

**CLUSTERING BASED METHOD FOR DISCOVERING
EVOLUTIONARY THEME PATTERNS IN A
COLLECTION OF TEXT ARTICLES**

A DISSERTATION

*Submitted in partial fulfillment of the
requirements for the award of the degree*

of

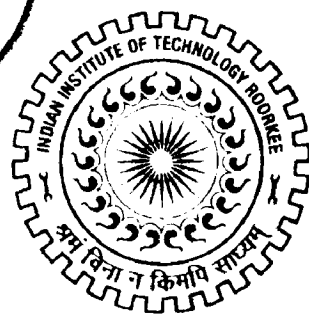
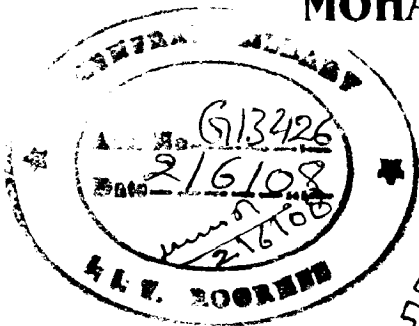
MASTER OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

By

MOHAN KUMAR DALAI



**DEPARTMENT OF ELECTRONICS & COMPUTER ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY ROORKEE
ROORKEE -247 667 (INDIA)**

JUNE, 2007

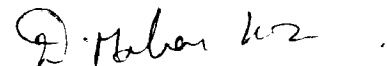
Candidate's Declaration

I hereby declare that the work being presented in the dissertation report titled "Clustering based method for discovering evolutionary theme patterns in a collection of text articles" in partial fulfillment of the requirement for the award of the degree of Master of Technology in Computer Science and Engineering, submitted in the Department of Electronics and Computer Engineering, Indian Institute of Technology Roorkee, is an authentic record of my own work carried out under the guidance of Dr. Kuldip Singh, Professor, Department of Electronics and Computer Engineering, Indian Institute of Technology Roorkee.

I have not submitted the matter embodied in this dissertation report for the award of any other degree.

Dated: 9-06-07

Place: IIT Roorkee.


(Mohan Kumar Dalai)

Certificate

This is to certify that above statements made by the candidate are correct to the best of my knowledge and belief.

Dated: 9/6/07

Place: IIT Roorkee.


Dr. Kuldip Singh.

Professor.

Department of Electronics and
Computer Engineering, IIT Roorkee,
Roorkee -247667 (India).

ACKNOWLEDGEMENTS

I am thankful to Indian Institute of Technology Roorkee for giving me this opportunity. It is my privilege to express thanks and my profound gratitude to my supervisor Prof. Dr. Kuldip Singh for his invaluable guidance and constant encouragement throughout the dissertation. I was able to complete this dissertation in this time due to constant motivation and support obtained from Prof. Dr. Kuldip Singh.

I am also grateful to the staff of software laboratory and Research Scholar's laboratory for their kind cooperation extended by them in the execution of this dissertation. I am also thankful to all my friends who helped me directly and indirectly in completing this dissertation.

Most importantly, I would like to extend my deepest appreciation to my family for their love, encouragement and moral support. Finally I thank God for being kind to me and driving me through this journey.

(Mohan Kumar Dalai)

ABSTRACT

In this thesis work we consider the problem of analyzing the development of a document collection over time without requiring meaningful citation data. Given a collection of time stamped documents, we formulate and explore the following two questions. First, what are the main topics and how do these topics develop over time? Second, what are the documents and who are the authors that are most influential in this process?. We propose methods addressing these questions by taking solely text of the document as input. Because proposed methods use only the text of the documents as input, the methods are applicable to a much wider range of document collections (email, blogs, etc.), most of which lack meaningful citation data. We evaluate our methods on two kinds of data sets one is the documents from the proceedings of the Neural Information Processing Systems (NIPS) conference and the other is collection of news articles. The results show that the methods are effective and that addressing the questions based on the text alone . In fact, the text-based methods sometimes even identify influential papers that are missed by citation analysis.

Table of Contents

Candidate's Declaration & Certificate	i
Acknowledgements	ii
Abstract	iii
Table of Contents	iv
Chapter 1 Introduction and Statement of the Problem	1
1.1 Introduction	1
1.2 Background	1
1.3 Motivation.....	2
1.4 Problem Statement	4
1.5 Organization of the Report	4
Chapter 2 Data mining.....	5
2.1 Introduction.....	5
2.2 What is data mining.....	6
2.2.1 KDD process	7
2.2.2 Data-mining Methods.....	11
2.2.3 Temporal data mining	14
2.3 What is text mining.....	14
2.4 Clustering.....	15
2.4.1 Definition.....	15
2.4.2 Distance measure.....	16
2.4.3 Clustering Methods.....	18
Chapter 3 Design and Implementation	23
3.1 Data set used.....	23
3.2 Vector representation of a document	24
3.3 Term weighing using TFIDF	24
3.4 How do day topics change over time ?.....	26

3.4.1 Method.....	26
3.4.2 K-means clustering algorithm.....	26
3.5 Which are the most influential documents? (method).....	29
3.6 Who are the most influential authors?(method0.....	32
Chapter 4 Results and Discussions.....	33
4.1 How do day topics change over time ? (Results).....	33
4.2 Which are the most influential documents? (Results).....	39
4.3 Who are the most influential authors?(Results).....	40
Chapter 5 Conclusion and Scope for the Future Work	44
References	45
Appendix: Source Code Listing	49

1.1 INTRODUCTION

One of the newest areas of data mining is text mining. In the field of data mining, text mining has been attracting increased interest in the industry. Text mining refers to a collection of methods used to find patterns and create intelligence from unstructured text data. . It has been estimated that 85% of corporate data is of unstructured type.

Many document collections have grown through an interactive and time-dependent process [5]. The word interactive means the documents share the same features (words in this context) and time-dependent means earlier documents shapes documents that followed later, with some documents introducing new ideas that lay the foundation for following documents. Examples of such collections are email repositories, the body of scientific literature, and the web. To access and analyze such collections, it is important to understand how they developed over time. For example, consider a historian trying to get an understanding of the ideas and forces leading to the Iraq war from news articles, or consider the head of a hiring committee trying to understand which scientists had the greatest influence on the development of a discipline.

1.2 Background

There is an enormous growth in the information available in text documents, such as news documents, research articles, web documents etc. Application of text mining techniques [1] to extract useful information from these text based resources has received a lot of interest recently in both academia and industry. Many of these text based documents have been archived, with respect to time of their publication or creation, giving an approximate time of occurrence for the events/information contained in the document.

The time stamps associated with text documents have been employed in discovering how the information in the text documents has evolved over time. In particular, time stamped documents have been collectively analyzed for identifying emerging trends [11, 13, 19] from research articles, detecting and tracking topics from news articles and for extracting timelines of events from articles. The usual approach for extracting temporal information (such as trends) from text documents has been to construct a time series representing the evolution of the keywords/topics in the document set over time.

That means in our representation x-axis contains time(T). A decomposition of the time period T spanning a document set is constructed by decomposing it into equal length subintervals. The document set is partitioned into subsets by assigning documents to each subinterval based on time stamps. For example one may show the graph for a given document collection, time is on x-axis and the number of documents published in that year in y-axis. Text mining functions are applied to each document subset to compute the information deemed significant for the corresponding subinterval. The information computed is a set of keywords/phrases/nouns/topics deemed significant for that subinterval .

1.3Motivation

Consider, for example, the Asian tsunami disaster that happened in the end of 2004. A query to Google News returned more than 80,000 online news articles [5] about this event within one month . It is generally very difficult to navigate through all these news articles, for someone who has not been keeping track of the event but wants to know about this disaster , a summary of this event would be extremely useful.

Ideally, the summary would include both the major topics about the event and the evolution of these themes/topics. For example, the themes may include the report of the happening of the event, the statistics of victims and damage, the aids from the world, and the lessons from the tsunami.

Consider another scenario in the literature domain. There are often hundreds of papers published annually in a research area. A researcher, especially a beginning researcher, often wants to understand how the research topics in the literature have been evolving. For example, if a researcher wants to know about information retrieval research area , both the historical milestones and the recent research trends of information retrieval area would be valuable for him.

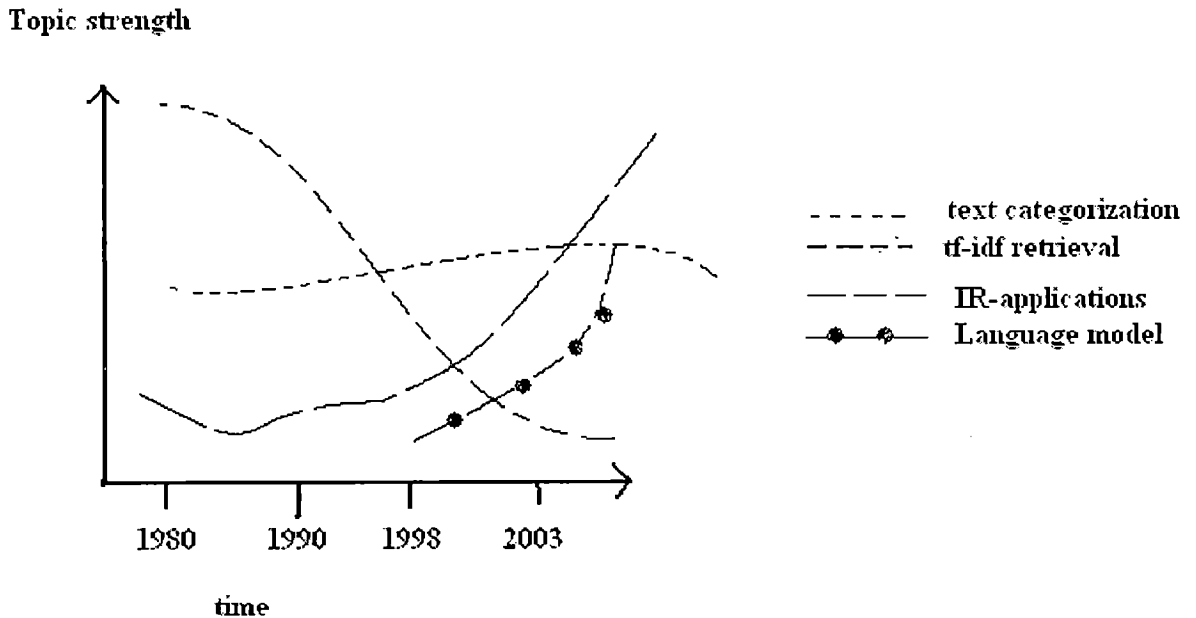


Figure 1: An example of topic strength in IR

A plot, such as the one shown in Figure 1, which visualizes the evolution patterns of research topics, would not only serve as a good summary[5] of the field, but also make it much easier for the researcher to selectively choose appropriate papers to read based on his research interests. In both scenarios, we clearly see a need for discovering evolutionary theme patterns in a text stream. In general, it is often very useful to discover the temporal patterns that may exist in a stream of text articles, a task which we refer to as Temporal Text Mining (TTM) .

