DETECTION OF TRENDS OF TECHNICAL PHRASES IN TEXT MINING USING CLUSTERING TECHNIQUES

P-3124

A PROJECT REPORT

Submitted by

GOPIKASRI N

GAYATHRI S

in partial fulfilment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY, COIMBATORE

ANNA UNIVERSITY : CHENNAI 600 025

APRIL 2010

ANNA UNIVERSITY : CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report "**DETECTION OF TRENDS OF TECHNICAL PHRASES IN TEXT MINING USING CLUSTERING TECHNIQUES**" is the bonafide work of "**GOPIKASRI N** and **GAYATHRI S**" who carried out the project under my supervision.

SIGNATURE

Dr. S. Thangasamy

DEAN

Department of Computer Science & Engg
Kumaraguru College of Technology,
Chinnavedampatti Post,
Coimbatore – 641606

SIGNATURE

Mrs. D.Chandrakala

SUPERVISOR

Assistant Professor

Department of Computer Science & Engg
Kumaraguru College of Technology,
Chinnavedampatti Post,
Coimbatore – 641606

The candidates with University Register Nos. **71206104016 and 71206104401** were examined by us in the project viva-voce examination held on

.....16/4/2010.....

ACKNOWLEDGEMENT

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be but incomplete without the mention of the people who made this possible and whose constant guidance and encouragement crowns all efforts with success.

We are extremely grateful to **Dr. S. Ramachandran,** Principal, Kumaraguru College of Technology for having given us this opportunity to embark on this project.

We express our sincere and heartfelt thanks to **Dr. S. Thangasamy,** Dean, Department of Computer Science and Engineering, for his kind guidance and support.

We would like to express our sincere thanks to our project coordinator **Mrs. P. Devaki,** Assistant Professor for her valuable guidance during the course of the project.

We would also like to thank our class advisor **Mrs. R. Kalaiselvi,** Senior Lecturer for her constant support and guidance.

We would like to thank our guide **Mrs. D.Chandrakala,**Assistant Professor without whose motivation and guidance we would not have been able to embark on a project of this magnitude. We express our sincere thanks for her valuable guidance, benevolent attitude and constant encouragement.

We reciprocate the kindness shown to us by the staff members of our college, people at home and our beloved friends who have contributed in the form of ideas, constructive criticism and encouragements for the successful completion of the project.

ABSTRACT

ABSTRACT

In text mining processes, the importance indices of the technical terms play a key role in finding valuable patterns from various documents. Further, methods for finding emergent terms have attracted considerable attention as an important issue called temporal text mining. However, many conventional methods are not robust against changes in technical terms. In order to detect remarkable temporal trends of technical terms in given textual datasets robustly,we propose a method based on temporal changes in several importance indices by assuming the importance indices of the terms to be a dataset. The method consists of an automatic term extraction method in given documents, three importance indices from text mining studies, and temporal trends detection based on results of Self Organizing Maps(som).Empirical studies show that the three importance indices are applied to the titles of four annual conferences about data mining field as sets of documents. After detecting the temporal trends of automatically extracted phrases,we compared the trends of the technical phrases among the titles of the annual conferences.

LIST OF FIGURES

LIST OF FIGURES

LIST OF TABLES

LIST OF TABLES

LIST OF ABBREVIATION

LIST OF ABBREVIATIONS

1.	SOM	Self Organising Maps
2.	CN	Compound Nouns
3.	tf	Term Frequency
4.	idf	Inverse Document frequency
5.	BMU	Best Matching Unit
6.	XML	Extended Markup Language

CONTENTS

TABLES OF CONTENTS

INTRODUCTION

1.INTRODUCTION

1.1 DATA MINING

Data mining is the process of extracting patterns from data. Data mining is becoming an increasingly important tool to transform this data into information. It is commonly used in a wide range of profiling practices, such as marketing, surveillance, fraud detection and scientific discovery .Data mining can be used to uncover patterns in data but is often carried out only on sam

ples of data. The mining process will be ineffective if the samples are not a good representation of the larger body of data. Data mining cannot discover patterns that may be present in the larger body of data if those patterns are not present in the sample being "mined". Inability to find patterns may become a cause for some disputes between customers and service providers. Therefore data mining is not foolproof but may be useful if sufficiently representative data samples are collected. The discovery of a particular pattern in a particular set of data does not necessarily mean that a pattern is found elsewhere in the larger data from which that sample was drawn. An important part of the process is the verification and validation of patterns on other samples of data.

1.2 TEXT MINING

The purpose of Text Mining is to process unstructured (textual) information, extract meaningful numeric indices from the text, and, thus, make the information contained in the text accessible to the various data mining (statistical and machine learning) algorithms.Information can be extracted to derive summaries for the words contained in the documents or to compute summaries for the documents based on the words contained in them. Hence, we can analyze words, clusters of words used in documents,

them or how they are related to other variables of interest in the data mining project.

In the most general terms, text mining will "turn text into numbers" (meaningful indices), which can then be incorporated in other analyses such as predictive data mining projects, the application of unsupervised learning methods (clustering), etc.

1.2.1 Typical Applications for Text Mining

Unstructured text is very common, and in fact may represent the majority of information available to a particular research or data mining project.

Analyzing open-ended survey responses:

In survey research (e.g., marketing), it is not uncommon to include various open-ended questions pertaining to the topic under investigation. The idea is to permit respondents to express their "views" or opinions without constraining them to particular dimensions or a particular response format. This may yield insights into customers' views and opinions that might otherwise not be discovered when relying solely on structured questionnaires designed by "experts." For example, we may discover a certain set of words or terms that are commonly used by respondents to describe the pro's and con's of a product or service (under investigation), suggesting common misconceptions or confusion regarding the items in the study.

Automatic processing of messages, emails:

Another common application for text mining is to aid in the automatic classification of texts. For example, it is possible to "filter" out automatically most undesirable "junk email" based on certain terms or words that are not likely to appear in legitimate messages, but instead identify undesirable electronic mail. In this manner, such messages can automatically be discarded. Such automatic systems for classifying electronic messages can

(automatically) to the most appropriate department or agency; e.g., email messages with complaints or petitions to a municipal authority are automatically routed to the appropriate departments; at the same time, the emails are screened for inappropriate or obscene messages, which are automatically returned to the sender with a request to remove the offending words or content.

Analyzing warranty or insurance claims, diagnostic interviews:
In some business domains, the majority of information is collected in open-ended, textual form. For example, warranty claims or initial medical (patient) interviews can be summarized in brief narratives, or when you take your automobile to a service station for repairs, typically, the attendant will write some notes about the problems that you report and what you believe needs to be fixed. Increasingly, those notes are collected electronically, so those types of narratives are readily available for input into text mining algorithms. This information can then be usefully exploited to, for example, identify common clusters of problems and complaints on certain automobiles, etc. Likewise, in the medical field, open-ended descriptions by patients of their own symptoms might yield useful clues for the actual medical diagnosis.

1.2.2 APPROACHES TO TEXT MINING

To reiterate, text mining can be summarized as a process of "numericizing" text. At the simplest level, all words found in the input documents will be indexed and counted in order to compute a table of documents and words, i.e., a matrix of frequencies that enumerates the number of times that each word occurs in each document. This basic process can be further refined to exclude certain common words such as "the" and "a" (stop word lists) and to combine different grammatical forms of the same words such as "traveling," "traveled," "travel," etc. (**stemming**). However, once a table of (unique)

and **data mining** techniques can be applied to derive dimensions or clusters of words or documents, or to identify "important" words or terms that best predict another outcome variable of interest.

Using well-tested methods and understanding the results of text mining:

Once a data matrix has been computed from the input documents and words found in those documents, various well-known analytic techniques can be used for further processing those data including methods for clustering, factoring, or predictive data mining .

"Black-box" approaches to text mining and extraction of concepts:

There are text mining applications which offer "black-box" methods to extract "deep meaning" from documents with little human effort (to first read and understand those documents). These text mining applications rely on proprietary algorithms for presumably extracting "concepts" from text, and may even claim to be able to summarize large numbers of text documents automatically, retaining the core and most important meaning of those documents. While there are numerous algorithmic approaches to extracting "meaning from documents," this type of technology is very much still in its infancy, and the aspiration to provide meaningful automated summaries of large numbers of documents may forever remain elusive. We urge skepticism when using such algorithms because 1) if it is not clear to the user how those algorithms work, it cannot possibly be clear how to interpret the results of those algorithms, and 2) the methods used in those programs are not open to scrutiny, for example by the academic community and peer review and, hence, we simply don't know how well they might perform in different domains.

Text mining as document search:

There is another type of application that is often described and referred to as "text mining" - the automatic search of large numbers of documents based

internet search engines that have been developed over the last decade to provide efficient access to Web pages with certain content. While this is obviously an important type of application with many uses in any organization that needs to search very large document repositories based on varying criteria.

1.3 CLUSTERING TECHNIQUES

Different approaches to clustering data can be described with the help of the hierarchy shown in figure 1.1.

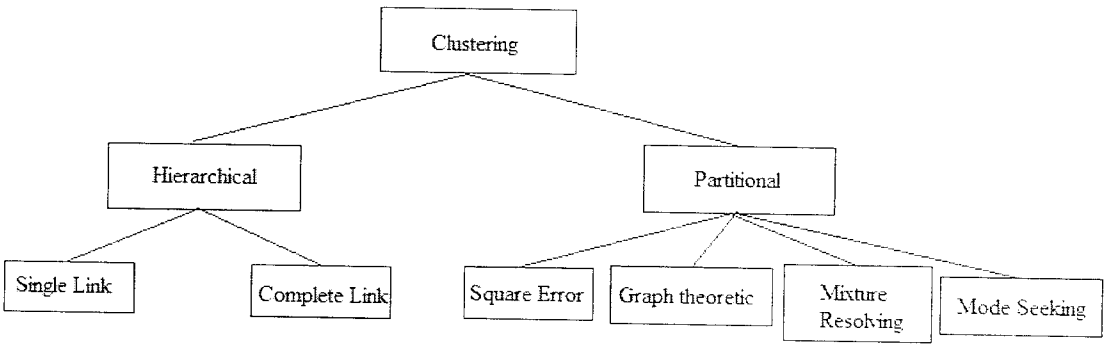


Figure 1.1 Taxonomy of clustering approaches

Agglomerative vs divisive:

This aspect relates to algorithmic structure and operation.An agglomerative approach begins with each pattern in a distinct(singleton)cluster,and successively merges clusters together until a stopping criterion is satisfied.A divisive method begins with all patterns in a single cluster and performs splitting until a stopping criterion is met.

