories can be ordered either in a static alphabetical order, or dynamically according to some importance score. The advantage of dynamic ranking is to present the most likely category first. The disadvantage is that it prevents the user from establishing a mental model of the relative position of each category in the browser window. In the dynamical experiment the importance was determined by the number of pages in the category. The category with the most items in it was shown first, and so on. Into the category the pages are sorted by the probability that this page is in this category. The categories used in their experiment were designed to cover the full range of web content. Nonetheless, not all user queries will match the category structure to the same extent, some query may fall entirely. In this case the category interface is like in the list interface and has a "NotCategorized" group. The study showed that the category interface is superior both objectively and subjectively. Users liked the category interface much better than the list interface and they were 50% faster at finding information that was organized into categories. Organizing search results allows users to focus on items in categories of interest rather than having to browse through all the results sequentially.

Kummamuru et. al. [KumLot04] present an idea to build a hierarchical topic for a collection of search results retrieved as response to a query. At each level of the hierarchy the algorithm, named "DisCover", progressively identified topics in a way that maximized the coverage while maintaining distinctiveness of the topics. For evaluating the quality of the topic the authors used some objective measures like "coverage" and "reach time". The authors used this like a method for comparing this algorithm with another two monothetic algorithms for clustering. Automatic taxonomy generation (ATG) algorithm based on how the documents are assigned to different cluster can be categorized into two classes:

- *Monothetic algorithms* in which a document is assigned to a cluster based on a single feature
- *Polythetic algorithms* in which a document is assigned to the cluster based on multiple features

The authors used monothetic algorithms because each cluster is described by a single feature or concept and all the documents presented in a cluster contain this feature. This leads to easier understanding of the cluster by users. The algorithm progressively identifies clusters and it tries to maximize the distinctiveness of the monothetic features describing the cluster while at the same time it maximizes the number of documents that can be described or covered by the monothetic features.

In this article the authors proposed some properties of taxonomies generated from a corpus of documents:

- *Document coverage* – ideally all the documents in the collection should be covered by the taxonomy. A document is said to be covered by taxonomy if it lies in at last one of the clusters in the top level of the taxonomy. This implies that the top level clusters should be chosen in such a way that they cover most of the documents in the corpus. It

will be considered a good taxonomy generation algorithm the algorithm that covers the most number of documents.

- *Compactness* – because the main purpose of the taxonomy is to summarize and provide a better browser experience, the taxonomy needs to be as compact as possible. The taxonomy can be too wide or too deep because the basic purpose may not be served.
- *Sibling cluster distinctiveness* – each of the nodes, especially those at the top level, represent a concept presented in the search results. At any level of the hierarchy this should be as different as possible from each other to maximize generality while browsing the documents in the hierarchy.
- *Node Label Productiveness* – a good taxonomy should help the user to find documents of interest with minimum effort. The labels of the node guide the user in locating a document in the taxonomy. Thus the node labels should be chosen such that they are good indicators of the documents they contain.
- *Reach time* – is the average time needed to locate interesting search results. It is an important criterion for taxonomies that need to be quickly able to locate search results of interest within the hierarchy.
- *General to specific* - the node labels in the hierarchy are actually the concepts associated with the query. In the hierarchy, the most general concept should be associated with the root, and the node label of any node should be more specific (less general) than that of its parent and at the same time it should be less specific than those of its children. The generality or specificity of the node labels needs to be considered within the context of the query.

In the presented article the authors take into consideration only the first three proprieties. It is assumed that each document in the collection can be represented by a set of words. Each node in the hierarchy is associated with a concept and all documents under a node contain that concept. Of course each of these documents will contain several other concepts as well. Monothetic clustering algorithm involves selecting a subset of concepts from all the concepts, optimal in some sense, and associating a child node with each of them. Thus the authors create a compact hierarchy that initially has a limited number of child nodes, but provides to the user the ability to progressively increase the number of child nodes of any node of interest.

For comparing and evaluating this hierarchy the authors performed a users study. This can demonstrate whether the hierarchies generated are actually helpful to real users for browsing. The authors performed the study using 17 volunteers, who were both technical and non-technical employees of the IBM Indian Research Laboratory, and 50 queries (25 ambiguous queries and 25 popular queries). Each volunteer needed to evaluate 3 queries except for one volunteer who evaluated 2 queries. The voluntaries needed to give 5 responses to each query, with a total of 25 responses for ambiguous queries and 25 responses for popular queries. Users were provided with a very short introduction to ATG and the motivation behind creating taxonomy from a document collection. The users were not specified which hierarchy corresponded to which algorithm. Each user needed to fill up an online questionnaire for each query. The questionnaire contained 6 questions, for each the user needed to evaluate each hierarchy on a scale of 1 to 10. These 6 questions were broadly divided into 3 groups. The first groups of 3 questions pertain to the top level nodes of the hierarchy. The second group of 2 questions pertains to the second level nodes, and in the final question the user needed to give an overall rating to the hierarchy. For comparisons, the authors used the proposed algorithm “DisCover” and two others algorithms for hierarchical representations CAARD (Clustering Algorithm for Asymmetrically Related Data [KumKri01]) and DSP (Dominating Set Problem [LawCro01]). For evaluation the authors used only 4 properties: document coverage, compactness, reach time and computational complexity. The results obtained show that “DisCover” is superior to DSP in almost every aspect and in

comparison with CAARD is better in terms of summarizations for both popular and ambiguous queries.