Since most information bears some kinds of time stamps, TTM can be expected to have many applications[1] in multiple domains. There are some previous studies [11,13], on

TTM ,but the proposed methods are generally inadequate for generating the evolutionary theme patterns as shown in the two examples above. So, an efficient method is required to generate the evolutionary theme patterns. There are some previous methods on topic detection [18] and identifying key players [9] but they are taking citation information in addition to the text of the document as input

1.4 Problem Statement

In this thesis work, we pose and consider the problem of discovering evolutionary theme pattern in document collection. This problem requires simultaneously understanding what topics are popular and which documents and authors drive the changes in popularity of the topics. In particular we address the following questions:

- What are the key topics in a collection of documents and how did their popularity change over time?
- Which are the most influential documents?
- Who were the authors that significantly drove the evolution of ideas?

In particular, since most collections lack meaningful citation and hyperlink structure, the analysis must be done entirely based on the text in the document.

1.5 Thesis Organization

This dissertation proposes a new and efficient technique for the identifying and visualizing the temporal patterns of given collection of documents. The organization of the dissertation report is as follows:

Chapter 2 reviews what is data mining and knowledge discovery process, text mining, clustering and clustering methods , k-means and clustering algorithm.

Chapter 3 reviews the dataset used and representation of data, and describes the proposed methodology to solve the problem.

Chapter 4 reviews the results and analysis of results.

Chapter 5 concludes the dissertation work and gives suggestions for future work.

2.1 Introduction:

Traditional statistical analysis is performed on data arrayed in spreadsheet format. That is, the data is arrayed in two dimensional matrices where each row represents a record and each column represents a feature or variable. Table 1 provides a sample of such a database. In Table 1, each row represents a claimant. The features are the variables claim number, accident date, claim status, attorney involvement, paid loss, outstanding loss, incurred loss, incurred allocated loss adjustment expenses (ALAE) and claimant state. As seen in Table 1, the data contain two key types of variables, quantitative or numeric variables such as incurred losses and incurred expenses and nominal or categorical variables such as claim status and state. Each numeric value denotes a specific quantity or value for that variable. Each value or category, whether numeric or alphanumeric, of a categorical variable embeds a coding that maps the value to one and only one category. This data is structured data. Structured databases result from intentional design where the variables have proscribed definitions and the values of the variables have proscribed meaning.

Table 1 : Sample Structured Data

Claim No	Accident Date	Status	Attorney	Paid	Out Standing	Incurred ALAE	Incurred Loss	State
19981	1/08/1999	C	Yes	37,284	0	0	37,284	NY
19984	1/16/1999	C	NO	0	0	3	0	NY
20022	1/30/2002	C	NO	195	0	31,807	195	CA
19986	9/19/1998	C	Yes	99,852	0	72	99,852	NJ

when data is unstructured there is no obvious procedure for converting the data which is composed of sequences of characters that vary in length and content in apparently random ways to information that can be used for analysis and prediction. Manual intervention on the part of human beings may be able to convert some unstructured data

to structured features which can be used to perform statistical analysis. Because of the effort required and difficulty of interpreting the unstructured text data, it is typically ignored for doing analysis. If information could be automatically extracted from unstructured data, a significant new source of data could become available to corporations. Text mining refers to a collection of methods used to find patterns and create intelligence from unstructured text data. In the field of data mining, text mining has been attracting increased interest in the industry.

2.2 What is data mining?

Data mining [4,12] refers to extracting or 'mining' knowledge from large amount of data. Data mining methodology extracts hidden predictive information from large data bases. The objective of data mining is to extract valuable information from data to discover the hidden gold. This gold is the valuable information in that data. Small changes in strategy, provided by data mining's discover process, can translate into a difference of millions of dollars to the bottom line.

This new discipline today finds application in a wide and diverse range of business, scientific and engineering scenarios. For example, large databases of loan applications are available which record different kinds of personal and financial information about the applicants (along with their repayment histories). These databases can be mined for typical patterns leading to defaults which can help determine whether a future loan application must be accepted or rejected. Several terabytes of remote-sensing image data are gathered from satellites around the globe. Data mining can help reveal potential locations of some (as yet undetected) natural resources or assist in building early warning systems for ecological disasters like oil slicks etc. Other situations where data mining can be of use include analysis of medical records of hospitals in a town to predict, for example, potential outbreaks of infectious diseases, analysis of customer transactions for market research applications etc. The list of application areas for data mining is large and is bound to grow rapidly in the years to come.

Simply , Data Mining, is the process of automatically searching large volumes of data for patterns using tools such as classification, association rule mining, clustering, etc. Data mining is a complex topic and has links with multiple core fields such as computer science and adds value to rich seminal computational techniques from statistics, information retrieval, machine learning and pattern recognition.

2.2.1 KDD (knowledge discovery in databases) Process

Knowledge discovery in Databases [4,12,26] or data mining is the effort to understand, analyze, and eventually make use of the huge volume of data available. Through the extraction of knowledge in databases, large databases will serve as a rich, reliable source for knowledge generation and verification, and the discovered knowledge can be applied to information management, query processing, decision making, process control and many other applications.

Knowledge Discovery in Databases is defined as the nontrivial process of identifying valid, potentially useful, and ultimately understandable patterns in data. There are several steps in a KDD process: data selection, preprocessing, transformation, data mining, and interpretation/evaluation of results, as shown in figure 2. Data mining is only one step of the process, involving the application of discovery tools to find interesting patterns from targeted data. However, since data mining is the central part of the KDD process, the term data mining and the term knowledge discovery in databases are taken as synonyms to each other.

Why do we need KDD:

The traditional method of turning data into knowledge relies on manual analysis [12] and interpretation. For example, in the health-care industry, it is common for specialists to periodically analyze current trends and changes in health-care data, say, on a quarterly basis. The specialists then provide a report detailing the analysis to the sponsoring healthcare organization; this report becomes the basis for future decision making and planning for health-care management.

In a totally different type of application, planetary geologists sift through remotely sensed images of planets and asteroids, carefully locating and cataloging such geologic objects of interest as impact craters. Be it science, marketing, finance, health care, retail, or any other field, the classical approach to data analysis relies fundamentally on one or more analysts becoming intimately familiar with the data and serving as an interface between the data and the users and products. For these (and many other) applications, this form of manual probing of a data set is slow, expensive, and highly subjective. In fact, as data volumes grow dramatically, this type of manual data analysis is becoming completely impractical in many domains. Databases are increasing in size in two ways:

(1) the number N of records or objects in the database and

(2) the number d of fields or attributes to an object.

Who could be expected to digest millions of records, each having tens or hundreds of fields? We believe that this job is certainly not one for humans; hence, analysis work needs to be automated, at least partially. The need to scale up human analysis capabilities to handling the large number of bytes that we can collect is both economic and scientific. Businesses use data to gain competitive advantage, increase efficiency, and provide more valuable services to customers.

Data we capture about our environment are the basic evidence we use to build theories and models of the universe we live in. Because computers have enabled humans to gather more data than we can digest, it is only natural to turn to computational techniques to help us unearth meaningful patterns and structures from the massive volumes of data. Hence, KDD is an attempt to address a problem that the digital information era made a fact of life for all of us: data overload.

Data Mining: A KDD Process

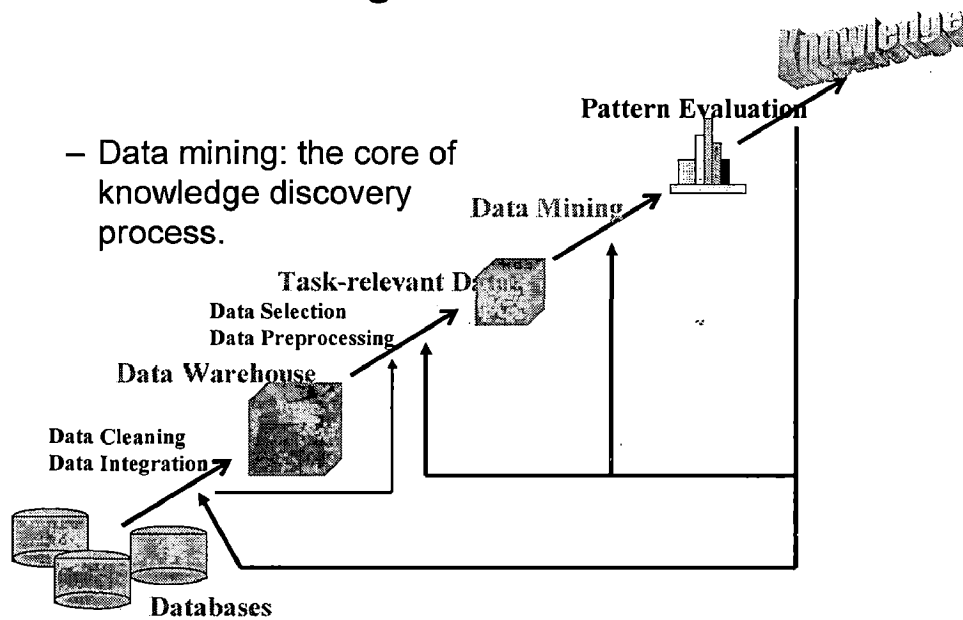


Figure 2: A KDD Process

The KDD process is interactive and iterative, involving numerous steps with many decisions made by the user. Here, we broadly outline some of its basic steps:

First is developing an understanding of the application domain and the relevant prior knowledge and identifying the goal of the KDD process from the customers viewpoint.

Second is creating a target data set: selecting a data set, or focusing on a subset of variables or data samples, on which discovery is to be performed.

Third is data cleaning and preprocessing. Basic operations include removing noise if appropriate, collecting the necessary information to model or account for noise, deciding on strategies for handling missing data fields, and accounting for time-sequence information and known changes.

Fourth is data reduction and projection finding useful features to represent the data depending on the goal of the task. With dimensionality reduction or transformation

methods, the effective number of variables under consideration can be reduced, or invariant representations for the data can be found.

Fifth is matching the goals of the KDD process (step 1) to a particular data-mining method. For example, summarization, classification, regression, clustering, and so on, are described later.

Sixth is exploratory analysis and model and hypothesis selection: choosing the data mining algorithm(s) and selecting method(s) to be used for searching for data patterns. This process includes deciding which models and parameters might be appropriate (for example, models of categorical data are different than models of vectors over the real) and matching a particular data-mining method with the overall criteria of the KDD process (for example, the end user might be more interested in understanding the model than its predictive capabilities).