Monothetic vs polythetic:

This aspect relates to the sequential or simultaneous use of features in the clustering process.Most algorithms are polythetic;that is,all features enter into the computation of distances between patterns,and decisions are based on those distances.A simple monothetic

algorithm reported in(Anderberg M.R,1973)considers features sequentially to divide the given collection of patterns.

Hard vs fuzzy:

A hard clustering algorithm allocates each pattern to a single cluster during its operation and in its output.A fuzy clustering method assigns degrees of membership in several clusters to each input pattern.A fuzzy clustering can be converted to a hard clustering by assigning each pattern to the cluster with the largest measure of membership.

Deterministic vs stochastic:

This issue is most relevant to partitional approaches designed to optimize a squared error function.This optimization can be accomplished using traditional techniques or through a random search of the state space consisting of all possible labeling.

Incremental vs Non-incremental:

This issue arises when the pattern set to be clustered is large,and constraints on execution time or memory space affect the architecture of the algorithm.The early history of clustering methodology does not contain many examples of clustering algorithms designed to work with large data sets,but the advent of data mining has fostered the development of clustering algorithms that minimize the number of scans through the pattern set,reduce the number of patterns examined during execution,or reduce the size of data structures used in the algorithm's operations.

Partitional clustering:

A clustering algorithm obtains a single partition of the data instead of a clustering structure,such as the dendrogram produced by a hierarchical technique.Partitional methods have advantages in applications invólving large datasets for which the construction of a dendrogram is computationally prohibitive.A problem accompanying the use of a partitional algorithm is the choice of the number of desired output clusters.The partitional techniques usually produce clusters by optimizing a criterioin function defined either locally(on a subset of the patterns).combinatorial search of the set of possible labeling for an optimum value of a criterion is clearly computationally prohibitive.In practice,therefore,the algorithm is typically run multiple times with different starting states,and the best configuration obtained from all of the runs is used as the output clustering.

Squared error clustering method:

1.Select an initial partition of the patterns with a fixed number of clusters and cluster centroids.

2.Assign each pattern to its closest cluster center and compute the new cluster centers as the centroids of the clusters.Repeat this step until convergence is achieved,i.e.,until the cluster membership is stable.

3.Merge and split clusters based on some heuristic information,optionally repeating step 2

Self Organising Map Method :

A self-organizing map (SOM) or self-organizing feature map (SOFM) is a type of artificial neural network that is trained using unsupervised learning to produce a low-dimensional (typically two-dimensional),

a map. Self-organizing maps are different from other artificial neural networks in the sense that they use a neighborhood function to preserve the topological properties of the input space.This makes SOM useful for visualizing low-dimensional views of high-dimensional data, akin to multidimensional scaling. The model was first described as an artificial neural network by the Finnish professor Teuvo Kohonen, and is sometimes called a Kohonen map.

Like most artificial neural networks, SOMs operate in two modes: training and mapping. Training builds the map using input examples. It is a competitive process, also called vector quantization. Mapping automatically classifies a new input vector.

A self-organizing map consists of components called nodes or neurons. Associated with each node is a weight vector of the same dimension as the input data vectors and a position in the map space. The usual arrangement of nodes is a regular spacing in a hexagonal or rectangular grid. The self-organizing map describes a mapping from a higher dimensional input space to a lower dimensional map space. The procedure for placing a vector from data space onto the map is to find the node with the closest weight vector to the vector taken from data space and to assign the map coordinates of this node to our vector.

While it is typical to consider this type of network structure as related to feedforward networks where the nodes are visualized as being attached, this type of architecture is fundamentally different in arrangement and motivation.

Useful extensions include using toroidal grids where opposite edges are connected and using large numbers of nodes. It has been shown that while self-organizing maps with a small number of nodes behave in a way that is similar to K-means, larger self-organizing maps rearrange data in a way that is fundamentally topological in character.

It is also common to use the U-matrix. The U-matrix value of a particular node is the average distance between the node and its closest neighbors . In a square grid for instance, we might consider the closest 4 or 8 nodes, or six nodes in a hexagonal grid.

Large SOMs display properties which are emergent. In maps consisting of thousands of nodes, it is possible to perform cluster operations on the map itself.

LITERATURE REVIEW

2.LITERATURE REVIEW

2.1 PROBLEM DOMAIN

In recent years, the development of information systems in every field such as business, academics, and medicine,and the amount of stored data have increased year by year.Accumulation is advanced to document data by not the exception but various fields. Document data provides valuable findings to not only domain experts in headquarter sections but also novice users on particular domains such as day trading,news readings and so on.

Hence, the detection of new phrases and words has become very important. In order to realize such detection, emergent term detection (ETD)methods have been developed.However, because the frequency of the word was used in earlier methods, detection was difficult as long as the word that became an object did not appear. In addition, most conventional methods do not consider the nature of terms and important indices separately.This causes difficulties in text mining applications,such as limitations on the extensionality of time direction,time consuming post processing and generality expansions.

After considering these problems,we focus on temporal changes of importance indices of phrases and their temporal patterns.Temporal change of the important indices of extracted phrases is paid attention so that a specialist may recognize emergent terms and/or such fields.The detected terms and their patterns lead to capture human recognition behind the given documents.

2.2 DRAWBACKS OF EXISTING SYSTEM

There exist some conventional studies on the detection of emergent

temporal trends of words. Then, by applying various metrics such as frequency, n-gram , and tf-idf , researchers developed some emergent term detection (ETD) methods.There is a method for finding emergent theme patterns on the basis of a finite state machine by using Hidden Markov Model (HMM) as one of the advanced ETD method. Topic modeling is a related method from the viewpoint of temporal text analysis. In these methods, researchers consider the changes in each particular index of the terms rather than considering the nature of the terms in each language model.Further, in the field of natural language processing, there are studies to find out meaningful terms in a document . One method to do so is based on $\chi 2$ statistics of co-occurrence of nouns. There is a proposed method of determining meaningful terms on the basis of adjacent frequency of compound nouns. By focusing on the methods for finding out meaningful terms consisting of two or more words on the basis of co-occurrence,researchers suggested a method for extracting technical terms consisting of co-existing nouns by calculating $\chi 2$ statistics on a contingency matrix of occurrences of each pair of nouns in a given corpus.In conventional studies on the detection of emergent words and/or phrases in documents such as Web pages and particular electronic message boards, researchers did not explicitly treat the trends of the calculated indices of words and/or phrases. However, on the basis of two different techniques, we consider a method for detecting temporal trends of phrases that consist of from two to nine words.We have focused on short phrases because a considerably long phrase may be a pattern including grammatical structure and anonymous words.

2.3 PROPOSED SYSTEM

We describe a method for detecting various temporal trends of technical terms by using multiple importance indices consisting of the following three subprocesses:

1. Technical term extraction in a corpus
2. Importance indices calculation
3. Trend detection

There are some conventionalmethods of extracting technical terms in a corpus on the basis of each particular importance index. Although these methods calculate each index in order to extract technical terms, information about the importance of each term is lost by cutting off the information with a threshold value. We suggest separating term determination and temporal trend detection based on importance indices. By separating these phases, we can calculate multiple types of importance indices in order to obtain a dataset consisting of the values of these indices for each term. Subsequently, we can apply many types of temporal analysis methods to the dataset based on statistical analysis,clustering, and machine learning algorithms.

2.4 ADVANTAGES

First, the system determines terms in a given corpus.There are two reasons why we introduce term extraction methods before calculating importance indices. One is that the cost of building a dictionary for each particular domain is very expensive task. The other is that new concepts need to be detected in a given temporal corpus. Especially, a new concept is often described in the document for which the character is needed at the right time in using the combination of existing words. Therefore, we apply a term extraction method that is based on the adjacent frequency of compound nouns.

1.5 SCOPE OF THE PROJECT

In the recent period any analyst may interested in knowing the latest trends. The proposed system is used to detect the techniques which had a great attention by the researchers over a particular period of time. It helps to know the subsiding trend and for the future enhancement.

SYSTEM ANALYSIS

3. SYSTEM ANALYSIS

3.1 PROJECT DESCRIPTION:

We propose a method for detecting trends of phrases by combining term extraction methods,importance indices of terms ,and trend analysis methods.Then by considering the titles of four data mining relating conferences as examples,two kinds of temporal trends of extracted phrases based on two important indices are presented.

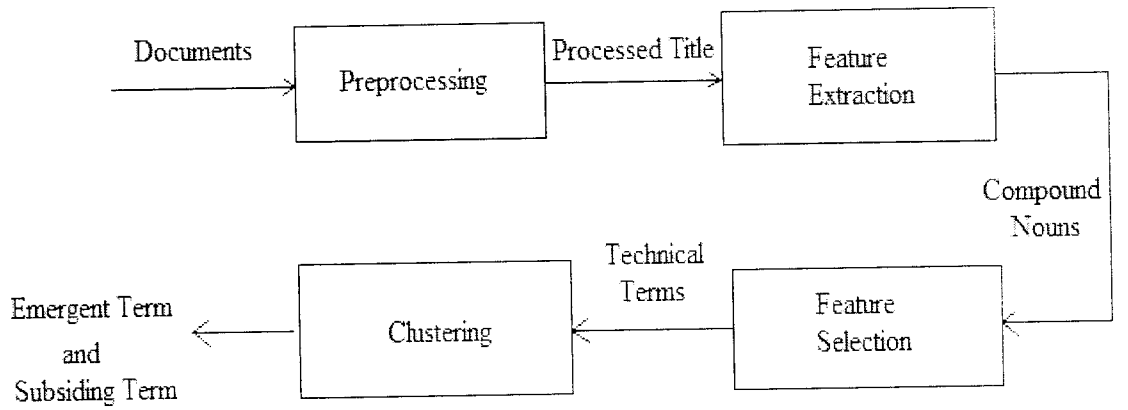


Fig 1:Architecture of Clustering System

3.2 MODULES INVOLVED:

- Stop words removal
- Feature Extraction
- Feature Selection
- Clustering

3.2.1 STOP WORDS REMOVAL

Stopwords are common words that carry less important meaning than keywords. stopwords drive much less traffic than keywords.High stopword density can make the content look less important.