3.2.2 Interactive map for representing the search result

A new approach in the representation of the search results is graphical user interface that allows easier navigation and exploring of these results. Kunz et al present in [ChrBot02] an interactive matrix for showing the hyperlinks of retrieved sites. In this representation the user can see the results using one category resulting in a List Browser and two categories at the same time with the so called Matrix Browser. To represent search results the authors use a familiar and well known interactive tree widget. The tree widget provides the ability to display multiple items in a tabular format. The tree widget is used to display hierarchically organized data. It is a vertical container for widgets of tree type items. The tree widget provides geometry management of arbitrary widgets arranged in a directed, acyclic graph.

In general visualization and exploring search results structure still constitutes a major problem for user interface design, in terms of minimizing search results viewing and supporting user's understanding and providing efficient interaction for exploring the search results. In the presented paper the authors combine the two ways of searching and exploring in the information space in a new graphical user interface for the search engine. Thus the results of the search engine are displayed in one interactive category tree or in an adjacency like system with two interactive trees as axis of a matrix. The first overview generated for the graphical user interface shows how many hits are found in the hierarchically ordered categories. In the List Browser representation the results set is listed in a window, like a tree widget which allows the user to expand and collapse the categories and subcategories by clicking on the interactive symbols in front of the bar and the category name. The size of the bar represents the consolidated amount of search hits in the category and all subcategories. So the user gets an overview in which categories the search results are located, and thus can explore the results in a flexible manner. In the Matrix Browser the users have the possibility to navigate within these trees and explore the structured search results. They have different kinds of interactive symbols inside the matrix visualization. Circles represent how many sites are found in two categories and squares represent the site references itself. In the hierarchical categories system the users can increase or reduce the displayed amount of information and refine their query without input of any text data and they have the opportunity to view more details of the metadata structure together with the local sites.

Another type of representation of the search results is a graphical representation where all information is placed intro tree structure with connections between documents. For example the site www.kartoo.com is very interesting because of the way it represents the search results, and because all other things have total graphic representation. As many search engines it uses other search engines from the web. It only graphically represents all the results. It allows the user to choose what search engine he wishes to be used for the search. The disadvantage is that they provide only 30 results at the same time (in the graphical representation). The interesting part is that the authors represent the names of the documents and their connections together with the keywords that connect them. It has a separate section for monitoring specified sites (this part is explained in the next section). This site is interesting as a representation mode of the search results but information was not available regarding the techniques used for implementation.

In [WizWal04] Wiza et. al. preset an innovating system for building the visualization of the search results in a 3D representation. The authors use for representation the X-VRML system (X - Virtual Realities Modeling Language). The authors use an interface selection which receives the search

results from the most popular search engine and selects the best representation for those results. This idea is interesting as it can model the results in the most appropriate manner. This idea has not been found in the works presented above. The system, called Periscope [Per], can choose to represent the data into a 2D or 3D space; because some 2D maps representation can provide reasonable good results. The 2D representation is presented with a flat panel, limited in size, where information can be presented in patch forms. Sometimes it uses different textures and it is used when the number of object presented is low. For creating the 3D representation the authors take in consideration 3 elements:

- *user interaction* - navigation in the space and interaction with its contents;
- *user cognition* - the spatial representation data is closer manner to the humans perceive the surrounding world and permit the user to change the viewpoint to improve perception and understanding of observed data;
- *information capacity* - the system can provide information in different shapes, colors, textures, positions, sizes, orientations and even behavior;

In comparison with 2D representation the 3D representation is not limited in size and space but only in user perception.

3.3 Monitor specified pages

There are situations when users find the right place where the interesting information is placed but the information is not available for the moment and they need to revisit the site periodically to find new information. According to this in [Ack97] and [AckSta97] is presented an agent that uses machine learning for detecting "interesting" changes in Web pages previously marked by the user to be relevant. Because this agent is based on the changes on known pages rather than finding new pages, it increases the likelihood that the found information will be interesting. The authors called this agent DICA (Do I Care Agent). This agent has some major functions:

- Periodically it visits a user defined list of target pages
- Identifies any changes since the last time it was visited
- Analyzes pages and decides if the changes are interesting
- Notifies the user if the changes are interesting
- Accepts relevance feedback on the interestingness of the change
- Facilitates information sharing and collaboration between individuals and groups.