Seventh is data mining: searching for patterns of interest in a particular representational form or a set of such representations, including classification rules or trees, regression, and clustering. The user can significantly aid the data-mining method by correctly performing the preceding steps.

Eighth is interpreting mined patterns, possibly returning to any of steps 1 through 7 for further iteration. This step can also involve visualization of the extracted patterns and models or visualization of the data given the extracted models.

Ninth is acting on the discovered knowledge: using the knowledge directly, incorporating the knowledge into another system for further action, or simply documenting it and reporting it to interested parties. This process also includes checking for and resolving potential conflicts with previously believed (or extracted) knowledge.

The KDD process can involve significant iteration and can contain loops between any two steps. The basic flow of steps (although not the potential multitude of iterations and loops) is illustrated in figure 2. Most previous work on KDD has focused on step 7, the

data mining. However, the other steps are as important (and probably more so) for the successful application of KDD in practice.

2.2.2 Data-Mining Methods

The two high-level primary goals of data mining in practice tend to be prediction and description. As stated earlier, prediction involves using some variables or fields in the database to predict unknown or future values of other variables of interest, and description focuses on finding human-interpretable patterns describing the data. Although the boundaries between prediction and description are not sharp (some of the predictive models can be descriptive, to the degree that they are understandable, and vice versa), the distinction is useful for understanding the overall discovery goal. The relative importance of prediction and description for particular data-mining applications can vary considerably. The goals of prediction and description can be achieved using a variety of particular data-mining methods which are listed below.

1. Classification is learning a function that maps (classifies) a data item into one of several predefined classes. Examples of classification methods used as part of knowledge discovery applications include the classifying of trends in financial.

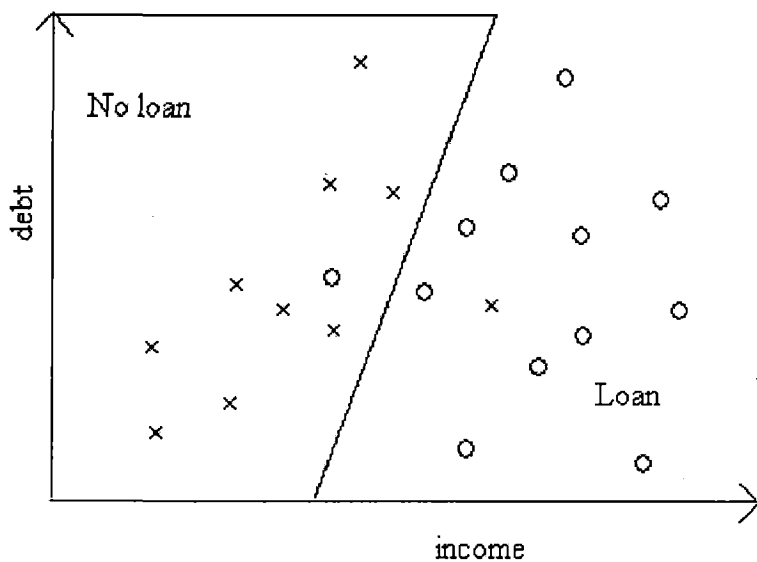


Figure 3: A simple linear classification boundary for loan data set.

markets and the automated identification of objects of interest in large image databases. Figure 3 shows a simple partitioning of the loan data into two class regions; note that it is not possible to separate the classes perfectly using a linear decision boundary. The bank might want to use the classification regions to automatically decide whether future loan applicants will be given a loan or not.

2. Regression is learning a function that maps a data item to a real-valued prediction variable. Regression applications are many, for example, predicting the amount of biomass present in a forest given remotely sensed microwave measurements,

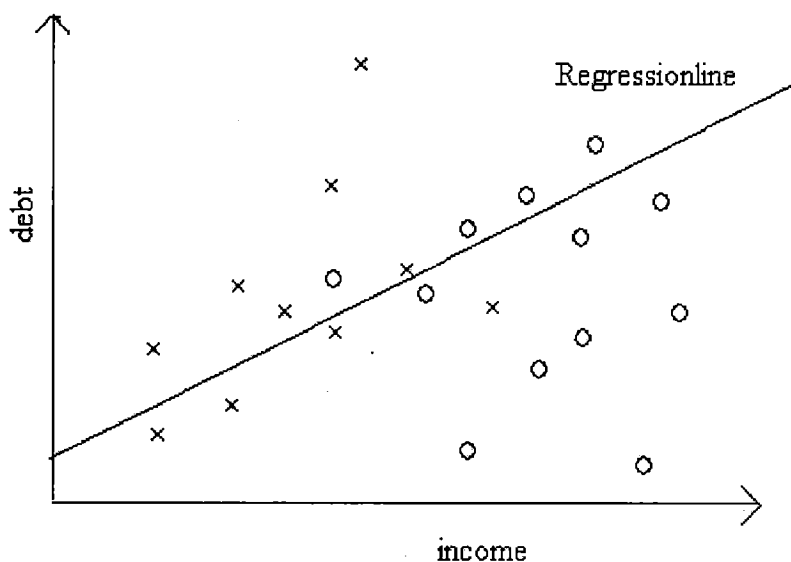


Figure 4: A linear regression for the loan data set

estimating the probability that a patient will survive given the results of a set of diagnostic tests, predicting consumer demand for a new product as a function of advertising expenditure, and predicting time series where the input variables can be time-lagged versions of the prediction variable.

3. Clustering is a common descriptive task, where one seeks to identify a finite set of categories or clusters to describe the data. The categories can be mutually exclusive and exhaustive or consist of a richer representation, such as hierarchical or overlapping categories. Examples of clustering applications in a knowledge discovery context include

discovering homogeneous subpopulations for consumers in marketing databases and identifying subcategories of spectra from infrared sky measurements.

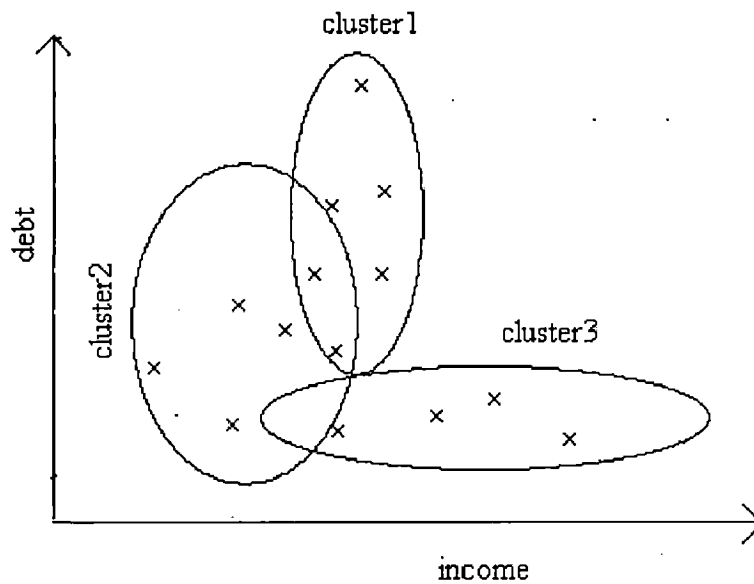


Figure 5 : A simple clustering of loan data set into 3 clusters.

Figure 5 shows a possible clustering of the loan data set into three clusters; note that the clusters overlap, allowing data points to belong to more than one cluster. The original class labels have been replaced by a + to indicate that the class membership is no longer assumed known. Closely related to clustering is the task of probability density estimation, which consists of techniques for estimating from data the joint multivariate probability density function of all the variables or fields in the database

4. Summarization involves methods for finding a compact description for a subset of data. A simple example would be tabulating the mean and standard deviations for all fields. More sophisticated methods involve the derivation of summary rules, multivariate visualization techniques, and the discovery of functional relationships between variables. Summarization techniques are often applied to interactive exploratory data analysis and automated report generation. Dependency modeling consists of finding a model that describes significant dependencies between variables. Dependency models exist at two levels:

(1) the structural level of the model specifies (often in graphic form) which variables are locally dependent on each other

(2) the quantitative level of the model specifies the strengths of the dependencies using some numeric scale.

For example, probabilistic dependency networks use conditional independence to specify the structural aspect of the model and probabilities or correlations to specify the strengths of the dependencies. Probabilistic dependency networks are increasingly finding applications in areas as diverse as the development of probabilistic medical expert systems from databases, information retrieval, and modeling of the human genome. Change and deviation detection focuses on discovering the most significant changes in the data from previously measured or normative values.

2.2.3 Temporal Data mining

Temporal data mining [20] is concerned with data mining of large sequential data sets. By sequential data, we mean data that is ordered with respect to some index. For example, time series constitute a popular class of sequential data, where records are indexed by time. Other examples of sequential data could be text, gene sequences, protein sequences, lists of moves in a chess game etc. Here, although there is no notion of time as such, the ordering among the records is very important and is central to the data description/ modeling.

2.3 What is Text Mining?

Text Mining [1] is the discovery by computer of new, previously unknown information, by automatically extracting information from different written resources. A key element is the linking together of the extracted information together to form new facts or new hypotheses to be explored further by more conventional means of experimentation.

Text mining is a variation on a field called data mining, that tries to find interesting patterns from large databases.

A typical example in data mining is using consumer purchasing patterns to predict which products to place close together on shelves, or to offer coupons for, and so on. For example, if you buy a flashlight, you are likely to buy batteries along with it. A related application is automatic detection of fraud, such as in credit card usage. Analysts look across huge numbers of credit card records to find deviations from normal spending patterns. A classic example is the use of a credit card to buy a small amount of gasoline followed by an overseas plane flight. The claim is that the first purchase tests the card to be sure it is active.

The difference between regular data mining and text mining is that in text mining the patterns are extracted from natural language text rather than from structured databases of facts. Databases are designed for programs to process automatically; text is written for people to read. We do not have programs that can "read" text and will not have such for the foreseeable future. Many researchers think it will require a full simulation of how the mind works before we can write programs that read the way people do.

Temporal Text Mining (TTM)

Temporal Text Mining (TTM) is concerned with discovering temporal patterns in text information collected over time. Since most text information bears some time stamps, TTM has many applications in multiple domains, such as summarizing events in news articles and revealing research trends in scientific literature.

2.4 Clustering

2.4.1 Clustering [6, 21] is the classification of objects into different groups, or more precisely, the partitioning of a data set into subsets (clusters), so that the data in each subset (ideally) share some common trait - often proximity according to some defined distance measure. A cluster is a collection of data objects that are similar to one another within the same cluster and are dissimilar to the objects in other clusters.