The titles are extracted from the given conferences and each term of the title is compared with the stored stop words such as articles,prepositions,conjunctions.The terms that are matched with the stored stop words are removed.Titles without stop words are obtained as the output. The flow is given in following flow chart.

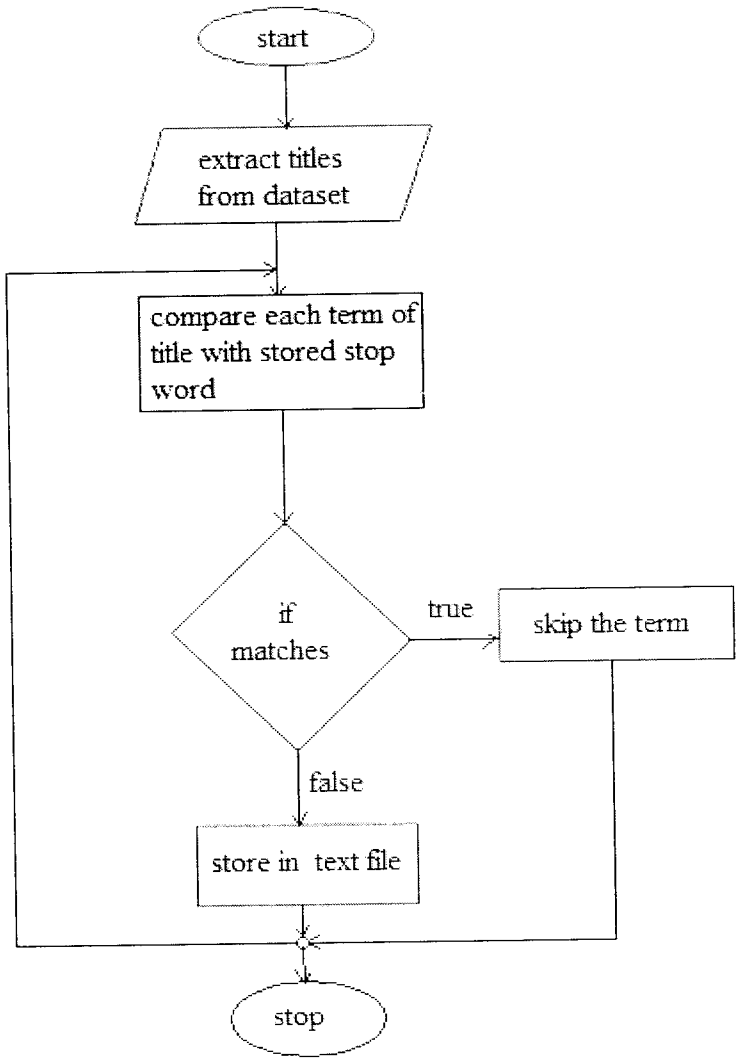


Fig 2: Preprocessing

3.2.2 FEATURE EXTRACTION

There are two reasons for the introduction of term extraction method before calculating important indices.one is that the cost of building a dictionary for each particular domain is very expensive task.The other is that new concepts

need to be detected in a given temporal datacorpus.Especially a new concept is often described in the document for which the character is needed at the right time in using the combination of existing words.Therefore,we apply a term extraction method that is based on adjacent frequency of compound nouns.This method involves the detection of technical terms by using the following values for each candidate compound noun CN:

$FLR(CN) = f(CN) \times (\pi(FL(Ni)+1)(FR(Ni)+1))$

Where f(CN) means frequency of the candidates CN,and $FL(Ni)$ and $FR(Ni)$ indicate the frequencies of the right and the left of each noun Ni.

In order to determine terms in this part of the process,we can also use other term extraction methods and terms/keywords from users.

3.2.3 FEATURE SELECTION

After determining terms in the given corpus,the system calculates multiple important indices of the terms for the documents of each period.As for important indices of words and phrases in a corpus,there are some well known indices.Term frequency divided by inversed document frequency(tf-idf) is one of the popular indices used for measuring the important of the terms.tf-idf for each term t can be defined as follows:

The term frequency can be calculated by:

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}}$$

Where $tf_{i,j}$ is the term frequency, $n_{i,j}$ is the number of occurences of considered term $n_{k,j}$ is the sum of no. of occurences of all terms .The Inverse document frequency can be calculated by:

$$idf_i = \log \frac{|D|}{}$$

Where $|D|$ is the total number of documents, $|\{d : t_i \in d\}|$ is the number of documents where the term t_i appears (that is $n_{i,j} \neq 0$).Then $(\text{tf-idf})_{i,j} = \text{tf}_{i,j} \times \text{idf}_i$.

3.2.4 CLUSTERING

As the input of temporal documents,we used the annual sets of the titles of the following four academic conferences like ACM,AW,CRC and CS.we determine the technical terms by using the term extraction method for each entire set of documents.subsequently,the values of tf-idf are calculated for each term in the annual documents.

To the datasets consisting of temporal temporal values of the important indices,we apply self organising maps to detect the following two temporal trends of the phrases:Emergent and subsiding.As an alternative input for this method,we also detect the trends of keywords provided by the authors of the ACM conference proceedings.we identify both emergent phrases and subsiding phrases in each temporal set of documents.By sorting the temporal degree of the three indices with ascending order,we got the top ten subsiding phrases in the four conferences.The steps of self organising map algorithm are:

- Randomize the map's nodes' weight vectors
- Grab an input vector
- Traverse each node in the map
    - Use Euclidean distance formula to find similarity between the input vector and the map's node's weight vector
    - Track the node that produces the smallest distance (this node is the best matching unit, BMU)

- Update the nodes in the neighbourhood of BMU by pulling them closer to the input vector.

$$\mathbf{Wv}(t+1) = \mathbf{Wv}(t) + \Theta(t)\alpha(t)(\mathbf{D}(t) - \mathbf{Wv}(t))$$

- Increment t and repeat from 2 while $t < \lambda$

  Where t = current iteration,λ = limit on time iteration,**Wv** = current weight vector,**D** = target input, $\Theta(t)$ = restraint due to distance from BMU, $\alpha(t)$ = learning restraint due to time.

The emergent phrases mean that researchers paid attention to their research interests through these phrases.Besides the subsiding phrases mean not decreasing of importance themselves but decreasing of appearances.Considering about the subsiding phrases,they include the phrases related to the name of learning algorithms.This shows that the current emerging techniques such as object oriented,collaborative filtering and frequent pattern mining have been developed based on these basic techniques.

Although these conference shares some emergent and subsiding phrases,characteristic phrases can be also determined.These phrases indicate some characteristics of these conferences,relating to people who have been contributed and attracted for each conference.By comparing these trends of the indices,we can find out not only the remarkable phrases but also similarity and dissimilarity of the conferences.

3.2.5 EXPERIMENTAL RESULTS:

Table 1: Dataset

YEAR	ACM		AW		CRC		CS		
	No. of titles	No. of words	No. of titles	No. of words	No. of titles	No. of words	No of titles	No. of words	TOTAL No. of words
1979	56	466	23	120	32	205	15	76	867
1984	40	349	32	210	23	120	-	-	679
1990	74	615	47	353	27	130	23	120	1218
1995	65	535	56	450	93	727	49	253	1965
1998	93	727	39	222	44	246	56	465	1660
1999	50	411	05	26	35	233	37	220	890
2000	35	233	53	422	24	122	43	245	1022
2001	22	119	40	342	40	342	-	-	803
2002	52	420	25	122	-	-	48	354	896
2003	62	515	26	119	74	615	45	247	1496
2006	33	213	92	830	63	530	-	-	1573
2008	29	123	55	419	62	515	-	-	1057
TOTAL	611	4726	493	3634	517	3785	316	1980	14,126

This table consists of the number of titles and words in the given four conferences namely ACM, AW, CRC, CS which is taken as the input.

Table 2: Preprocessing and Feature Extraction

Year	Initial no. of words	No. of words after software removal	No. of Compound Nouns
1979	867	802	756
1984	679	639	570
1990	1218	1114	1026
1995	1965	1813	1578
1998	1660	1552	1392
1999	890	818	762
2000	1022	979	930
2001	803	747	612
2002	896	843	750
2003	1496	1280	1242
2006	1573	1331	1128
2008	1057	855	876

The above table consists of total no. of words in the titles of the given four conferences and the output of Preprocessing and Feature Extraction.

Table 3: Feature Selection

Terms	Tf	Tf-idf
Object Oriented	6.557	9.563
Distributed Database	6.346	10.691
Multimedia Systems	6.199	9.375
Multidatabase systems	5.820	7.502
Query Processing	5.370	8.944
Relational Database	4.949	7.871
Parallel Computing	4.427	8.730
Transaction management	4.321	6.387
Genome mapping	3.796	5.638
Object SQL	3.184	5.324

This table consists of top ten compound nouns with highest tf-idf values.