Because the user provides feedback on whether change reasons are interesting or not and there is a limited list of web pages the precision is greatly improved. For this the author use two features for improving the precision: percentage of retrieved items that are "interesting" and the percentage of potentially interesting items that are actually retrieved. The analysis of changes in the monitored pages is based on the fact that changes are usually incremental and new things are usually added in ways that are consistent with what has been done in the past. In order to detect whether a change is interesting or not the authors use Bayesian classification process and extract a set of compatible attributes and compute the probability that the object belongs to each of the known set of categories. The agent notifies the user when the computed probability exceeds a configurable threshold. For classification the authors create a classification profile that is a list of feature values and the corresponding probability that documents having those value belong to one or more classes.

All interesting changes are sent to the user by e-mail and all changes are also listed in the agent's associated web pages which users can browse at their leisure. Such as in these pages attribute values that define an interesting change are listed. These pages are also used for relevance

feedback. In these pages the changes found by the agent in the monitored pages are sorted according to how interesting they are likely to be, with more interesting changes on top. All changes are initially marked "Unrated" and the user can change this to "I Care" or "I don't Care" and after that the agent learns again.

Because different pages may be interesting for different reasons, users may maintain multiple DICA agents specialized on particular topics or kinds of Web pages. For example, the criteria for an interesting journal announcement may be different from a news item. An agent can receive changes from another agent and can decide if those changes are interesting or no from its point of view. For more details on Agent's Paradigm and detailed information on Agents see [Bar02].

Other two agents that learn user profiles are presented in [AckBil97]. First "Syskill & Webert" is an agent that is designed to help users with long term information seeking goals, such as finding previously unvisited web sites on a particular technical topic. The authors use a simple Bayesian classifier to create a user profile from the user's preferences. The profile is used afterwards to compute the probability that a web page is interesting to the user or not. For learning the user profile the authors use feedback on the interestingness and usefulness of the web pages. The authors developed an interface where they present the results and the user can click on "Hot" or "Cold" buttons to specify if he is or isn't interested in this document. When creating a query for the search engine the agent uses two types of words. The first query contains the words that occur in the most number of pages that have been rated "hot" and the second contains words whose presence helps discriminate pages that are rather hot then cold using mutual information. Because the authors use a Lycos search engine that doesn't accept a very long query they use the seven most informative words that are found in a higher proportion of hot pages and the seven most commonly words occurring as cold. The agent retrieves each returned page and analyzes the HTML files to produce its recommendations. The agent displays its recommendations by annotating links that indicate whether the user is likely to be interested in the page and a number indicating the probability that a page is interesting.

The second agent presented by Ackerman [Ack97,AckBil97] is named "GrantLearner" that is an agent that notifies an individual of new research grant opportunities that meet the learned profile of the individual's research interests. This agent learns to distinguish between interesting and uninteresting funding opportunities based a user's rating of these descriptions. This agent is connected to a grant database maintained at UCI (University of California, Irvine) and provides a user interface to browse it. The system displays a list with all grant subjects and a description of the grant. Using "hot" and "cold" buttons the user can indicate whether a grant is related to his interests. After that when the user provides information about 10 interesting grants the agent creates the profile of the user interests and then processes all database using this profile providing a new list sorted by relevance for the user. For this the agent uses the same Bayesian classification method as for learning the user probabilistic profile.

3.4 User's browser behavior

In the last years a common idea used for increasing the quality of the search results is creating a user's profile based on user's interests. Then, this profile is used to improve search results or to develop the query to obtain better results. In this sense Goecks and Shavilk in [GoeSha99] present an agent that learns a user browser behavior and tries to establish a user profile using this information. Almost all of this type of research was focused on learning a user profile based on observing the hyperlinks clicked or the amount of scrolling performed. The authors present here an

agent that combines more techniques. Thus when the user navigates on a new page, the agent records (a) the text of the HTML file; (b) the number of hyperlinks the user clicked; (c) the amount of user scrolling activity; and (d) the amount of user mouse activity. Of these, (a) creates the input and (b)-(d) constitute the outputs of its training examples. When the agent observes that the user performs a large number of actions on the page, it can label that page as a positive instance of the user's interests. The agent *sums* the actions of the user on each page visited over a finite period of time. It represents the information recorded (HTML text) like a vector of word frequencies and uses a neural network (back-propagation algorithm) to learn the user's profile. The authors use the HTML markup tags to distinguish the context of the word and for different places of the word it gives different weights for that word. The experiment shows that the error for predicting mouse activity was much larger than the error of the other output units. These results could indicate that the user's mouse activity on a page does not correlate with the user's interest in a page.