2.4.2 Distance Measure

An important step in any clustering is to select a distance measure, which will determine how the similarity of two elements is calculated. This will influence the shape of the clusters, as some elements may be close to one another according to one distance and further away according to another. For example, in a 2-dimensional space, the distance between the point $(x=1, y=0)$ and the origin $(x=0, y=0)$ is always 1 according to the usual norms, but the distance between the point $(x=1, y=1)$ and the origin can be 2.428 or 1 if you take respectively the 1-norm, 2-norm or infinity-norm distance.

Common distance functions:

- Euclidean distance (or the squared Euclidean distance).
- Manhattan distance (also called taxicab norm or 1-norm)
- maximum norm
- Mahalanobis distance

1. **Euclidean Distance:** In mathematics, the Euclidean distance or Squared Euclidean distance is the "ordinary" distance between two points that one would measure with a ruler, which can be proven by repeated application of the Pythagorean theorem. Euclidean Distance is the most common use of distance. In most cases when people said about distance, they will refer to Euclidean distance. Euclidean distance or simply 'distance' examines the root of square differences between coordinates of a pair of objects.

One-dimensional distance

For two 1D points, $P=(p_x)$ and $Q=(q_x)$, the distance is computed as:

$$\sqrt{(p_x - q_x)^2} = |p_x - q_x|$$

The absolute value signs are used, since distance is normally considered to be an unsigned scalar value.

Two-dimensional distance

For two 2D points, $P=(p_x, p_y)$ and $Q = (q_x, q_y)$, the distance is computed as:

$$\sqrt{(p_x - q_x)^2 + (p_y - q_y)^2}$$

Alternatively, expressed in circular coordinates (also known as polar coordinates), using $P = (r_1, \Theta_1)$ and $Q = (r_2, \Theta_2)$ the distance can be computed as:

$$\sqrt{r_1^2 + r_2^2 - r_1 r_2 \cos(\Theta_1 - \Theta_2)}$$

N-dimensional Distance:

The Euclidean distance between two points $P = (p_1, p_2, p_3, \dots, p_n)$ and $Q = (q_1, q_2, q_3, \dots, q_n)$, in Euclidean n-space, is defined as

$$\sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \dots + (p_n - q_n)^2} = \sqrt{\sum_{i=1}^n (p_i - q_i)^2}$$

In text mining each document is represented in n-dimensional point and the Euclidean distance between the points will be measured by using the above formula.

2. Manhattan distance (or taxicab distance): The taxicab distance between two points in a Euclidean space with fixed Cartesian coordinate system is the sum of the lengths of the projections of the line segment between the points onto the coordinate axes. For example, in the plane, the taxicab distance between the point P_1 with coordinates (x_1, y_1) and the point P_2 at (x_2, y_2) is $|x_1 - x_2| + |y_1 - y_2|$.

Taxicab distance depends on the rotation of the coordinate system, but does not depend on its reflection about a coordinate axis or its translation.

3. Mahalanobis distance: Mahalanobis distance is a distance measure introduced by P. C. Mahalanobis in 1936. It is based on correlations between variables by which different patterns can be identified and analyzed. It is a useful way of determining similarity of an unknown sample set to a known one. It differs from Euclidean distance in that it takes into account the correlations of the data set and is scale-invariant, i.e. not dependent on the scale of measurements.

Formally, the Mahalanobis distance from a group of values with mean $\mu = (\mu_1, \mu_2, \mu_3, \dots, \mu_p)^T$ and covariance matrix Σ for the multivariate vector $x = (x_1, x_2, x_3, \dots, x_p)^T$ is defined as :

$$D_M(x) = \sqrt{(x - \mu)^T \Sigma^{-1} (x - \mu)}$$

2.4.3 Clustering Methods

There exist a large number of clustering algorithms in the literature. The choice of clustering algorithm depends both on the type of data available and on the particular purpose and application.

In general, major clustering methods can be classified into the following categories.

1. Partitioning methods.
2. hierarchical methods
3. Density-based methods
4. Grid-based methods
5. Model-based methods.

1. Partitioning Methods: Given a database of n objects or data tuples, a partitioning method constructs 'K' partitions of the data , where each partition represents a cluster and $K \leq n$. That is , it classifies the data into K groups , which together satisfy the following requirements:

- (1) Each group must contain at least one object
- (2) Each object must belong to exactly one group.

Given K , the number of partitions to construct , a partitioning method creates an initial partitioning . It then uses an iterative relocation technique that attempts to improve the partitioning by moving the objects from one group to another. The general criterion of a good partitioning is that objects in the same cluster are "close" or related to each other, whereas objects of different clusters are "far apart" or very different. Most applications adopt one of two popular heuristic methods:

1 .K-means algorithm: where each cluster is represented by the mean value of the objects in the cluster.

2. K-medoids algorithm: where each cluster is represented by one of the objects located near the center of the cluster.

These heuristic methods work well for finding spherical-shaped clusters in small to medium-sized databases. To find clusters with complex shapes and for clustering very large data sets, partitioning-based methods need to be extended.

2. Hierarchical methods:

A hierarchical clustering method [15] works by grouping data objects into a tree of clusters. Hierarchical clustering methods can be further classified into agglomerative and divisive hierarchical clustering, depending on whether the hierarchical decomposition is formed in a bottom-up or to-down fashion.

1. agglomerative hierarchical clustering: This bottom-up strategy starts by placing each object in its own cluster and then merges these atomic clusters into larger and larger clusters, until all of the objects are in a single cluster or until certain termination conditions are satisfied. Most hierarchical clustering methods belong to this category. They differ only in their definition of inter cluster similarity. A detailed description of algorithm can be found in [28, 29].

2.Divisive hierarchical clustering: This top-down strategy does the reverse of agglomerative hierarchical clustering by starting with all objects in one cluster. It subdivides the cluster into smaller and smaller pieces, until each object forms a cluster on its own or until it satisfies certain termination conditions, such as a desired number of clusters is obtained or the distance between the two closest clusters is above a certain threshold distance. Figure 2.1 shows the sequence of steps in both divisive and agglomerative hierarchical clustering.

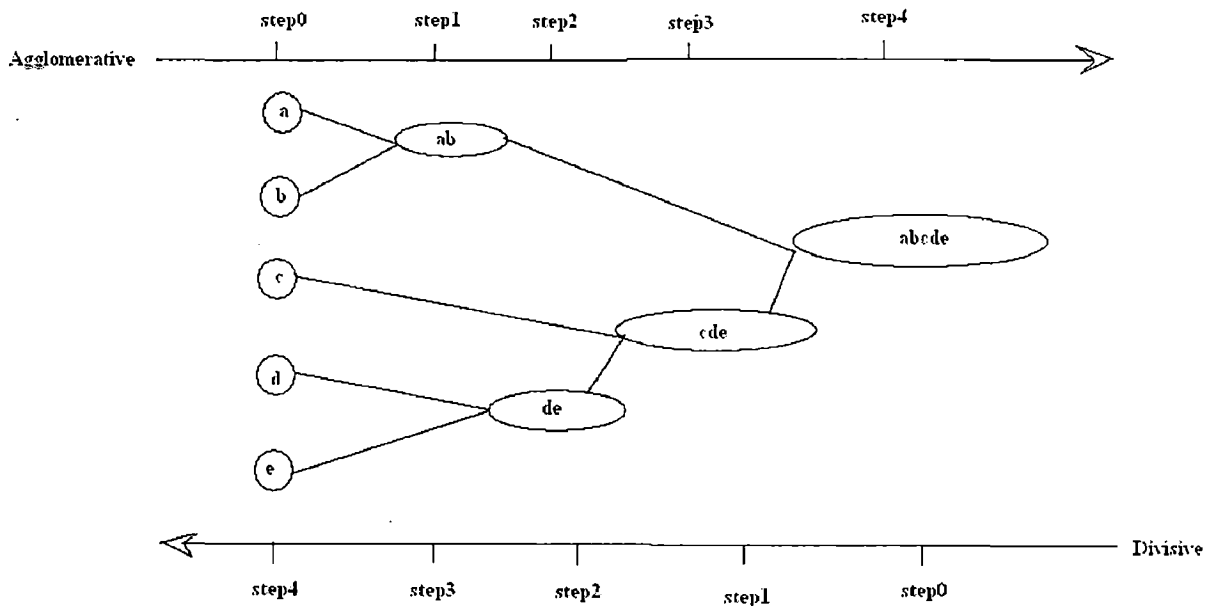


Figure 2.1: Agglomerative and divisive hierarchical clustering on data objects {a,b,c,d,e}

Hierarchical methods suffer from the fact that once a step (merge or split) is done, it can never be undone. This rigidity is useful in that it leads to smaller computation costs by not worrying about a combinatorial number of different choices

Algorithm:

Given a set of N items to be clustered, and an $N \times N$ distance (or similarity) matrix, the basic process of Johnson's (1967) hierarchical clustering is this:

1. Start by assigning each item to its own cluster, so that if you have N items, you now have N clusters, each containing just one item. Let the distances (similarities) between the clusters equal the distances (similarities) between the items they contain.
2. Find the closest (most similar) pair of clusters and merge them into a single cluster, so that now you have one less cluster.
3. Compute distances (similarities) between the new cluster and each of the old clusters.
4. Repeat steps 2 and 3 until all items are clustered into a single cluster of size N .

Step 3 can be done in different ways, which is what distinguishes single-link from complete-link and average-link clustering. In single-link clustering (also called the

connectedness or minimum method), we consider the distance between one cluster and another cluster to be equal to the shortest distance from any member of one cluster to any member of the other cluster. If the data consist of similarities, we consider the similarity between one cluster and another cluster to be equal to the greatest similarity from any member of one cluster to any member of the other cluster. In complete-link clustering (also called the diameter or maximum method), we consider the distance between one cluster and another cluster to be equal to the longest distance from any member of one cluster to any member of the other cluster. In average-link clustering, we consider the distance between one cluster and another cluster to be equal to the average distance from any member of one cluster to any member of the other cluster

3. Density based methods: Most partitioning methods cluster objects based on the distance between objects. Such methods can find only spherical-shaped clusters and encounter difficulty at discovering clusters of arbitrary shapes. Other clustering methods have been developed based on the notions of density. Their general idea is to continue growing the given cluster as long as the density (number of objects or data points) in the neighborhood exceeds some threshold; that is, for each data point within a given cluster, The neighborhood of a given radius has to contain at least a minimum number of points. Such a method can be used to filter out noise (outliers) and discover clusters of arbitrary shape. To discover clusters with arbitrary shape, density –based clustering methods have been developed. These typically regard clusters as dense regions of objects in the data space that are separated by regions of low density (representing noise). DBSCAN and OPTICS are the two well known density based algorithms. Detailed discussion about these methods can be found in [23, 24]

DBSCAN (Density based spatial clustering of applications with noise) [23] :is a density-based clustering algorithm. The algorithm grows regions with sufficiently high density into clusters and discovers clusters of arbitrary shape in spatial data-bases with noise. It defines a cluster as a maximal set of density-connected points.