Table 3.1: Tf-idf of top term object oriented for the four conferences

Year	ACM	AW	CRC	CS
1979	3.72	1.34	2.09	0.52
1984	3.35	2.43	1.78	0
1990	4.16	3.02	1.56	1.22
1995	3.97	3.64	5.93	3.84
1998	7.58	3.49	4.36	4.05
1999	3.62	0.27	2.18	2.98
2000	2.33	3.52	1.26	2.76
2001	1.97	2.69	2.38	0
2002	3.21	2.74	0	2.37
2003	4.42	1.32	4.75	2.83
2006	2.14	5.29	4.33	0
2008	1.36	3.55	3.67	0

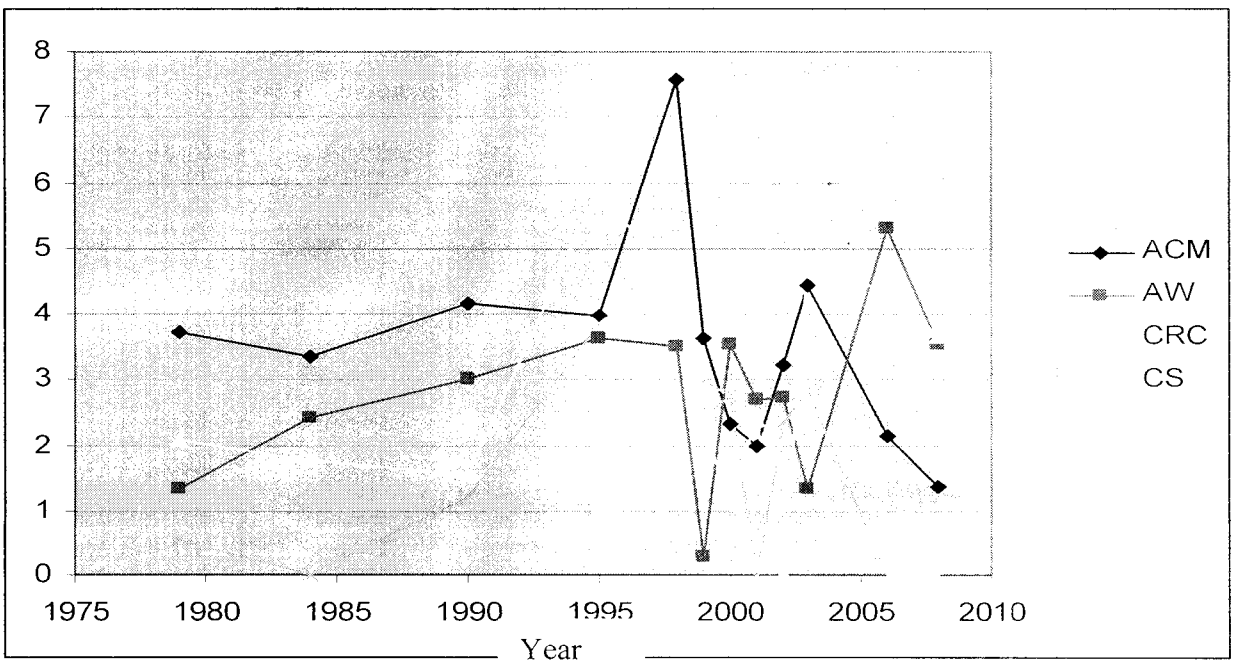


Fig 3: Tf-idf of object oriented for the four conferences

Table 3.2: Tf-idf of last term object sql for the four conferences

Year	ACM	AW	CRC	CS
1979	1.32	1.34	1.09	0.52
1984	2.05	1.14	0.08	0
1990	2.27	1.39	0.56	1.02
1995	2.07	1.26	3.35	2.24
1998	3.35	1.29	2.31	2.15
1999	1.31	0.18	1.05	1.41
2000	1.02	2.21	0.14	1.32
2001	0.35	1.19	1.07	0
2002	1.24	1.05	0	1.37
2003	2.05	0.41	1.11	1.02
2006	1.23	3.81	2.07	0
2008	0.16	2.39	1.35	0

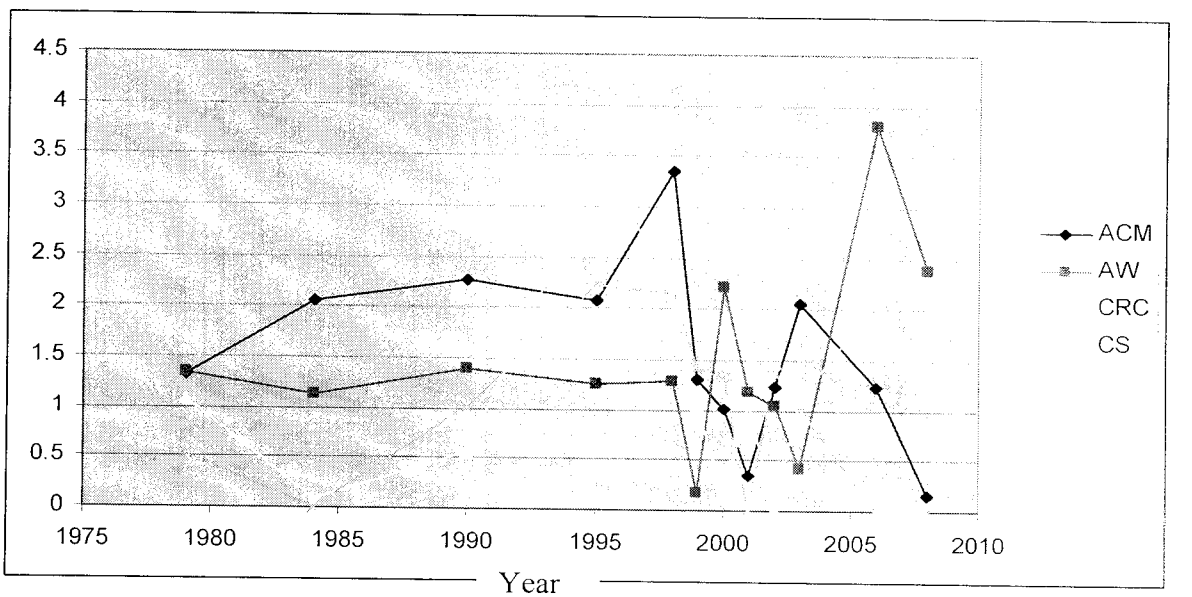


Fig 4: Tf-idf of object sql for the four conferences

Table 4: Clustering

Training Data	No. of Technical Terms	Mean	Standard Deviation	Best Matching Unit
425	857	4.655	1.932	0.971

Table 4.1: Emergent and subsiding terms for the year 1993

EMERGENT TERMS	SUBSIDING TERMS
Data Model	Transaction Time
Indexing Techniques	Temporal reasoning
Temporal Deductive	Relational Database
Deductive Database	Temporal Data
Query Processing	Framework Modelling

SOFTWARE DESCRIPTION

4.1SOFTWARE REQUIREMENTS:

Language	:	c# .NET
Platform	:	Windows Vista home premium

C# Language:

C# is an elegant,simple,modern and type-safe object oriented language derived from C++ and Java that enables developers to build a wide range of secure and robust applications that run on the .NET Framework. C# used to create traditional Windows client applications, XML Web services, distributed components, client-server applications, database applications, and much more. Microsoft Visual C# 2010 provides an advanced code editor, convenient user interface designers, integrated debuggers and many other tools to facilitate rapid application development based on version 3.0 of the C# language and .NET Framework.

.NET PLATFORM:

C# programs run on .NET Framework, an integral component of Windows that includes a virtual execution system called the common language runtime(CLR) and a unified set of class libraries. The CLR is Microsoft's commercial implementation of the common language infrastructure (CLI), an international standard that id the basis for creating execution and development environments in which languages and libraries work together seamlessly.

(IL) that conforms to the CLI specification. The IL code,along with resources such as bitmaps and strings, is stored on disk in an executable file called an assembly, typically with an extension of .exe or .dll. an assembly contains a manifest that provides information in the assembly's types, version, culture and security requirements. When the C# program is executed, the assembly is loaded into the CLR,which might take various actions based on the information in the manifest.

Then if the security requirements are met, the CLR performs just in time (JIT) compilation to convert the IL code into native machine instructions. The CLR also provides other services related to automatic garbage collection, exception handling and resource management. Code that is executed by the CLR is sometimes referred to as "managed code", in contrast to "unmanaged code" which is compiled into native machine language that targets a specific system. Language interoperability is a key feature of the .NET Framework.

MAIN FEATURES OF C#

SIMPLE

1. Pointers are missing in C#.

2. Unsafe operations such as direct memory manipulation are not allowed.

3. In C# there is no usage of "::" or "->" operators.

4. Since it`s on .NET, it inherits the features of automatic memory management and garbage collection.

5. Varying ranges of the primitive types like Integer, Floats etc.

6. Integer values of 0 and 1 are no longer accepted as Boolean values. Boolean values are pure true or false values in C# so no more errors of "="operator and "=="operator. "==" is used for comparison operation and "=" is used for assignment operation.

MODERN

1.C# has been based according to the current trend and is very powerful and simple for building interoperable, scalable, robust applications.

2. C# includes built in support to turn any component into a web service that can be invoked over the Internet from any application running on any platform.

OBJECT ORIENTED

1. C# supports Data Encapsulation, inheritance, polymorphism, interfaces.

2. (int, float, double) are not objects in java but C# has introduces structures(structs) which enable the primitive types to become objects.

TYPE SAFE

1. In C# we cannot perform unsafe casts like convert double to a Boolean.
2. Value types (primitive types) are initialized to zeros and reference types

3. Arrays are zero base indexed and are bound checked.

4. Overflow of types can be checked.

INTEROPERABILITY

1. C# includes native support for the COM and windows based applications.

2. Allowing restricted use of native pointers.

3. Users no longer have to explicitly implement the unknown and other COM interfaces, those features are built in.

4. C# allows the users to use pointers as unsafe code blocks to manipulate your old code.

5. Components from VB NET and other managed code languages and directly be used in C#.

SCALABLE AND UPDATEABLE

1. .NET has introduced assemblies, which are self-describing by means of their manifest. Manifest establishes the assembly identity, version, culture and digital signature etc. Assemblies need not to be register anywhere.