This idea can be used to define an agent that silently observe the user's browsing behavior and then uses these training examples to learn different functions and gather pages that are likely to be of interest to the user.

3.5 User refined search

Single keywords are usually ambiguous, or to general for the search engine. Moreover, they can occur in vast quantities of documents, thus making the search return hundreds of results, most of them being irrelevant. Giving additional keywords can refine the search and provide considerable improvement in the retrieved results. Good refinement words must have meaning that helps disambiguate or make more specific the original search word. Providing the refinement words would help a search engine to prune out documents where the word is used with any of its other meanings. There are three ways to expand the query: manual query expansion, semi-manual query expansion, and automatic query expansion. In manual query expansion, the user knows the intended meaning of the keywords he used. In semi-manual query expansion the system provides some synonyms for the keywords and the user selects relevant synonyms. In the automatic query expansion agents are used to find and use best extra keywords using the user's profile to obtain better results.

Intelligent software agents are being developed to deal with these issues. One of these agents called "WebMate" is presented in [CheSye98] and can be found at [WebM]. It is a personal agent for browsing and searching the web. It does roughly two things: (1) learning user interests incrementally, with continuous update and automatically providing documents that match the user interests (e.g. a personalized newspaper), and (2) helping the user to refine search so as to increase retrieval of relevant documents. The agent is composed of a stand-alone proxy that monitors user's actions to provide information for learning and refinement of the search, and an interface used for interaction with the user. In the filtering task the agent judges whether an article is relevant or irrelevant to the user, based on the user's profile, in a probabilistic environment. In contrast with other systems that learn a user's profile and after that use it statically to determine relevant documents, WebMate learns the user profile incrementally and continuously. The system extracts and combines relevant keywords from the relevant pages provided by the user and uses them for keyword refinement. In WebMate agent, the context of the searched keywords in the "relevant" pages is used to refine the search because they think that the context of the search keywords is more informative than the content of the pages. They do this by considering what pages were marked by the user as relevant.

3.6 User profile

There are at least two methods of specifying the user profile. In the first method the profile is specified explicitly as the users provides information about him (her) self. Usually users don't update and don't modify over time their profile, and the profile is almost always out of date. In the second method the profile is specified implicitly as the profile is created based on the users' actions observation.

Albanese et al in [AlbPic04] presented an algorithm for customizing the content and the structure of web sites in order to provide users with the information they are interested in. As far as personalizing the content of web site is concerned, the novelty of the structure is that they use a two phase classification approach rather than a single phase approach. The authors take into account both users provided data and browsing patterns and they classify both users and contents.

The algorithm has two phases. In the first phase a pattern analysis and classification is performed by means of an unsupervised clustering algorithm, using the registration information provided by the users. In the second phases a reclassification is iteratively repeated until a suitable convergence is reached. Also the authors use reclassification to overcome the inaccuracy of the registration information and they use observations based on users' navigation behavior. The authors use a clustering procedure for partitioning the feature space built upon the user provided data into a certain number of clusters that group together users appearing to be similar. The reclassification phase is based on the interaction (query, navigation, searching among directories) of each user with the web site.

Barbu and Marin in [BarMar03] present a new algorithm for learning the user's profile. Because search engines return many pages for a given query there are various methods to prune irrelevant documents. Some methods proposed in recent years use the user's profile for filtering the information from search engines. This user's profile can be used to filter the documents returned from search engines or can be used to reformulate the query based on the interests of the user. The authors try to keep track of the user's interests by building an individual user's profile and modify this profile over time. The authors want to identify what part of the user's profile is relevant with the current search. Thus the authors proposed a scheme that learns the user's profile continuously and incrementally using the initial user's profile, actions of the user and a semantic interpretation of the query. The authors take into account the user's current interests and their decay in time if the interests change. From this result a model called *time-word vector hyperspace* that computes the dynamics of the user's profile. Thus is created a vector that has word component computed using TD-IDF (Terms Documents – Inverse Document Frequency) technique and the temporal dimension set to zero. Queries are represented as features vectors but in addition to the TF-IDF weights and the temporal dimension set to an initial value that decays in time. This allows that some specific user interests could decrease with time or if the user is interested in that category it can be maintained / increased. For the semantic analysis of the query the authors used WordNet. WordNet [WNet] is an online lexical system developed by the Cognitive Science Laboratory at Princeton University and it is inspired by current psycholinguistic theories of human lexical memory. Each is organized into synonym sets, each representing one underlying lexical concept. The WordNet provides various meanings for a given word.