OPTICS (Ordering Points To Identify the Clustering Structure): Although DBSCAN can cluster objects given input parameters, it still leaves the user with the responsibility of selecting parameter values that will lead to the discovery of acceptable clusters. Such parameter settings are usually empirically set and difficult to determine, especially for real-world , high-dimensional data sets. Most algorithms are more sensitive to such parameter values. To help overcome this difficulty , a cluster analysis method called OPTICS was proposed. Rather than produce a data set clustering explicitly , OPTICS computes an augmented cluster ordering for automatic and interactive cluster analysis. This ordering represents the density-based clustering obtained from a wide range of parameter settings.

4. Grid-based methods: Grid-based methods quantize the object space into a finite number of cells that form a grid structure. All of the clustering operations are performed on grid structure (i.e., on the quantized space). The main advantage of this approach is its fast processing time , which is typically independent of the number of data objects and dependent only on the number of cells in each dimension in the quantized space. STING – a typical grid-based clustering algorithm .

5. Model-based methods: Model-based methods [14] hypothesize a model for each of the clusters and find the best fit of the data to the given model. A model-based algorithm may locate clusters by constructing a density function that reflects the spatial distribution of the data points. It also leads to a way of automatically determining the number of clusters based standard statistics , taking “noise” or outliers into account and thus yielding robust clustering methods.

3.1 Data set used

Before presenting the methods addressing the three questions stated in the problem statement, the dataset (collection of text articles) we use must satisfy some of the properties like

1. the text of the each documents is accessible,
2. the documents have meaningful timestamps and these timestamps and authors names are accessible.

Examples of such collections are email, proceedings of scientific conferences, scientific journals, news, and blogs. As a testbed, we chose a collection of scientific articles, in particular the articles published in the proceedings of the Neural Information Processing Systems (NIPS) conference between 1987 and 2000. The reason for choosing this data set is. First, we believe that scientific document collections fulfill the assumptions stated above. Second, for scientific articles, citation data is available and we can compare our methods against citation counts. . There are a total of 1955 documents. We use only the text (not the citation or bibliographic information) from these documents. As meta-data, we use only the time-stamps of the documents and the author names of each article.

We have also used one more dataset . It consists of 357 news articles collected from various news sources like BBC, Times of India, and CNN. The data set consists of documents related to various topics like Microsoft news, Tsunami, Indian cricket, Bombay blasts and kargil war. In general the dependency between these news articles is very less when compared to the dependency between the NIPS documents.

A well known method to represent the dataset is described below.

3.2 Vector representation of a document: The initial step of the before applying the proposed methodology is representation of a document and the data set. In particular, we convert the text documents to a standard TFIDF representation [17, 27]. In this representation, the features are words from the available text. After removing stop words and words that only occur once. To build a TFIDF vector for each document, we count the number of times term 't' appeared in the document. Then we multiply by the IDF weighting factor of $\frac{1}{n \log(nt)}$, where n is the number of documents in the corpus and 'nt' is the number of documents that contain the term t_i . To determine the similarity of two documents, we use the standard cosine similarity between the TFIDF vectors.

3.3 Term weighting using TFIDF:

Usually we want to say that some terms are more important than some other terms. We can express this by weighting terms of a vector. Very often, the standard TFIDF function is used:

$$tfidf(t_k, d_j) = \#(t_k, d_j) * \log(|Tr| / \#Tr(t_k)),$$

in which

$\#(t_k, d_j)$ denotes the number of times term t_k occurs in document d_j

$\#Tr(t_k)$ denotes the number of documents in Tr in which t_k occurs

$|Tr|$ denotes the number of documents .

For instance, in the Doc1, the term 'application' occurs once, and in the collection it occurs 2 times (no.of documents 10):

$$Tfidf(\text{application}, \text{Doc1}) = 1 * \log(10 / 2) = \log 5 \sim 0.70$$

$$tfidf(\text{current}, \text{Doc1}) = 1 * \log(10 / 7) = \log 1.4 \sim 0.15$$

Now we have features (words) and their weights. We construct a $m * n$ matrix where 'm' is the number of documents and 'n' is the total (sum of non repeated features in all the

documents) number of features. An element A_{ij} in matrix is the tfidf value of feature 'j' in document 'i'. $A_{ij} = 0$ if the document i doesn't contain the term 'j'. An example vector representation of dataset is shown in below figure3.

Figure 3: An example vector representation of data set:

```
0,0,0,0,0,0,0,0,0,4.49,4.49,0,0,0,0,4.49,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
1,0,0,0,0,0,0,0,0,0,4.49,0,0,0,0,7.61,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,10.2,
2,0,0,0,0,0,0,0,0,7.61,4.49,0,0,0,4.49,7.61,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
3,0,0,0,0,0,0,0,0,0,4.49,0,0,0,0,4.49,0,0,0,0,0,0,0,0,0,0,0,4.49,0,0,4.49,0
4,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,4.4
5,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,4.4
6,0,0,0,0,0,0,0,0,0,10.2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
7,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,4.49,0,0,0,0,0,0,0,0,0,0,0,0,0,0,4.49,0,0
8,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,4.49,0,0,0,0,0,0,0,4.49,0,0,0,0,0,0,0,0,0,0
9,0,0,0,0,7.61,7.61,0,0,0,0,0,0,0,0,0,10.2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
10,4.49,7.61,0,0,0,0,0,0,0,4.49,0,0,0,0,10.2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
11,4.49,4.49,0,0,0,0,0,0,0,0,0,0,0,0,0,0,25.08,4.49,4.49,0,4.49,0,0,0,0,0,0,0,
12,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,4.49,4.49,0,0,4.49,0,0,
13,0,0,0,0,4.49,4.49,7.61,0,0,0,0,0,0,0,7.61,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
14,0,0,0,0,4.49,7.61,4.49,0,0,0,0,0,0,0,10.2,4.49,0,0,4.49,0,0,0,0,10.2
15,0,0,0,10.2,0,10.2,0,0,7.61,0,7.61,4.49,0,4.49,0,0,0,14.45,0,0,0,0,7.61
16,4.49,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,4.49,0,0,
17,10.2,0,0,0,0,0,0,0,0,0,4.49,0,0,0,0,4.49,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1
18,0,4.49,0,0,0,0,0,0,7.61,0,0,4.49,0,0,0,0,7.61,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
18,0,4.49,0,0,0,0,0,0,4.49,4.49,0,0,0,0,0,0,0,0,0,0,0,0,0,0,4.49,0,0,0,0,0,
```

The above matrix is of $18 * 30$ size, that means it represents 18 documents and total number of features 30 in one matrix. Each row represents a document and each column represents a one of the features. A zero in matrix represents the document doesn't contain the feature in that particular column. The first column in the matrix represents document number.

Sections 3.4 and 3.5 and 3.6 discusses the proposed methodologies to address the three questions stated in problem statement in section 1.4.

3.4 How do key topics change over time?

The first problem we consider is that identifying and visualizing the key topics of a document collection i.e we are trying to discover all the topics in our dataset, and how the popularity of these topics develops over time i.e if a train accident is happened today then the popularity of this topic is more today, there is no popularity for this topic yesterday and the popularity of this topic decreases when time goes on..

3.4.1. Method

Our method towards solving the first problem proceeds in three steps.

Step1: In the first step, we determine the key topics in the document collection via clustering. Each cluster represents a key topic.

Step2: In the second step, we describe the topics which are identified in step1.

Step3: In the final step, we visualize the temporal behavior of topics as a flow through time indicating increasing or decreasing popularity.

As the clustering algorithm in the first step we use one of the well known portioning methods, the k-means clustering algorithm

3.4.2 K-means clustering Algorithm:

The k-means algorithm [7, 8, 10, 16, 25] is an algorithm to cluster objects based on attributes into k partitions/clusters. It assumes that the object attributes form a vector space. The objective it tries to achieve is to minimize total intra-cluster mean, or, the squared error function

$$V = \sum_{i=1}^k \sum_{x_j \in S_i} |x_j - \mu_i|^2$$

Where there are k clusters S_i , $i=1,2,3, \dots, k$. and μ_i is the mean point or centroid of all points $x_j \in S_i$.

The algorithm starts by partitioning the input points into k initial sets, either at random or using some heuristic data. It then calculates the mean point, or centroid, of each set. It constructs a new partition by associating each point with the closest centroid. Then the

centroids are recalculated for the new clusters, and algorithm repeated by alternate application of these two steps until convergence, which is obtained when the points no longer switch clusters (or alternatively centroids are no longer changed). The algorithm has remained extremely popular because it converges extremely quickly in practice. In fact, many have observed that the number of iterations is typically much less than the number of points.

Algorithm:

For given N objects (text documents in our case) the K-means algorithm works as follows:

Step 1. Begin with a decision on the value of $k = \text{number of clusters}$ ($N \geq k$)

Step 2. Put any initial partition that classifies the data into k clusters. You may assign the training samples randomly, or systematically as the following:

Take the first k training sample as single-element clusters

1. Assign each of the remaining $(N-k)$ training sample to the cluster with the nearest centroid.
2. After each assignment, recomputed the centroid of the gaining cluster.

Step 3 . Take each sample in sequence and compute its distance from the centroid of each of the clusters. If a sample is not currently in the cluster with the closest centroid, switch this sample to that cluster and update the centroid of the cluster gaining the new sample and the cluster losing the sample.

Step 4 . Repeat step 3 until convergence is achieved, that is until a pass through the training sample causes no new assignments.

If the number of data is less than the number of cluster then we assign each data as the centroid of the cluster. Each centroid will have a cluster number. If the number of data is bigger than the number of cluster, for each data, we calculate the distance to all centroid and get the minimum distance. This data is said belong to the cluster that has minimum distance from this data.

Since we are not sure about the location of the centroid, we need to adjust the centroid location based on the current updated data. Then we assign all the data to this new centroid. This process is repeated until no data is moving to another cluster anymore. Mathematically this loop can be proved to be convergent. The convergence will always occur if the following condition satisfied:

1. Each switch in step 2 the sum of distances from each training sample to that training sample's group centroid is decreased.
2. There are only finitely many partitions of the training examples into k clusters.

Now, we have K clusters each cluster representing a topic, i.e there are K topics in our data set. Now we know there are K topics in our dataset but what are those topics, So, our next challenge is how to describe these clusters. To describe each cluster's topic, we extract the five words with the highest weights in the cluster's centroid document. These five words are the most important terms in defining the topic which the cluster is representing. The number five is somewhat arbitrary, but was chosen because we found that five words are sufficient to convey a good sense of the cluster's content.