2. To scale our application we delete the old files and updating them with new ones. No registering of dynamic linking library.

3. Updating software components is an error prone task. Revisions made to the code can effect the existing program C# support versioning in the language. Native support for interfaces and method overriding enable complex frame works to be developed and evolved over time.

CONCLUSION

In this project, we have proposed a method to detect remarkable trends of technical terms by focusing on the temporal changes of the importance indices.we implemented the method by combining the technical term extraction method,the three important indices and self organising maps.The case study shows that the temporal changes of the importance indices can detect the trend of each phrase,according to the degrees of the values for each annual set of the titles of the four conferences.Regarding to the result,our method can support to find out remarkable technical phrases in documents based on the temporal changes of the importance indices.

FUTURE ENHANCEMENTS

In the future,we will apply other term extraction methods,important indices,and trend detection methods.As for important indices,we are planning to apply evaluation metrics of information retrieval studies,probability of occurrence of the terms,and statistics values of the terms.To extract the trends,we will introduce temporal pattern recognition methods,such as temporal clustering.Then we will apply this framework to other documents from various domains.

APPENDIX

7.1 SAMPLE CODE

Main:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Xml.Linq;
using System.Threading.Tasks;
using System.IO;
using System.Windows.Forms.DataVisualization.Charting;

namespace Trends
{
  public partial class Form1 : Form
  {
    public Form1()
    {
      InitializeComponent();
    }

    private void button1_Click(object sender, EventArgs e)
    {


    }

    private void button1_Click_1(object sender, EventArgs e)
    {
      XElement X =
XElement.Load(@"G:\Final\Trends\Trends\TestData.xml");

      var LoadedData = from k in X.Elements() select k;
      dataGridView1.DataSource = LoadedData.ToList();
    }
    List<TrendType> Titles = new List<TrendType>();
    private void button2_Click(object sender, EventArgs e)
```

```
        pictureBox1.Visible = true;
        XElement X =
XElement.Load(@"G:\Final\Trends\Trends\TestData.xml");

        var LoadedData = from k in X.Elements("incollection") select new {
C = (string)k.Attribute("key").Value, T = (string)k.Element("title").Value, yr
= (string)k.Element("year").Value };


        foreach (var s in LoadedData)
        {
            TrendType Temp = new TrendType();
            Temp.Title = s.T;
            Temp.Year = s.yr;
            Temp.Conference = s.C;
            Temp.ProcessedTitle = ProcessStopWord.RemoveStopwords(s.T);

            Titles.Add(Temp);
        }

        var docu = from dk in Titles group dk by dk.Year into g select new {
key = g.Key, datas = g };
        foreach (var d in docu)
        {
            treeView1.Nodes.Add(d.key, d.key);
            _numDocs += 1;
            foreach (var dc in d.datas)
            {
                treeView1.Nodes[d.key].Nodes.Add(dc.Title);
            }
        }
        dataGridView2.DataSource = Titles.ToList();
        pictureBox1.Visible = false;


    }
    List<TechnicalTerm> Lists = new List<TechnicalTerm>();
    private void button3_Click(object sender, EventArgs e)
    {


        foreach (TrendType str in Titles)
```

```
            //Convert the string into an array of words
            string[] source = str.ProcessedTitle.Split(new char[] { '.', '?', '!', ' ',
';', ':', ',', '-' }, StringSplitOptions.RemoveEmptyEntries);
            var matchQuery = from word in source
                             select word;
            int wordCount = matchQuery.Count();
            int wc = wordCount;
            var wordmatches = matchQuery.ToList();
            for (int i = 0; i <= wordmatches.Count - 1; i++)
            {
               string s = wordmatches[i].ToString().Trim();
               if ((i > 0) && (i != wordCount))
               {
                  if (s.Length > 2 && wordmatches[i - 1].Length > 2)
                  {
                     s.Replace('(', ' ');
                     TechnicalTerm TempT = new TechnicalTerm();
                     TechnicalTerm TempT1 = new TechnicalTerm();
                     TempT.Terms = wordmatches[i - 1] + " " + s; //+ " " +
wordmatches[i + 1];
                     TempT.Conference = str.Conference;
                     TempT.Year = str.Year;
                     Lists.Add(TempT);
                    /*  if (i-3 < wc)
                     {
                        TempT1.Terms = s + " " + wordmatches[i + 1];
                        TempT1.Conference = str.Conference;
                        TempT1.Year = str.Year;
                        Lists.Add(TempT1);
                     }
                     i++;*/
                  }
               }
            }
         }

         dataGridView3.DataSource = Lists.ToList();
      }
      List<TermFreq> Tf = new List<TermFreq>();
      private void button4_Click(object sender, EventArgs e)
      {
```

```
        string[] s = new string[Lists.Count];
        for (int i = 0; i <= Lists.Count - 1; i++)
        {
            s[i] = Lists[i].Terms.ToString();
        }
        TFIDFMeasure Tdf = new TFIDFMeasure(s);


        for (int i = 0; i <= Lists.Count - 1; i++)
        {
            TermFreq tt = new TermFreq();
            tt.Term = s[i].ToString();
            tt.Freq = Tdf.GetInverseDocumentFrequency(i).ToString();
            tt.Conference = Lists[i].Conference;
            tt.Year = Lists[i].Year;
            Tf.Add(tt);

        }
        TextWriter tw = new StreamWriter(@"G:\Final\Trends\Trends\SampleTestData.txt");
        XElement XDoc = XElement.Load(@"G:\Final\Trends\Trends\XMLFile1.xml");
        foreach (TermFreq tt in Tf)
        {
            tw.WriteLine(tt.Term + "," + tt.Freq);
            XElement X = new XElement("Trend",
                new XAttribute("Year", tt.Year),
                new XAttribute("Conference", tt.Conference),
                new XAttribute("Term", tt.Term),
                new XAttribute("Freq", tt.Freq));
            XDoc.Add(X);
        }
        tw.Close();
        XDoc.Save(@"G:\Final\Trends\Trends\XMLFile1.xml");
        dataGridView4.DataSource = Tf.ToList();

    }
    private int[] _docFreq;
    private int _numDocs = 0;
    public float GetInverseDocumentFrequency(int term, int num)
    {
        _docFreq = new int[num];
```

```
        return Log((float)(_numDocs) / (float)df);
    }
    private float Log(float num)
    {
        return (float)Math.Log(num);//log2
    }

    private void button5_Click(object sender, EventArgs e)
    {

        XElement XDoc = XElement.Load(@"G:\Final\Trends\Trends\XMLFile1.xml");
        var tst = from k in XDoc.Elements("Trend")
                  group k by k.Attribute("Year").Value into G
                  orderby G.Key descending
                  select new
                  {
                      Key = G.Key,
                      Items = from kk in G
                              orderby kk.Attribute("Freq").Value descending
                              select new { Year = (string)kk.Attribute("Year").Value, Conference = (string)kk.Attribute("Conference").Value, Term = (string)kk.Attribute("Term").Value }
                  };
        treeView2.Nodes.Add("Emergent Terms").ForeColor = Color.Red;
        foreach (var t in tst)
        {
            treeView2.Nodes.Add(t.Key, t.Key);
            var tt = t.Items.Take(5);
            int count = t.Items.Count();
            foreach (var v in tt)
            {
                treeView2.Nodes[t.Key].Nodes.Add(v.Term + " [ " + v.Conference + " ].").ForeColor = Color.Green;
            }

        }
        SOM som = new SOM(1, 10, @"G:\Final\Trends\Trends\SampleTestData.txt");
        ChartArea C1 = new ChartArea();
        Axis X = new Axis();
        X.Title = "Points";
```

```
            Y.Title = "Clusters";
            C1.AxisX = X;
            C1.AxisY = Y;
            chart1.ChartAreas.Add(C1);
            Series S1 = new Series("SOM ", 25);
            S1.Color = Color.Red;
            S1.ChartType = SeriesChartType.Line;
            foreach (SOMPLOT s in som.DumpCoordinates())
            {
               try
               {
                  DataPoint Dp = new DataPoint(s.X, s.Y);
                  S1.Points.Add(Dp);
               }
               catch (Exception ex)
               {
               }
            }
            chart1.Series.Add(S1);
            treeView2.ExpandAll();
         }
      }
   }
```

SOM:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;
using System.Globalization;

namespace Trends
{
   public class SOM
   {
      private Neuron[,] outputs;  // Collection of weights.
      private int iteration;      // Current iteration.
      private int length;         // Side length of output grid.
      private int dimensions;     // Number of input dimensions.
      private Random rnd = new Random();
```

```
private List<string> labels = new List<string>();
private List<double[]> patterns = new List<double[]>();

public SOM(int dimensions, int length, string file)
{
   this.length = length;
   this.dimensions = dimensions;
   Initialise();
   LoadData(file);
   NormalisePatterns();
   Train(0.0000001);
   DumpCoordinates();
}

private void Initialise()
{
   outputs = new Neuron[length, length];

   for (int i = 0; i < length; i++)
   {
      for (int j = 0; j < length; j++)
      {
         outputs[i, j] = new Neuron(i, j, length);
         outputs[i, j].Weights = new double[dimensions];

         for (int k = 0; k < dimensions; k++)
         {
            outputs[i, j].Weights[k] = rnd.NextDouble();
         }
      }
   }
}

private void LoadData(string file)
{
   StreamReader reader = File.OpenText(file);
   reader.ReadLine(); // Ignore first line.

   while (!reader.EndOfStream)
   {
      string[] line = reader.ReadLine().Split(',');
      labels.Add(line[0]);
```