The user can either input an explicit query or he can narrow his search process when he clicks on the links of the displayed web pages. Contextual relevant information improves the search performance by filtering the retrieved documents by a relevance score computed as the cosine similarity between the User's Recent Profile feature vector and the feature vectors of the documents retrieved by a classical search engine. The preliminary result shows that the quality of

the search result is improved. Thus the authors show that for a classical search engine, from the results of a query only 6 documents from the first 10 are relevant with what the user wants. When they use a standard profile filtering this number is increased to 8 out of 10 and when they use a dynamic profile filtering all presented articles (10 out of 10) are relevant with what the user wants.

Another interesting algorithm to improve the user's profile used for filtering the result of the classical search engines is presented by Tresp and Kai in [KaiSch02] where the authors combine collaborative filtering and content based filtering into probabilistic framework called by them *Collaborative Ensemble Learning*. Content Based Filter (CBF) represents a system that analyzes the content of a set of items together with a rating provided by the individual user from which it deduces unseen items that may be interesting for the current user. In contrast Collaborative Filter (CF) creates a database of ratings of items taken from a large set of users. The prediction for the current user is based on the rating provided by all other users. The difficulty problem for CBF systems is the gap between low level content features and high level user interface (human perception can't be formalized just based on the content analysis). Human understanding can be formalized when discussing text analysis but formalizing human perception of an image or a sound would only lead to irrelevant details regarding the ensemble. The CF systems memorize the user's preferences without incorporating the actual context of items. This system can not recommend a new page for which there isn't a previous rating given by a user. The algorithm uses a probabilistic model of Support Vector Machine to represent each user's profiles (like CBF) and in the prediction phase, it is combined with user's profile groups (from CF).

For modeling the profile of the user it is used the Support Vector Machine (SVM) [Nel00]. SVM is a classification technique that was applied with great success in many classification problems. The authors extend the standard SVM, that doesn't have a measure of confidence, with a probabilistic extension suggested by Platt [Pla99] (called PSV) where the authors associate a probability of class membership to each user. In the collaborative ensemble learning the authors combine current user's preferences, described by the user's model using PSVM, and other users' preferences, described by PSVM, and does the prediction for current user.

Unseen items are described by their feature vectors. To estimate these items the authors use the user's rating (if a user likes or dislikes this item). The authors put all items into a probabilistic framework under four assumptions. The first assumption is that every user is described by a profile that is generated from apriori distribution. The second is that the distribution of the actual items x is independent of the user's profiles. Thirdly, the user has a given rating based on his profile, and fourthly, given a user profile the opinions for individual items are mutually independent. The authors interpreted the results of predicting a current user preference as a combination of user's profiles society.

For modeling the user's profile using support vector machines the authors used a linear kernel for the part of text retrieval problem and a radial basis function kernel for the part of art image retrieval problem. The authors assume that each user is interested in exactly one category.

In another article [KaiTre03] Tresp and Kai improve the above algorithm by using a hierarchical Bayes framework for information filtering. The authors model the user's profile using content based probabilistic filtering that are combined with a user's preference society (like in collaborative filtering). The combination scheme can be interpreted as a hierarchical Bayesian approach in which a common apriori distribution is learned from related experiments. In hierarchical Bayes framework the authors assume that preference model for each user has been generated as a simple form of an apriori distribution.

In the experimental part collaborative ensemble learning is compared with two methods of information filtering, (1) one pure collaborative filtering using Pearson correlation, and (2) a pure content based filtering using SVM with a linear kernel. The results were reported based on two data sets, the Reuters text data set (that has 36 categories covering 10034 articles) and a data base of user opinions on art images (642 images).

For text data experiment the authors assumed that each user is interested in exactly one category, and presented results for 360 users by choosing a random set of examples. The authors presented results for two scenarios: first with only 5 examples for each user (insufficient information about users) and second with 30 examples for each user. Collaborative filtering performs worst in both cases, content based filtering using SVM provides reasonable good results and collaborative ensemble learning outperforms both other methods.

For the data base of user's opinions on art images the authors collected user's preferences for art images in a web-based survey, choosing a total of 642 images and asked for their opinions (like/ dislike/ not sure). The authors collected data from 190 users, at average each of them had rated 89 images. To describe the images content they used 256 features, 10 features based on wavelet texture and 9 features on color moment, obtaining a 275 dimensional feature vector for each image. The authors evaluated the performance for 2, 5, 10, 20, 50 and 100 rated images, chosen at random from the collected data for this particular user. Content based filtering gave a very poor performance, since the low level image features are not indicative of the image content. Collaborative filtering performs well, once the number of rated images for the text user is suitably large. Collaborative ensemble learning achieved excellent performance when very few examples are given for a text user.

4 Conclusions and further work

At the beginning of the year 2000 the web had over 800 million pages covering most areas of human endeavor, and had six terabytes of data stored in almost three million servers. Nearly a million pages are added daily, and a typical page is changed every few months, and each month several hundred gigabytes are changed. These characteristics grow continually. Thus the World Wide Web has become an opportunity and a great challenge for researchers in computer sciences.