Finally, we need to plot a graph which shows how the topic popularity varies over time. For that on x-axis we have taken time and have divided that axis into equal time intervals. For each time interval, we compute the number of documents that fall into each cluster in that time interval and plot a graph for each cluster. For example if our data set contains documents having timestamps in the range jan-2000 to dec-2000 then our x-axis is divided into 12 equal intervals each interval represents a month in year 2000.

Now, all we have to do is for each cluster draw a curve such that each point on that curve is (x,y) , where x -represents the month in year 2000 and y is the total number of documents in that cluster published in that month, i.e a point on a cluster curve $(2,15)$ conveys the information that this topic has 15 documents in feb-2000.

3.2 Which are the most influential documents?

After visualizing clusters and determining how the topics developed over time, there is need to identify the most influential documents. Unlike traditional methods we used only text of the document to find the influential documents. Our goal is to provide a general solution that will work for any text document.

Method

Before presenting the method let us define what the word “influence” means. We define the impact/influence of a document as the amount of followup work it generates. If there are two documents in a dataset which are sharing many words/features (i.e using similar vocabulary) and the document1’s publication date is earlier than document2 then we can say that the document1 influenced/leads document2 and document2 is depended/followed on document1.

The general idea is illustrated in Figure 4. In this figure, the x-axis represents time and the y-axis documents. The circles in the graph represent documents, where the open circles are of one topic and the black dots are of another topic. The open circles (and black dots) towards the left of their respective clusters are leaders because they are published earlier than the documents which shares the same vocabulary. On the other hand, the circles (and dots) towards the right are followers because documents with similar content precede these documents.

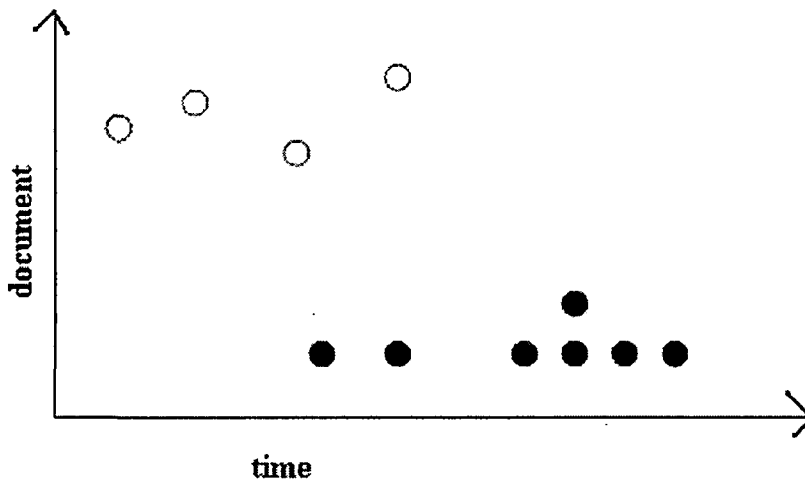


Figure 4 Which are the most influential documents

We propose a measure named lead/lag index to measure whether a document is more of a leader or more of a follower based on the text content or features they are sharing and comparing their timestamps. We assume that leaders have many papers following them, and vice versa.

More formally, the index is defined as follows. For each document d , we find the k nearest neighbors $knn(d)$ in terms of the Euclidean distance between TFIDF vectors. We then count the number of neighbors that are published later than d

$$klater = |\{d' \in knn(d) \wedge (time(d') > time(d))\}|$$

and the number of papers that precede the paper

$$kearlier = |\{d' \in knn(d) \wedge (time(d') < time(d))\}|.$$

For a document d the $klater$ measure will be the number of documents that are nearer to that document and having the timestamp more than this document. By using the distance measures discussed in section 2.3.2 we can calculate the distance between two documents. For a document ' d ', all we have to do is find the first ' k ' nearest documents to that document and find how many documents are published later than this document, and how many documents are published earlier than this document. The number of documents published later will become $klater$ and the number of documents published earlier will become $kearlier$. We have to compute the $klater$ and $kearlier$ measures to all the documents in the dataset. By comparing these two numbers, it is possible to determine whether the document is more of a leader or follower. If the $klater$ value is more than $kearlier$ then we can say that the document is more of a leader than follower. The document which is having the highest $klater$ value is proposing new ideas. But by seeing only the $klater$ value we can't say that the document is introducing new ideas, because for that document if the $kearlier$ measure is almost equal to the $klater$ then this indicates that the document has taken many ideas from earlier documents so our measure for calculating the influence of the document must take both $klater$ and $kearlier$ into

account. We are proposing one such measure called ‘raw lead/lag index’ of a document d is computed by subtracting the number klater of papers following the current paper in time from the number kearlier of papers preceding the current paper in time.

$$I_{\text{raw}}^d = \text{klater} - \text{kearlier}$$

However, the index is strongly affected by edge effects. For example, kearlier is guaranteed to be zero for documents from the first timestamp. To avoid such biases, we scale each year’s documents by normalizing [30] it across all papers from the same time stamp. In particular, we subtract the average of the raw lead/lag indices for a year from each raw lead/lag index in that year.

$$I_{\text{scaled}}^d = \frac{1}{k} \left(I_{\text{raw}}^d - \frac{|\{d_i : \text{time}(d_i) = \text{time}(d_0)\}|}{\sum_{\{d_i : \text{time}(d_i) = \text{time}(d)\}} I_{\text{raw}}^{d_i}} \right)$$

The resulting scaled lead/lag index corrects for such edge effects. The higher the scaled lead/lag index, the more influential the paper. Since the scaled lead/lag index is also normalized with respect to k , the scaled lead/lag index scores typically fall in the interval from -1 to +1, with extremely strong papers receiving scores slightly above +1 and extremely lagging papers receiving scores slightly below -1.

3.3. Who are the most influential authors?

Since papers do not write themselves, once we can determine the most leading documents, the next logical step is to ask who wrote them. Given a collection of documents, we would like to answer the questions of which authors produce the most original work, which authors are most influential in spreading their ideas.

Method

The document lead/lag index already provides a method for determining the influence of a document. To identify the most influential authors in the document collection, we can aggregate the document lead/lag information by author. More specifically, consider an author with n papers receiving scaled lead/lag scores $L^{d1}_{scaled}, \dots, L^{dn}_{scaled}$. One simple method to find rank of the author is calculating the average of lead/lag scores. For more accurate results we are using the formula proposed in [30].

$$\text{Rank} = m + 2 * \frac{\sqrt{v}}{n}$$

Where 'm' is the mean calculated for the lead/lag scores the author received for his articles, v is the variance, and n is the number of documents.

4.1 How do the topics change over time:(Results)

We applied our proposed method on two data sets news articles, and NIPS data set. Our data set1 contains 357 news articles when each document relates to one of five different topics, Microsoft news, Bombay blasts, tsunami, kargil war and Indian cricket. Dataset2 contains 1955 research papers. In particular we have used K-means clustering algorithm for clustering the documents. Figure 4.1 shows the results of the method as applied to the dataset1 for k=3.

Figure 4.1: k-means clusters(k=3) for dataset1

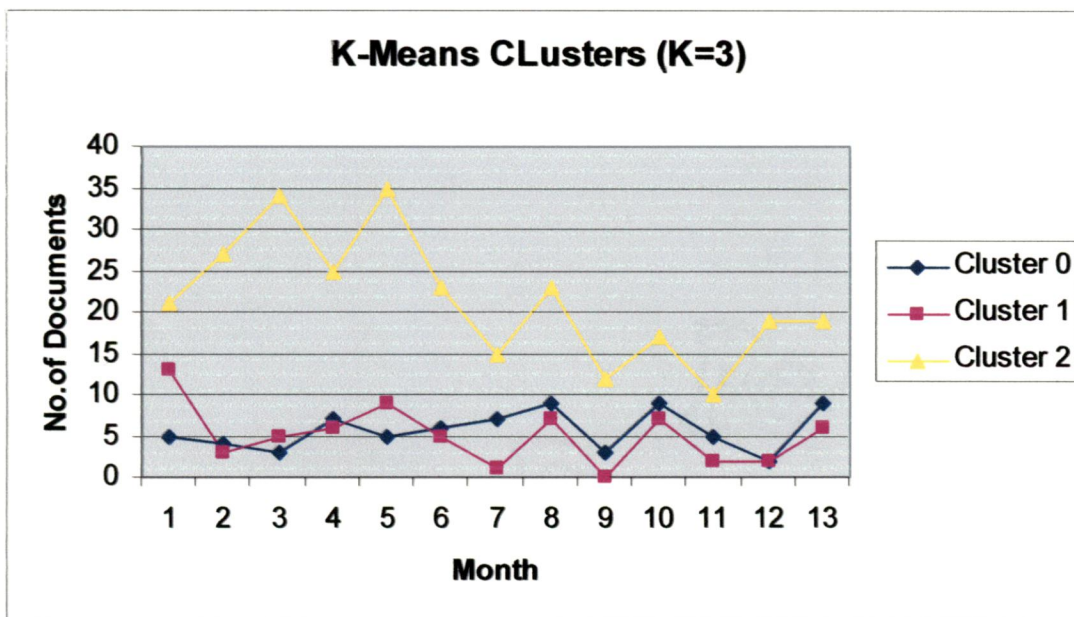


Figure 4.2: description of each cluster

Cluster descriptions:				
Cluster 0:	week	Microsoft	software	chief news
Cluster 1:	week	earlier	coach	news team
Cluster 2:	world	disaster	nagapatnam	villages research

Figure 4.2 shows the description of each cluster after extracting five high weighted features from the centre document of each cluster. Our method of extracting keywords from the cluster centroids works reasonably well – many of the words are highly informative for the cluster content. The top five words shown give a reasonable description of the main topics in the given dataset

Total number of documents in cluster0 are 74 , out of those 73 are belongs to the topic Microsoft and remaining document belongs to kargil war topic. The extracted five words (week, Microsoft, software, chief, news) are good enough to describe the topic related to Microsoft. Total number of documents in cluster1 are 66 , out of those 65 are belongs to the topic Indian cricket and remaining document belongs to Bombay blasts topic. The extracted five words (week, earlier, coach , news , team) are good enough to describe the topic related to Indian cricket. Most of the documents in this cluster discusses about the coach selection of Indian cricket team, So here the two terms ‘coach’ and ‘team’ are good enough to describe the topic. Total number of documents in cluster2 are 280 , out of those 50 documents belongs to the topic kargil war and 6 documents to Indian cricket and 90 belongs to tsunami and 73 to Bombay blasts topic. The extracted five words (world, disaster ,nagapatnam , villages , research) are good enough to describe the topic related to tsunami.