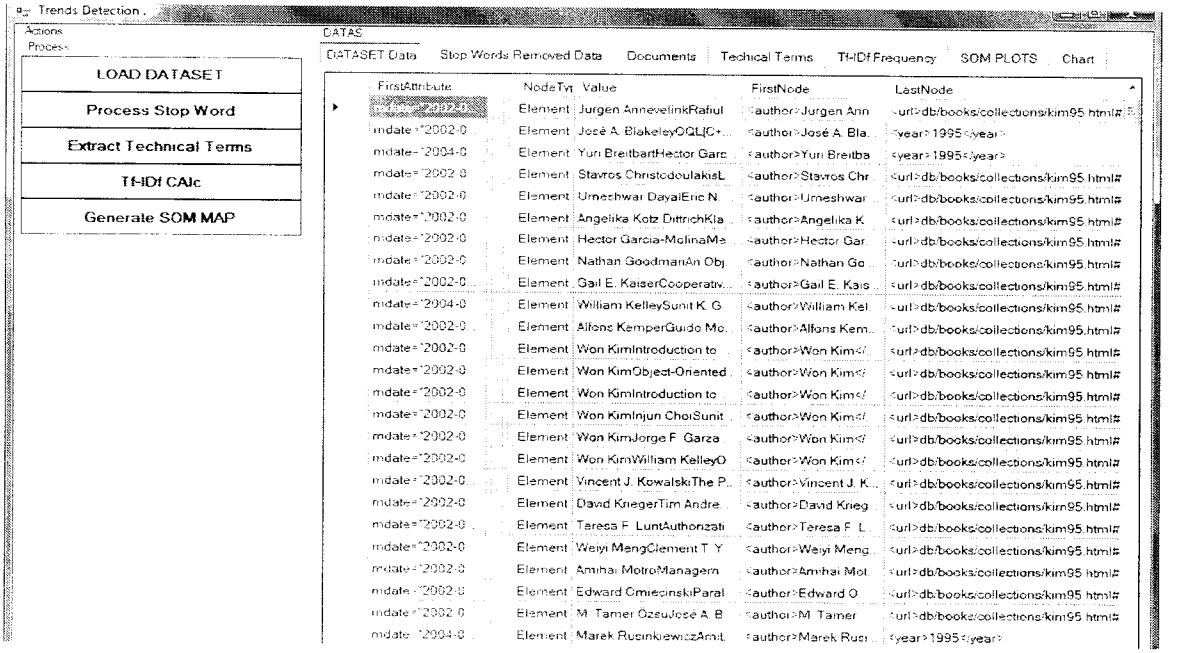
```
            for (int i = 0; i < dimensions; i++)
            {
            inputs[i]          =          double.Parse(line[i+1]);          //line[i+1],
CultureInfo.GetCultureInfo("en-US"));
            }
            patterns.Add(inputs);
        }
        reader.Close();
    }

    private void NormalisePatterns()
    {
        for (int j = 0; j < dimensions; j++)
        {
            double sum = 0;
            for (int i = 0; i < patterns.Count; i++)
            {
                sum += patterns[i][j];
            }

            double average = sum / patterns.Count;

            for (int i = 0; i < patterns.Count; i++)
            {
                patterns[i][j] = patterns[i][j] / average;
            }
        }
    }

    private void Train(double maxError)
    {
        double currentError = double.MaxValue;

        while (currentError > maxError)
        {
            currentError = 0;
            List<double[]> TrainingSet = new List<double[]>();
            foreach (double[] pattern in patterns)
            {
                TrainingSet.Add(pattern);
            }
```

```
            for (int i = 0; i < patterns.Count; i++)
            {
                double[] pattern = TrainingSet[rnd.Next(patterns.Count - i)];
                currentError += TrainPattern(pattern);
                TrainingSet.Remove(pattern);
            }
        }
    }

    private double TrainPattern(double[] pattern)
    {
        double error = 0;
        Neuron winner = Winner(pattern);
        for (int i = 0; i < length; i++)
        {
            for (int j = 0; j < length; j++)
            {
                error += outputs[i, j].UpdateWeights(pattern, winner, iteration);
            }
        }
        iteration++;
        return Math.Abs(error / (length * length));
    }
List<SOMPLOT> Sp = new List<SOMPLOT>();
    public List<SOMPLOT> DumpCoordinates()
    {
        for (int i = 0; i < patterns.Count; i++)
        {SOMPLOT S = new SOMPLOT();
            try
            {

                Neuron n = Winner(patterns[i]);
                S.Label = labels[i];
                S.X = n.X;
                S.Y = n.Y;

            }
            catch (Exception ex)
            {
            }
            Sp.Add(S);
        }
```

```
    }

    private Neuron Winner(double[] pattern)
    {
        Neuron winner = null;
        double min = double.MaxValue;

        for (int i = 0; i < length; i++)
            for (int j = 0; j < length; j++)
            {
                double d = Distance(pattern, outputs[i, j].Weights);
                if (d < min)
                {
                    min = d;
                    winner = outputs[i, j];
                }
            }
        return winner;
    }

    private double Distance(double[] vector1, double[] vector2)
    {
        double value = 0;
        for (int i = 0; i < vector1.Length; i++)
        {
            value += Math.Pow((vector1[i] - vector2[i]), 2);
        }
        return Math.Sqrt(value);
    }
}

public class Neuron
{
    public double[] Weights;
    public int X;
    public int Y;
    private int length;
    private double nf;

    public Neuron(int x, int y, int length)
    {
        X = x;
```

```
        this.length = length;
        nf = 1000 / Math.Log(length);
    }

    private double Gauss(Neuron win, int it)
    {
        double distance=0;
        try
        {
            distance = Math.Sqrt(Math.Pow(win.X - X, 2) + Math.Pow(win.Y
- Y, 2));

        }
        catch (Exception ex)
        {
        }
        return  Math.Exp(-Math.Pow(distance,  2)  /  (Math.Pow(Strength(it),
2)));
    }

    private double LearningRate(int it)
    {
        return Math.Exp(-it / 1000) * 0.1;
    }

    private double Strength(int it)
    {
        return Math.Exp(-it / nf) * length;
    }

    public double UpdateWeights(double[] pattern, Neuron winner, int it)
    {
        double sum = 0;

        for (int i = 0; i < Weights.Length; i++)
        {
            double delta = LearningRate(it) * Gauss(winner, it) * (pattern[i] -
Weights[i]);
            Weights[i] += delta;
            sum += delta;
        }
        return sum / Weights.Length; }    }  }
```

DATASET:


Trends Detection
Actions
Process
LOAD DATASET
Process Stop Word
Extract Technical Terms
Tf-IDf CAlc
Generate SOM MAP
DATAS
DATASET Data
Stop Words Removed Data
Documents
Technical Terms
Tf-IDf Frequency
SOM PLOTS
Chart
FirstAttribute
NodeTyp
Value
FirstNode
LastNode

s Detection .