Some so-called organizers suggest invariably that the users need to organize the information during the creation time, using some preordered rules. This is pointless because most people will not respect it. This is considered a characteristic for unordered egalitarianism of Internet [Der00]. Any attempt to apply the same organizing rules will determine the users to leave. The result for the time being is that most information needs to be organized after it was generated, and searching tools and organizing tools need to work together tightly.

In this technical report I am tried to present a short perspective of the research in this fertile domain and the tendencies in research. Because the web grows continually the volume of unstructured data (text and hypertext) exceeds that of structured data, and the research of machine learning techniques has changed its trend. Thus in the last years research in this field was focused on four areas such as strategies, algorithms, architecture and user interface to find a way to search more easily and in a more personal way for the user.

For a user that wants to find something on the web, searching can become a frustrating activity when search engines return thousands of documents for a given search. Thus many researches are focused on organizing the search results, finding new strategies in implementing search engines and representing architectures of the search results, without taking into account the user. In the last years the web has more types of inexperienced users from different domains of activities and the new challenge is to make the search more simple and easy to understand for the user. Due to this the research has split into three directions: (1) finding relevant information, (2) representing the results and (3) guiding the user in finding information.

In *finding relevant information* the research focuses on trying to combine the existing search engines with information about the user. Most search engines nowadays find relevant information using static methods (like Page Ranks, et.al.) that take in consideration only information about structure and content of the web, without taking into consideration information about the user that it is interested in that information. The challenge of this field is to build the user profile. There are presented methods that build a profile of the user using information taken from the user and some documents that are interesting to the user and then trying to use this user profile in finding relevant information. The problem is that users change their interests fairly frequent and almost in all cases they can not specify what their interests in a very specific way are. A solution to this problem is using an actual user profile for filtering new documents and based on new filtered documents taken into consideration updating the user's profile.

Studies regarding *representing the results* demonstrated that an interface based on categories is superior to the list interface both in a subjective and objective way. There are many directions for future research. One issue is to explore how the results generalize into domains. Another issue is a better representation of search results in concise views. In this field there have been constructed more types of representation of the search results: hierarchical and graphical representations. From this point of view the hierarchical representation is intuitively, easier to use and understand by the

user but the main problem is that of choosing the categories. Thus a fixed category structure has been created and it is used to group the search results. The problem arises due to their fixed structure. Ideas have been proposed that represent the search results in dynamical hierarchical structure but there is a problem regarding the characteristics of classified documents that are relevant to the user. In this type of representation usually only a small portion of the most important and representative information is displayed in the initial screen, and overlay techniques are used to cover more details. New methods of representing the search results have been tried, one of this being 3D perspective representation. Viewing and exploring search results into this perspective still constitutes a major problem for the designers of the user interface in terms of minimizing the visual search, supporting user understanding and providing efficient interaction for exploring the network.

Guiding users in finding information is actually trying to ease their quest for information. In most of the cases the users don't know exactly what is interesting for them. Research is being focused on suggesting synonyms for words that appear in the query or fields in which the words appear so that only interesting results can be retrieved. An interesting idea is trying to take words from the relevant documents in order to create a new and better query. Another field of research is trying to include the user in a group of users considering the momentary interests of the user and to suggest parallel fields with what the user specified in the query and to open new perspectives for the user.

The perspective presented in this PhD report is oriented on trying to dynamically reorganize the view of the web (interface of the web) from the user's point of view, without modifying the basic structure of the Web. Thus the researchers try to personalize the interface of the web for every user, making it easier and more accessible for everyone, and presenting the information in an enjoyable interface. This perspective in a small part from the grater domain of research that is oriented on finding a better representation and organization of information on the web, increasing the communication and collaboration between users and applications. This direction is trying to transform the Web's orientation from documents to relevant data, from the user perspective to the machine perspective. Thus the researchers in this domain try to reorganize data on the Web, give them a meaning, and making possible the automatic processing of data. This new representation is called *Semantic Web* and it wants to assure that it is organizing the data and the information into a natural mode, from the user's point of view, and, at the same time, easy to use (understand) for automated processing.

The definition of *Semantic Web* according to Tim Berners-Lee, the inventor of *World Wide Web* is: "The extension of the current web in which information is given well-defined meaning, better enabling computer and human to work in cooperation". It can be considered that two programs can put together their knowledge by changing the ontology that provides necessary vocabulary for discussion. The Semantic Web is the abstract representation of data on the Web, based on the RDF (Resource Description Framework) standards and other standards to be defined. The data can be defined and linked in such a way that there is more effective discovery, automation, integration, and reuse across different applications. This idea is not quite easy implemented at the moment due to the web's size and because it is difficult to create a general idea for representing the web, considering its content [FenHen03].