Figure 4.3 and 4.4 shows the resulted graph and cluster descriptions for k=4. Total number of documents in cluster0 are 65 , all of them belongs to the topic Microsoft topic. The difference between cluster0 for k=3 and for k=4 is , number of documents related to the topic changed so as the center of the cluster , and the five words which describe the cluster. The extracted five words (week, Microsoft, software, chief, news) are good enough to describe the topic related to Microsoft. The number of documents in cluster1 are 43 , all the documents belongs to the topic Indian cricket . Even though the number of documents in this cluster have been reduced from 63 to 43 by increasing the k value , the center of the cluster so as the description of the has not changed. The extracted

Figure 4.3 shows the results of the method as applied to the dataset1 for k=4

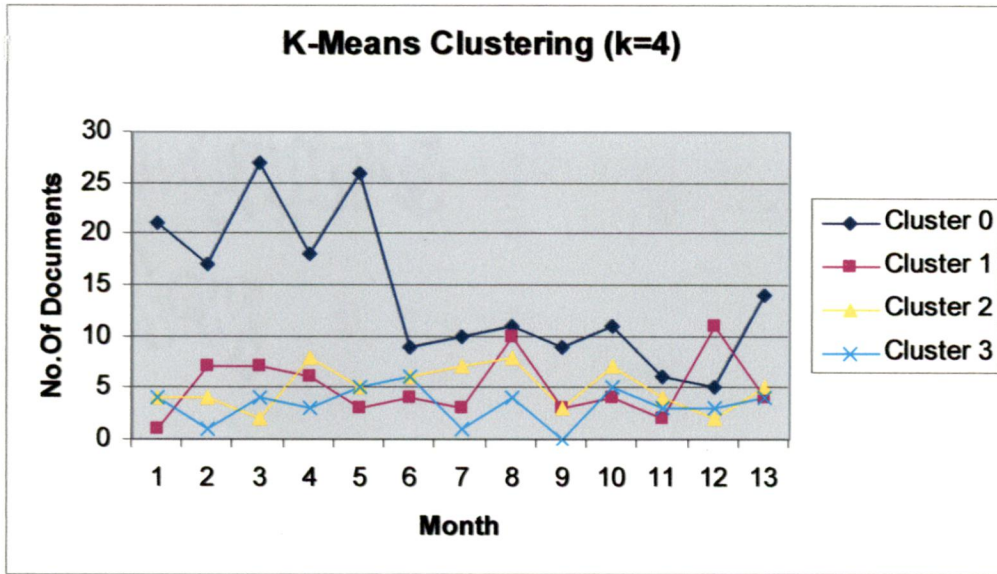


Figure 4.4: description of each cluster (for k=4)

Cluster descriptions:

Cluster 0:	based	Microsoft	days	data	software
Cluster 1:	week	earlier	coach	news	team
Cluster 2:	dead	villages	nagapatnam	similar	research
Cluster 3:	chief	district	city	dead	blasts

five words (week, earlier, coach, news, team) are good enough to describe the topic related to Indian cricket. Total number of clusters in cluster2 are 184, out of those 50 documents belongs to the topic kargil war and 17 documents to Indian cricket and 90 belongs to tsunami and 27 to Bombay blasts topic. The extracted five words (world, disaster, nagapatnam, villages, research) are good enough to describe the topic related to tsunami.

Figure 4.5 and 4.6 shows the results of the method as applied to the dataset1 for k=5

Total number of documents in cluster0 are 65 , all of them belongs to the topic Microsoft topic. The extracted five words (week, Microsoft, software, chief, news) are good enough to describe the topic related to Microsoft. The number of documents in cluster1 are 43, all the documents belongs to the topic Indian cricket .. The extracted five words (week, earlier, coach, news, team) are good enough to describe the topic related to Indian cricket. Total number of clusters in cluster2 are 100 , out of those 51 documents belongs

Figure 4.5 shows the results of the method as applied to the dataset1 for k=5

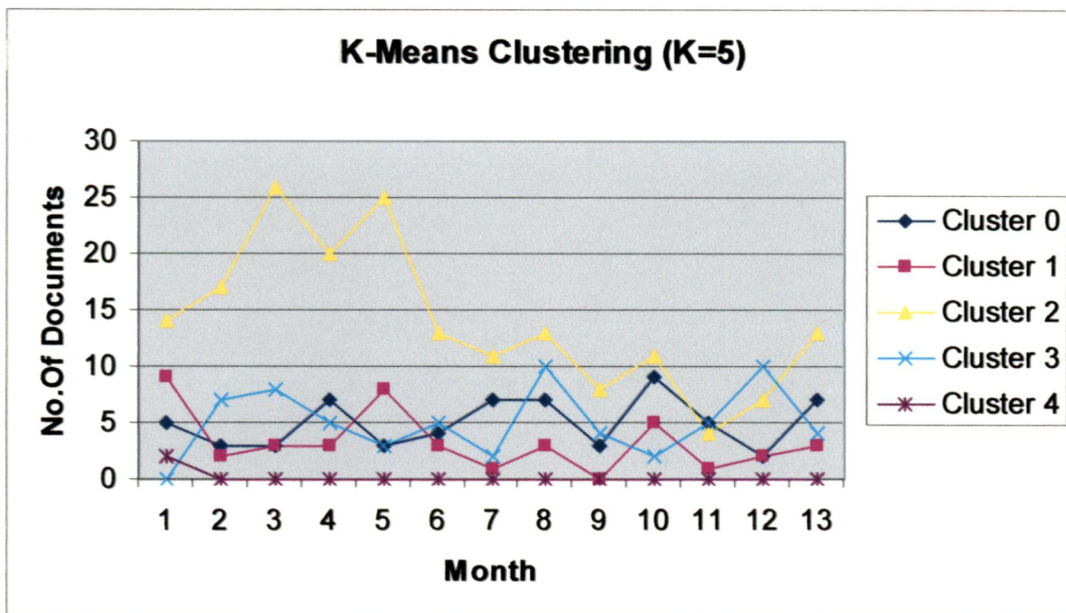


Figure 4.6 : cluster descriptions of clusters in figure 4.5

Cluster Descriptions:				
Cluster 0:	based	Microsoft	days	data software
Cluster 1:	week	earlier	coach	news team
Cluster 2:	dead	die	war	India kargil
Cluster 3:	chief	district	city	dead blasts
Cluster 4:	world	aid	million	villages coastal

to the topic kargil war and 28 documents to Indian cricket and 20 belongs to tsunami and 11 documents belongs to Bombay blasts topic. The extracted five words (dead, die

war, India, kargil) are good enough to describe the topic related to kargil war. Till now we haven't got a document belongs to kargil war topic in the center of any cluster, but in cluster 3 (for k=5) the extracted 5 words are form the document which belongs to kargil war. Total number of documents in cluster4 are 74 , all of them belongs to tsunami topic The extracted 5 words (world , aid , million, , villages, coastal) are good enough to explain the cluster topic tsunami.

Now, we have got all the topics as clusters , For k=3 and k=4 we are not able to recognize all the topics in the given dataset. But for k=5 we found all the topics .So, our proposed method for describing the clusters is giving good results if the documents are well clustered.

As explained in section 3, our dataset2 contains 1995 documents from NIPS conference . Figure 4.7 shows resultant graph after application of our method to find the trends in topics on dataset2. Figure 4.8 and shows the cluster descriptions for k=7. Figure 4.9 shows the tabular description of total number of documents in each year, for each cluster.

Figure 4.7 shows how our method finds the trends in the topics detected by using the clusters on NIPS data set.

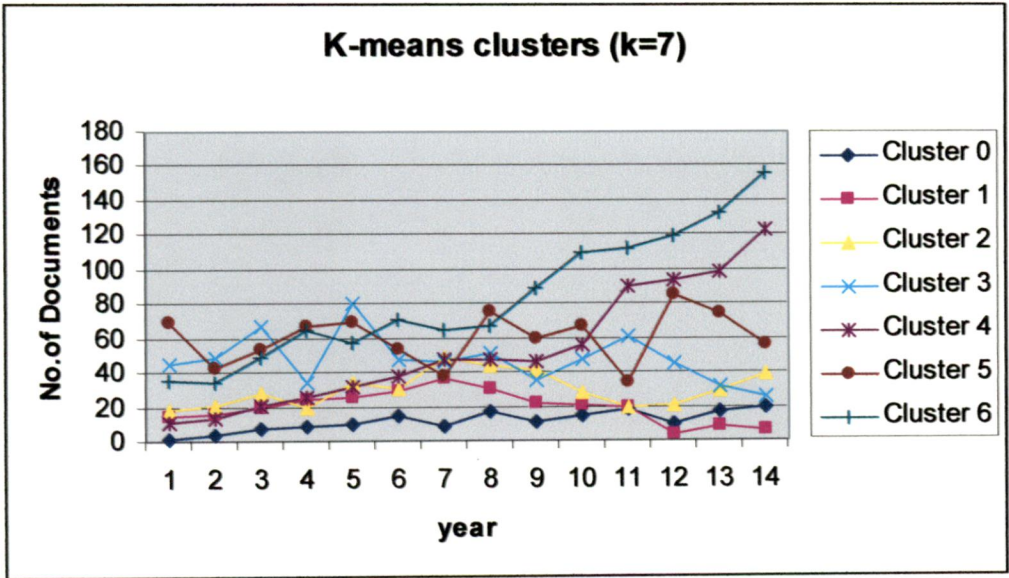


Figure 4.8 : cluster descriptions of figure 4.7.

Cluster descriptions:

Cluster 0: units , node, training, nodes, tree
 Cluster 1: Spike cells neurons cell firing
 Cluster 2: Image images object objects recognition
 Cluster 3: policy reinforcement state controller action
 Cluster 4: bayesian, mixture, posterior, likelihood, em
 Cluster 5: speech, word, hmm, recognition, mlp
 Cluster 6:kernel, margin, svm, vc, xi

Figure 4.9. Figure shows number of documents published in each cluster

Year/cluster	1987	1989	1990	1992	1993	1995	1997	1999	2000
Cluster 0	1	7	9	14	9	11	18	17	123
Cluster 1	15	20	24	29	37	22	19	8	156
Cluster 2	18	28	19	31	49	41	19	29	123
Cluster 3	45	67	34	48	46	35	61	32	156
Cluster 4	11	21	26	38	47	46	90	98	123
Cluster 5	69	54	67	54	38	59	34	74	156
Cluster 6	35	49	64	70	64	89	112	132	156

After having a glance at a graph shown above one can easily find the trends in given dataset. The cluster4 and cluster6 are having 11 and 35 research papers in year 1987 but in year 2000 they are having 123 and 156 documents respectively, that clearly shows that cluster4 and cluster 6 are the active research areas in year 2000.. The cluster1 and cluster2 are started slowly in year 1987 but almost disappeared in year 2000.