DATAS

LOAD DATASET

Process Stop Word

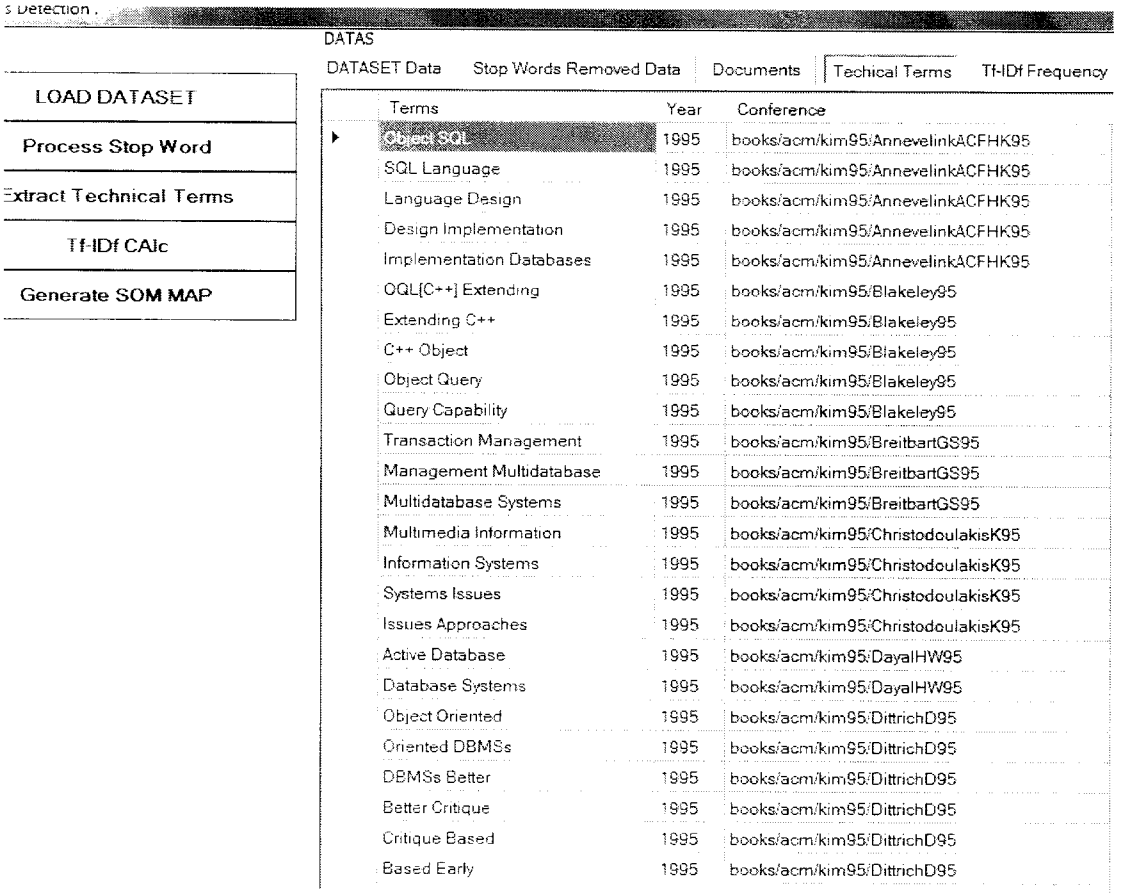
Extract Technical Terms

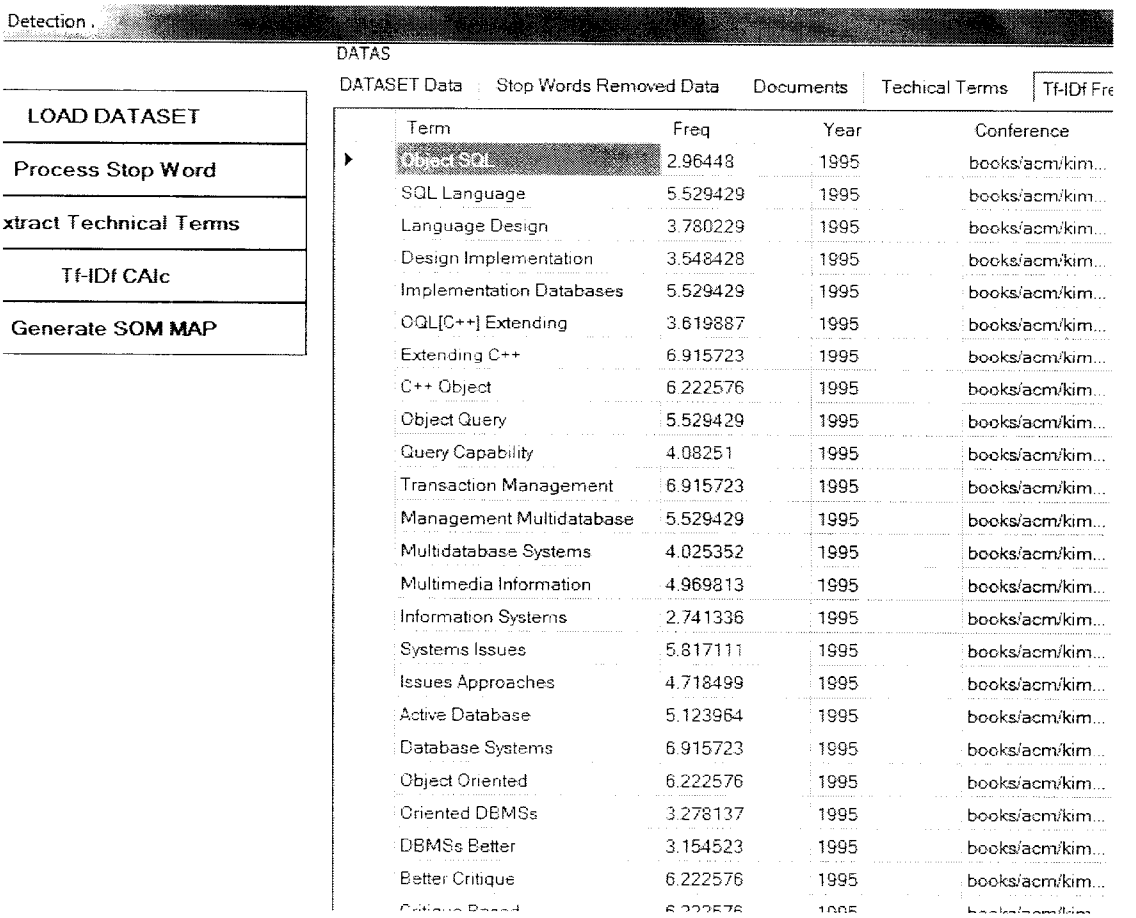
Tf-IDf CAlc

Generate SOM MAP

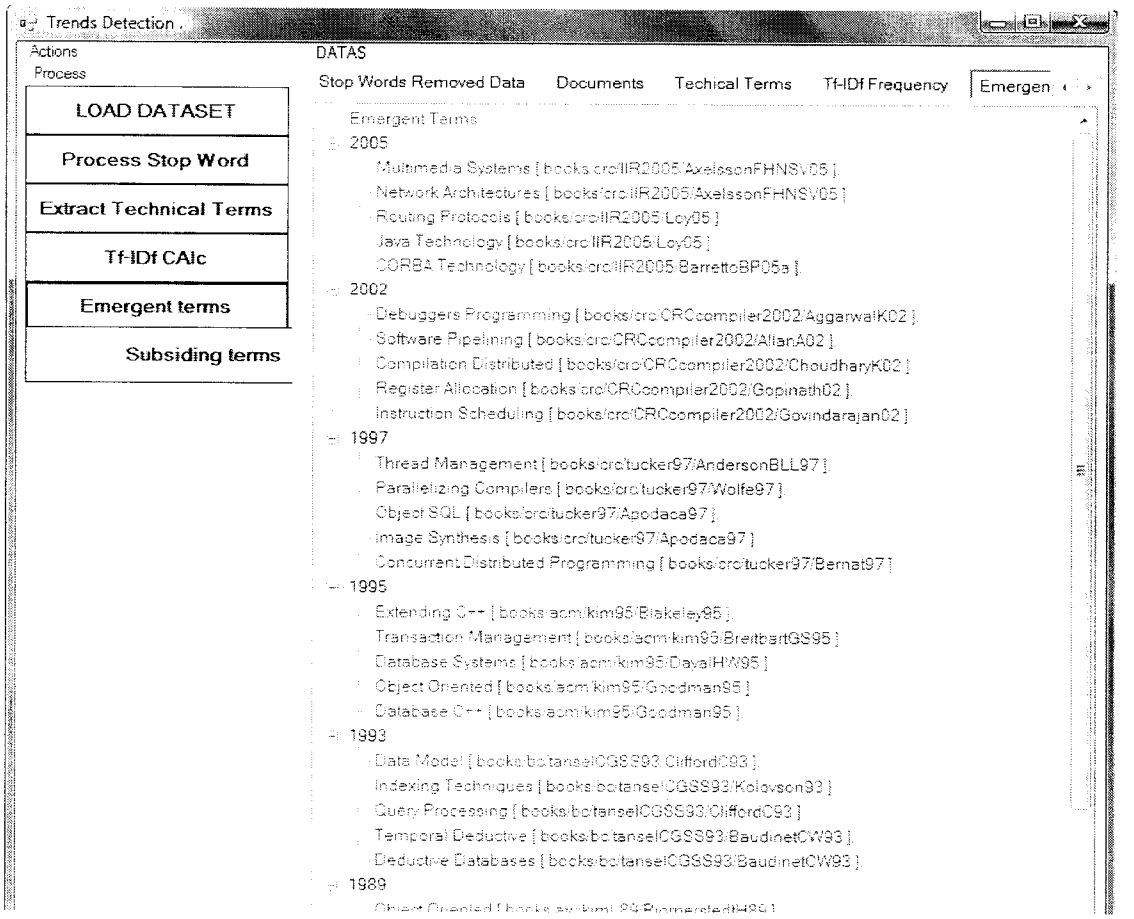
DATASET Data | Stop Words Removed Data | Documents | Techical Terms | Tf-IDf Frequen

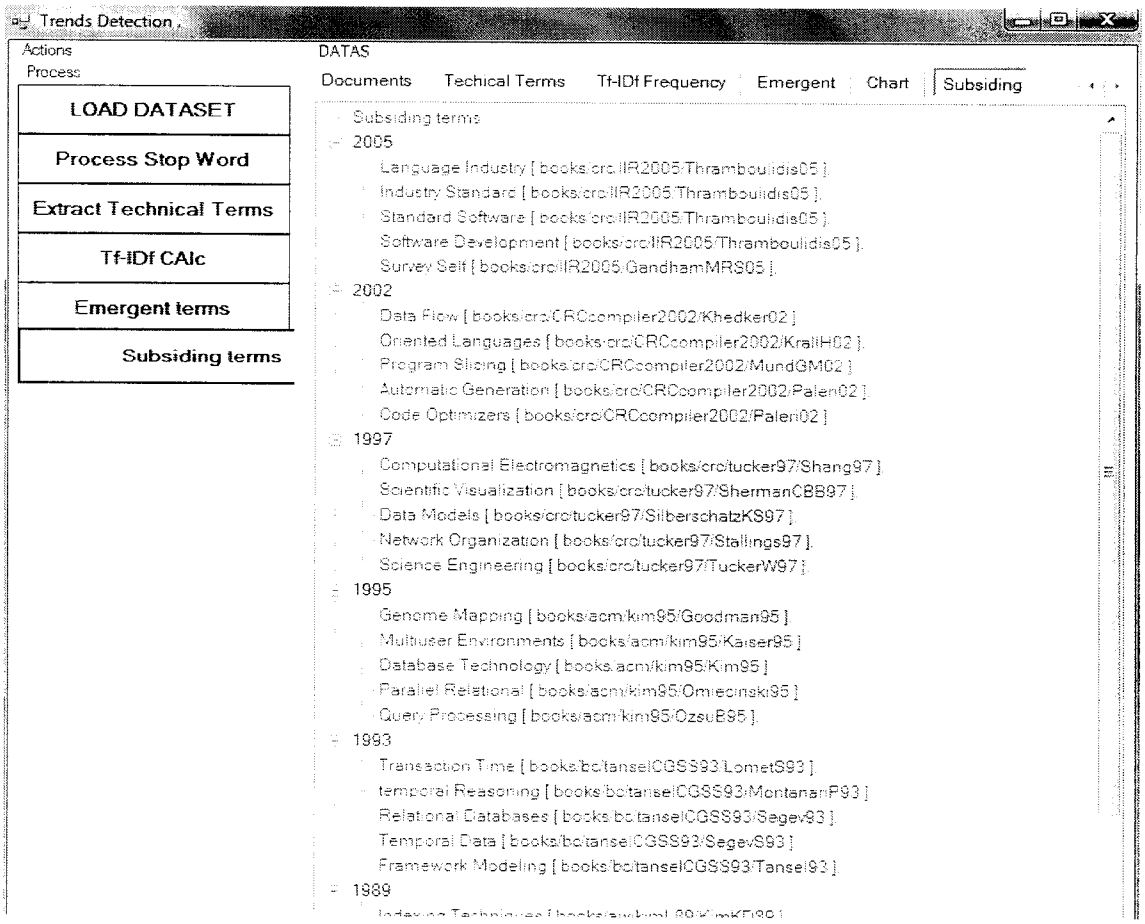
Title	Year	Conference	ProcessedTitle
Object SQL - A ...	1995	books/acm/kim...	Object SQL - Language Design Implement...
OQL[C++]: Exte...	1995	books/acm/kim...	OQL[C++]: Extending C++ Object Query Ca...
Transaction M...	1995	books/acm/kim...	Transaction Management Multidatabase ...
Multimedia Inf...	1995	books/acm/kim...	Multimedia Information Systems: Issues A...
Active Databas...	1995	books/acm/kim...	Active Database Systems
Where Object-...	1995	books/acm/kim...	Object-Oriented DBMSs Better: Critique Ba...
Distributed Dat...	1995	books/acm/kim...	Distributed Databases
An Object-Orie...	1995	books/acm/kim...	Object-Oriented DBMS War Story: Develop...
Cooperative Tr...	1995	books/acm/kim...	Cooperative Transactions Multiuser Enviro...
Schema Archit...	1995	books/acm/kim...	Schema Architecture UniSQL/M Multidata...
Physical Object...	1995	books/acm/kim...	Physical Object Management
Introduction to ...	1995	books/acm/kim...	Introduction 1: Next-Generation Database ...
Object-Oriente...	1995	books/acm/kim...	Object-Oriented Database Systems: Promi...
Introduction to ...	1995	books/acm/kim...	Introduction 2: Technology Interoperating L...
On Resolving ...	1995	books/acm/kim...	Resolving Schematic Heterogeneity Multid...
Requirements ...	1995	books/acm/kim...	Requirements Performance Benchmark O...
On View Supp...	1995	books/acm/kim...	View Support Object-Oriented Databases ...
The POSC Sol...	1995	books/acm/kim...	POSC Solution Managing EData
C++ Bindings t...	1995	books/acm/kim...	C++ Bindings Object Database
Authorization i...	1995	books/acm/kim...	Authorization Object-Oriented Databases
Query Processi...	1995	books/acm/kim...	Query Processing Multidatabase Systems
Management ...	1995	books/acm/kim...	Management Uncerainty database Systems
Parallel Relati...	1995	books/acm/kim...	Parallel Relational Database Systems
Query Processi...	1995	books/acm/kim...	Query Processing Object-Oriented Databa...
Specification a...	1995	books/acm/kim...	Specification Execution Transactional Wor...
Spatial Data St...	1995	books/acm/kim...	Spatial Data Structures
Spatial Data M...	1995	books/acm/kim...	Spatial Data Models Query Processing