For my second PhD report I plan to do text classification using a Support Vector Machine method. Thus, for the first step, using techniques of text mining a set of vectors that represents the signature of each document in the set of documents is being created. For this I use keyword based retrieval method presented in section 2.2.1.2. Based on these vectors I will try to classify a set of documents using a new interesting algorithm based to statistical learning called Support Vector Machine. In its standard formulation SVM doesn't have as an output any measure of confidence for their

prediction, thus we need to associate a probabilistic extension to its outputs. A major problem in text classification is the high dimension of the feature space. Another onset is to try different methods of eliminating non-informative terms (elements in vectors) according to existing statistics, and the construction of new features that combine lower level features with higher level orthogonal dimension features.

References

[Sou00] Soumen Chakrabarti.- Data mining for hypertext: A tutorial survey, Newsletter of the Special Interest Group (SIG) on Knowledge Discovery & Data Mining, ACM Explorations 1(2), pages. 1-11,2000

[JaiMic01] Jiawei Han, Micheline Kamber - Data Mining. Concepts and techniques, Morgan Kaufmann Press, 2001

[Ian00] Ian H. Witten, Eibe Frank – Data Mining, Practical Machine Learning Tools and Techniques with Java implementation, Morgan Kaufmann Press, 2000

[Mit97] Tom Mitchell – Machine Learning, McGraw Hill Publishers, 1997

[Der00] Dertouzos, Michael – What will be: How the New World of Information Will Change Our Lives, Technical Publisher, Bucharest, 2000 (Translation in Romanian by I. Moisil, B. Barbat et. al.)

[cs.utk] www.cs.utk.edu/~dongarra/etemplates/node372.html - representing implementation of sparse matrixes and vectors

[LawGil99] S. Lawrence and C. L. Giles. - Accessibility of information on the web,Nature,400, pages 107-109, 1999

[RayHen00] Raymond Kosala and Hendrik Blockeel – Web mining research: A survey, In SIGKDD Explorations: Newsletter of the Special Interest Group (SIG) on Knowledge Discovery & Data Mining, ACM Press, pages 1-15,2000

[MlaGro99] D. Mladenic and M. Grobelnik – Feature selection for unbalanced class distribution and naïve bayes, In Proceedings of the 16th International Conference on Machine Learning ICML, p.258-267,1999

[YanPed97] Y. Yang, J.O. Pedersan – A Comparative Study on Feature Selection in Text Categorization, Proceedings of ICML, 14th International Conference of Machine Learning, pages 412-420, 1997

[DeeDum90] S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Harshman – Indexing by latent semantic analysis, Journal of the American Society for Information Science, 41, pages. 391-407, 1990

[Hof99] Thomas Hofmann – Probabilistic Latent Semantic Analysis, In Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, Stockholm, pages. 289-296, UAI 1999

[Cha03] Soumen Chakrabarti, Mining the Web. Discovering Knowledge from Hypertext Data, Morgan Kaufmann Publishers, USA, 2003

[CooMob99] R. Cooley, B. Mobasher and J. Srivastava – Data preparation for mining World Wide Web browsing patterns, Journal of Knowledge and Information Systems 1(1), pages 5-32, 1999

[SriCoo00] J. Srivastava, R. Cooley, M. Deshpande, P.Tan – Web Usage Mining: Discovery and Applications of Usage Patterns from web Data, ACM SIGKDD Explorations, pages 12-23, 2000

[AndBuz04] P. Andronico, M. Buzzi, B. Leporini – Can I Find What I'm Looking For?, In Proceedings of the World Wide Web Conference 2004, New York, pp. 430-431, 2004

[Goo] www.google.com (search engine)

[Yah] www.yahoo.com (search engine and web directory)

[Aol] search.aol.com (web directories)

[Exc] www.excite.com (web directories)

[WebCr] www.webcrawler.com (one level hierarchical representation of the search results)

[VivS] www.vivisimo.com (2 levels hierarchical representation of the search results)

[Kart] www.kartoo.com (graphical representation of search results and monitoring a specific site)

[SuW] www.surfwax.com (help user to find different synonyms grouped after domain for every search word)

[dmoz] www.dmoz.org (Open Directory Project - web directories)

[surfM] www.surfmind.com (web directories and searching into a specific domain)

[DumChe00] S. Dumais, Hao Chen – Hierarchical Classification of Web Content, Proceedings of the 23rd international ACM SIGIR Conference on Research and Development in Information Retrieval, pages 256-263, 2000

[SchSmo02] Bernhard Schoslkopf, Alexander Smola – Learning with Kernels, Support Vector Machine, MIT Press, London, 2002