s Detection .
DATAS
LOAD DATASET
Process Stop Word
Extract Technical Terms
Tf-IDf CAlc
Generate SOM MAP
DATASET Data
Stop Words Removed Data
Documents
Techical Terms
Tf-IDf Frequency
Terms
Year
Conference
Object SQL 1995 books/acm/kim95/AnnevelinkACFHK95
SQL Language 1995 books/acm/kim95/AnnevelinkACFHK95
Language Design 1995 books/acm/kim95/AnnevelinkACFHK95
Design Implementation 1995 books/acm/kim95/AnnevelinkACFHK95
Implementation Databases 1995 books/acm/kim95/AnnevelinkACFHK95
OQL[C++] Extending 1995 books/acm/kim95/Blakeley95
Extending C++ 1995 books/acm/kim95/Blakeley95
C++ Object 1995 books/acm/kim95/Blakeley95
Object Query 1995 books/acm/kim95/Blakeley95
Query Capability 1995 books/acm/kim95/Blakeley95
Transaction Management 1995 books/acm/kim95/BreitbartGS95
Management Multidatabase 1995 books/acm/kim95/BreitbartGS95
Multidatabase Systems 1995 books/acm/kim95/BreitbartGS95
Multimedia Information 1995 books/acm/kim95/ChristodoulakisK95
Information Systems 1995 books/acm/kim95/ChristodoulakisK95
Systems Issues 1995 books/acm/kim95/ChristodoulakisK95
Issues Approaches 1995 books/acm/kim95/ChristodoulakisK95
Active Database 1995 books/acm/kim95/DayalHW95
Database Systems 1995 books/acm/kim95/DayalHW95
Object Oriented 1995 books/acm/kim95/DittrichD95
Oriented DBMSs 1995 books/acm/kim95/DittrichD95
DBMSs Better 1995 books/acm/kim95/DittrichD95
Better Critique 1995 books/acm/kim95/DittrichD95
Critique Based 1995 books/acm/kim95/DittrichD95
Based Early 1995 books/acm/kim95/DittrichD95


Detection .
DATAS
LOAD DATASET
Process Stop Word
xtract Technical Terms
Tf-IDf CAlc
Generate SOM MAP
DATASET Data
Stop Words Removed Data
Documents
Techical Terms
Tf-IDf Fre
Term
Freq
Year
Conference
Object SQL 2.96448 1995 books/acm/kim...
SQL Language 5.529429 1995 books/acm/kim...
Language Design 3.780229 1995 books/acm/kim...
Design Implementation 3.548428 1995 books/acm/kim...
Implementation Databases 5.529429 1995 books/acm/kim...
OQL[C++] Extending 3.619887 1995 books/acm/kim...
Extending C++ 6.915723 1995 books/acm/kim...
C++ Object 6.222576 1995 books/acm/kim...
Object Query 5.529429 1995 books/acm/kim...
Query Capability 4.08251 1995 books/acm/kim...
Transaction Management 6.915723 1995 books/acm/kim...
Management Multidatabase 5.529429 1995 books/acm/kim...
Multidatabase Systems 4.025352 1995 books/acm/kim...
Multimedia Information 4.969813 1995 books/acm/kim...
Information Systems 2.741336 1995 books/acm/kim...
Systems Issues 5.817111 1995 books/acm/kim...
Issues Approaches 4.718499 1995 books/acm/kim...
Active Database 5.123964 1995 books/acm/kim...
Database Systems 6.915723 1995 books/acm/kim...
Object Oriented 6.222576 1995 books/acm/kim...
Oriented DBMSs 3.278137 1995 books/acm/kim...
DBMSs Better 3.154523 1995 books/acm/kim...
Better Critique 6.222576 1995 books/acm/kim...

EMERGENT TERMS:




Trends Detection
Actions
Process
LOAD DATASET
Process Stop Word
Extract Technical Terms
Tf-IDf CAlc
Emergent terms
Subsiding terms
DATAS
Documents
Techical Terms
Tf-IDf Frequency
Emergent
Chart
Subsiding
Subsiding terms
2005
Language Industry [books/crc/IR2005/Thramboulidis05].
Industry Standard [books/crc/IR2005/Thramboulidis05].
Standard Software [books/crc/IR2005/Thramboulidis05].
Software Development [books/crc/IR2005/Thramboulidis05].
Survey Self [books/crc/IR2005/GandhamMRS05].
2002
Data Flow [books/crc/CRCcompiler2002/Khedker02]
Oriented Languages [books/crc/CRCcompiler2002/KrallH02].
Program Slicing [books/crc/CRCcompiler2002/MundGM02]
Automatic Generation [books/crc/CRCcompiler2002/Palen02].
Code Optimizers [books/crc/CRCcompiler2002/Palen02]
1997
Computational Electromagnetics [books/crc/tucker97/Shang97].
Scientific Visualization [books/crc/tucker97/ShermanCBB97].
Data Models [books/crc/tucker97/SilberschatzKS97].
Network Organization [books/crc/tucker97/Stallings97].
Science Engineering [books/crc/tucker97/TuckerW97].
1995
Genome Mapping [books/acm/kim95/Goodman95]
Multiuser Environments [books/acm/kim95/Kaiser95]
Database Technology [books/acm/kim95/Kim95]
Parallel Relational [books/acm/kim95/Omiecinski95]
Query Processing [books/acm/kim95/OzsuB95].
1993
Transaction Time [books/bc/tansel/CGSS93/LometS93].
temporal Reasoning [books/bc/tansel/CGSS93/MontanariP93]
Relational Databases [books/bc/tansel/CGSS93/Segev93].
Temporal Data [books/bc/tansel/CGSS93/SegevS93]
Framework Modeling [books/bc/tansel/CGSS93/Tansel93].
1989

REFERENCES

1. Hidenao Abe, Shusaku Tsumoto."Detection of Trends of Technical Phrases in Text Mining" . IEEE Transaction 2008

2.The dblp computer science bibliography. http://www.informatik.uni-trier.de/ley/db/.

3. H. Nakagawa. "automatic term recognition based on statistics of compound nouns". *Terminology*, 6(2):195–210, 2000.

4. B. Lent, R. Agrawal, and R. Srikant. Discovering trends in text databases. pages 227–230. AAAI Press, 1997.

5. A. Kontostathis, L. Galitsky, W. M. Pottenger, S. Roy, and D. J. Phelps. A survey of emerging trend detection in textual data mining. *A Comprehensive Survey of Text Mining*, 2003.

8. REFERENCES

1. Suhit Gupta, Gail Kaiser, "DOM based Content Extraction of HTML Document"

2006.

2. Nikolaos K.Papadakis , Dimitrious Skoutas, "A System for Information Extraction Using Clustering Techniques" IEEE Transaction 2005

3.Deepak.P, Deepak Khemani, " UnURL :Unsupervised Learning from URL's,COMAD 2006 .

4.Shankar K Paul, Fellow Varun Talwar, " Webmining in Soft Computing Framework", IEEE Transaction 2002.

5.Raghu Krishnapuram, "Low Complexity Fuzzy Relational Clustering for webmining",IEEE Transaction 2001.

6.www.google.com