[look] http://www.looksmart.com/ (web directories with preclasified web pages)

[CheDum00] Hao Chen, S. Dumais – Bringing Order to the Web: Automatically Categorizing Search Results, In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pages 145-152, 2000

[KumLot04] Kummamuru K., Lotilikar R., Roy S., Singal K., Krishnapuram R. – A Hierarchical Monothetic Document Clustering Algorithm for Sumarization and Browsing Search Results, In Proceedings of the Thirteenth International World Wide Web Conference, pages 658-665, 2004

[KumKri01] Kummamuru K. and Krishnapuram – A clustering algorithm for asymmetrically related data with its applications to text mining, In Proceedings of CIKM, pages 571-573, 2001

[LawCro01] Lawrie D., Croft W. B. and Rosenberg A. – Finding topic words for hierarchical summarization, In proceedings of SIGIR, pages 349-357, 2001

[ChrBot02] Christoph Kunz, Veit Botsh – Visual Representation and Contextualization of Search Results, List and Matrix Browser – In Proceedings of International Conference on Dublin Core and Metadata for e-Communities, pp. 229-234, 2002

[WizWal04] Wojciech Wiza, Krzysztof Walczak, Wolciech Cellary - Periscope – A System for Adaptive 3D Visualization of Search Results, Association for Computing Machinery , p.29-40, 2004

[Per] http://periscope.kti.ae.poznan.pl/ (graphical representation of the search results)

[Ack97] Ackerman Mark – The DO-I-Care Agent: Effective Social Discovery and Filtering on the Web, Proceedings of RIAO'97: Computer-Assisted Information Searching on the Internet, pages 17-31, 1997

[AckSta97] Ackerman Mark, Brain Starr, Michael Pazzani - DO I Care? – Tell Me What's Change on the Web, In Proceedings of the American Association for Artificial Intelligence Spring Symposium on Machine Learning, 1997

[Bar02] Barbat, Boldur - Agent - oriented intelligent systems, Romania Academy Publisher, Bucharest, 467 pages, 2002 (in Romanian).

[AckBil97] Ackerman M, D. Billsus, S. Gaffney, S. Hettich, G. Khoo, D. Kim, R. Klefstad, C. Lowe, A. Ludeman, J. Muramatsu, K. Omori, M. Pazzani, D. Semler, B. Starr, P. Yap - Learning Probabilistic User Profiles: Applications to Filtering Interesting Web Sites, Notifying Users of Relevant Changes to Web Pages, and Locating Grant Opportunities - AI Magazine 18(2) pages 47-56, 1997

[GoeSha99] Jeremy Goecks, Jude Shavilk – Automatically Labeling Web Pages Based on Normal User Action, Proceedings of the International Joint Conference on Artificial Intelligence. Workshop on Machine Learning for Information Filtering, pages 573-580, 1999

[CheSye98] L. Chen, K. Sycara – WebMate: A Personal Agent for Browsing and Searching, Proceedings of the 2nd International Conference on Autonomous Agents, pages 132-139, 1998

[WebM] www.cs.cmu.edu/softagents/webmate (the proactive agent that learn the user profile to increase the quality of the search results)

[AlbPic04] M. Albanese, A. Picariello, C. Sansone, and L. Sansone – A Web Personalization System based on Web Usage Mining Techniques, In Proceedings of the World Wide Web Conference 2004, New York pp. 288-289, 2004

[BarMar03] Barbu Costin, Simina Marin – Information Filtering Using the Dynamics of the User Profile, In Proceedings of The 16th International FLAIRS Conference, 2003

[WNet] http://www.cogsci.princeton.edu/wn2.0 (online lexical system)

[KaiSch02] Kai Yu, Anton Schwaighofer, Volker Tresp, Wei-Ying Ma, HongJing Zhang - Collaborative Ensemble Learning: Combining Collaborative and Content Based Information Filtering, Technical Report Siemens, 2002

[Nel00] Nello Cristianini, John Swawe-Taylor – An introduction to Support Vector Machines, Cambridge University Press, 2000

[Pla99] Platt John C. – Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods, In Advances in Large Margin Classifiers, A. Smola, P. Bartlett, B. Scholkopf, D. Schuurmans, eds., MIT Press, Cambridge, pages 61-74, 1999

[KaiTre03] Kai Yu, Anton Schwaighofer, Volker Tresp, Wei-Ying Ma, HongJing Zhang - Collaborative Ensemble Learning: Combining Collaborative and Content Based Information Filtering via Hierarchical Bayes, Proceeding of the 19th Conference, Uncertainty in Artificial Intelligence, pages 616-623, 2003

[FenHen03] Fensel D, Hendler J., Liberman H., Wahlster W. – Spinning the Semantic Web – Bringing the World Wide Web to its Full Potential – MIT Press, 2003