Overall, we believe that our proposed method for finding trends and visualization and describing the clusters is producing good results.

4.2 which are the most influential documents?

We computed the i_d^{raw} and the scaled lead lag index for our two datasets. The value of k is the only parameter that needs to be selected. With a small k , only the closest documents to a particular document are considered. If a paper is very influential, then other documents influenced by that paper are missed in this analysis. On the other end of the spectrum, if k is too large, documents that are only marginally affected by a particular paper are included in the ranking. This can lead to noisier results. We run the experiments for different values of k .

Figure 4.10. Based on the lead/lag index, below is a list of the most influential NIPS papers when considering the paper's $K=14$ nearest neighbors.

Rank	Year	Citations	Paper Title
1.156	1997	191	“improving the accuracy and speed of Support vector machines”
1.064	1999	44	“Using analytic qp and sparseness to speed training of support vector machines”
0.997	1999	34	“approximate learning of dynamic models”
0.975	1997	371	“support vector method for function approximation, regression estimation, and signal processing.”
0.953	1998	32	“training methods for adaptive boosting of neural networks”
0.950	1997	4	“modeling complex cells in an awake macaque during natural image viewing”
0.945	1996	27	“em optimization of latent-variable density models”
0.904	1996	756	“a new learning algorithm for blind signal separation”
0.896	1996	20	“fast learning by bounding likelihoods in sigmoid type belief networks”
0.876	1999	73	“dynamically adapting kernels in support vector machines”

Different values of k agreed more or less on which documents are most influential, with only small changes in the ordering among the top scoring papers. This indicates that the

method is robust with respect to the choice of k , and that most reasonable values of k produce comparable results.

Figure 4.11 Based on the lead/lag index , below is a list of the most influential NIPS papers when considering the paper's $K=24$ nearest neighbors.

Rank	Year	Citations	Paper Title
1.324	1996	756	“a new learning algorithm for blind signal separation”
1.246	1997	371	“support vector method for function approximation , regression estimation, and signal processing.
0.986	1997	191	“improving the accuracy and speed of support vector machines”
0.975	1992	293	“Second order derivatives for network pruning”
0.953	1998	32	“training methods for adaptive boosting of neural networks”
0.948	1999	73	“dynamically adapting kernels in support vector machines”
0.945	1992	56	“Induction of Multi scale Temporal Structure”
0.943	1999	44	“Using analytic qp and sparseness to speed training of support vector machines”
0.896	1996	20	“fast learning by bounding likelihoods in sigmoid type belief networks”
0.876	1997	4	“modeling complex cells in an awake macaque during natural image viewing”

The results from our method are different from what citation analysis would produce. The number of citations of each paper according to Google Scholar is given in the third column of Figure 4.10. Our method reflects the importance of ideas presented in the paper, not how often this paper was cited, that too for a document we are calculating the idscaled lead/lag index with in the dataset given , but the citation information takes all the papers into account. These results validate our assumption that measuring textual similarity provides an adequate method for determining which papers have influence on later papers in the document collection..

Figure 4.12 Based on the lead/lag index , below is a list of the most influential NIPS papers when considering the paper’s K=5 nearest neighbors on “Microsoft “ cluster .

Rank	Publication date	Title of article
0.96	12-01-05	Microsoft Releases 3 New Windows Security Patches
0.93	17-12-04	Microsoft says IE updates possible
0.91	21-12-04	Microsoft to step up SP2 downloads
0.533333333	07-02-05	Microsoft pitches new Visual Studio tools
0.4332233	02-09-05	Microsoft launches iTunes rival

As described in chapter 3 our dataset2 contains 74documents related microsoft news, figure 5 shows the resulted ranks , in addition to their publication dates and titles of those articles after applying our methodology to find the influential documents in given collection. As this data doesn’t contain any citation data , the citation data column is omitted in this table.

We need to experiment our data for different k values and analyze the results. If a document is ranked one for one k value, we can’t say that this document is the most influenced document , because the same document may get different ranks for different k values. However our method is identifying influential documents correctly with slight change in the ordering.

4.3 Who are the most influential authors?

For different 'k' values , we computed the authors lead/lag index for all authors in the two data sets. Figure 4.13 and 4.14 lists the authors ranks for NIPS dataset , and figure 4.15 lists the authors ranks for one of the news article clusters (Microsoft)

We computed the author lead/lag index for all authors in the NIPS collection. Figure 4.13 has the results for $k = 14$. Overall, we find that this ranking for the most part identifies a document collection's key players. From bibliometrics, we know that typically the best predictor of an author's importance is the number of citations that author receives. Therefore, we compare our ranks of the authors to the number of citations an author has received on Google Scholar. (We accumulated the citations of each document written by the author) . Additionally, we present number of papers the author has published in NIPS.

Figure 4.13: The below list contains the NIPS authors with the highest ranking in the author lead/lag index. From considering each paper's $K=14$ nearest neighbors for the document lead lag index and then aggregating the document lead/lad index by author.

Author Name(s)	Rank	No.of papers	Citations
Jordan, Michael	0.035	27	1334
Smola, Alex	0.004	13	982
Scholkopf	-0.004	10	389
Atkeson, Christopher g.	-0.022	10	762
Williams, Christopher k.i.	-0.067	16	1245
Sejnowski, Terrence j.	-0.069	46	2768
Hinton, Geofferey e.	-0.075	27	1875
Jaakkolla, Tommi	-0.091	10	1135
Miller, Kenneth d.	-0.106	11	678
Bengio, Yoshua	-0.112	21	345

Figure 4.14: The below list contains the NIPS authors with the highest ranking in the author lead/lag index. From considering each paper's K=24 nearest neighbors for the document lead lag index and then aggregating the document lead/lad index by author.

Author Name(s)	Rank	No.of papers	Citations
Sejnowski, Terrence j.	0.046	46	2768
Jordan, Michael	0.007	27	1334
Jaakkolla, Tommi	0.004	10	1135
Smola, Alex	-0.002	13	982
Hinton, Geofferey e.	-0.085	27	1875
Williams, Christopher k.i.	-0.076	16	1245
Saad, David	-0.075	11	765
Miller, Kenneth d.	-0.109	11	678
Bengio, Yoshua	-0.116	21	345
Tresp, Volker	-0.122	15	672

More over our method to find the influential authors is producing good results with slight changes in the order. By using this method we can confidently say that these are the top 'n' authors , or this author is one of the top 'n' authors, but we can't say the exact position of author. That means we can't say "Jordan" is the most influenced author by seeing the figure 4.13 or we can't say "Sejnowski" is the most influenced author by seeing the figure 4.14, but we can say both these authors are in top 6.

Figure 4.15: The below list contains the dataset1 ('Microsoft 'cluster) authors with the highest ranking in the author lead/lag index. From considering each paper's K=5 nearest neighbors

Author Name	Rank (Based on Lead/Lag index)	Number of Papers
Dawn kawamoto	-0.023284	9
David Becker	-0.037231	5
Alorie Gibert	0.021000	8
Martin Lamonica	-0.058967	5
Elinore Mills	-0.094680	6
Mike Ricciuti	-0.271440	7
Ian Fried	-0.001457	15
Paul Roberts	-0.156860	5

Conclusions:

We proposed the problem of discovering evolutionary theme patterns in given dataset for which there is no meaningful citation data available. As proof of concept, we propose simple methods that show that this problem is feasible and interesting. Unlike existing approaches from bibliometrics, the new methods are applicable even if no citation or hyperlink data is available. Using the proceedings of the NIPS conference as a testbed, the obtained results were found to give an accurate summary of the popularity of topics over time. To identify the papers with largest influence on topic development, we defined a document lead/lag index that is an effective indicator of the influence of a document. Finally, we extended the influence analysis to authors by aggregating document lead/lag indices. These lead/lag scores are the first measures able to identify key authors .

Future work:

We believe that discovering and visualizing evolutionary theme patterns is an exciting area that deserves future research. The methods presented in this thesis give evidence that such analyses are possible even without citation information. However, more principled approaches are likely to be even more accurate and could provide more meaningful insights.

REFERENCES

- [1] Sophia Ananiadou, Douglas B. Kell and Jun-ichi Tsujii “Text mining and its potential applications in systems biology” , International conference on Trends in Biotechnology, Vol 24, Issue 12, pp 571-579, Dec 2006.
- [2] Qiaozhu Mei, Chao Liu, Hang Su, ChengXiang Zhai “Data mining: A probabilistic approach to spatiotemporal theme pattern mining on weblogs”, Proceedings of the 15th international conference on World Wide Web WWW '06, Pages 17-26, may 2006.
- [3] R. Guha, D. Sivakumar, R. Kumar, and R. Sundaram. “Unweaving a Web of Documents.”, In Proceedings of KDD Chicago, -2005, Pages 112-123, Aug 2005.
- [4] Jiawei Han, Micheline Kamber , “ Data Mining Concepts and Techniques” , Text book, Elsevier Publisheers-2005.
- [5] Qiaozhu Mei, ChengXiang Zhai , “ Discovering evolutionary theme patterns from text: an exploration of temporal text mining” ,Proceeding of the eleventh ACM SIGKDD Pages 356-365 ,August 2005
- [6] Hsi-Cheng Chang; Chiun-Chieh Hsu , “Using topic keyword clusters for automatic document clustering”; Third international conference on information technology and application, Volume 1, Page(s):419 – 424 , July 2005 .
- [7]Min-Ling Zhang; Zhi-Hua Zhou, “A k-nearest neighbor based algorithm for multi-label classification”, IEEE International Conference on Granular Computing, 2005, vol 2 Pages 25-28, Jul2005.

- [8] Christian Böhm and Florian Krebs , “The k-Nearest Neighbour Join: Turbo Charging the KDD Process “, journal of Knowledge and Information Systems, volume 6 ,pp 101-119, November 2004
- [9] F. B. Viegas, M. Wattenberg, and K. Dave. “Studying Cooperation and Conflict between Authors with history flow Visualizations”. In Proceedings of CHI-2004, Vienna, Austria, Pages 67-76, April 2004..
- [10] Carlos Ordonez, “Clustering binary data streams with K-means”, Proceedings of ACM SIGMOD workshop, vol 3 , Pages 12-19, june 2003.
- [11] A. Kontostathis, L. Galitsky, W. M. Pottenger, S. Roy, and D. J. Phelps. “A survey of emerging trend detection in textual data mining”. Proceedings of KDD , vol 4 ,pages 185-224, may 2003.
- [12] Brachman, R., and Anand, T, „The Process of Knowledge Discovery in Databases: A Human-Centered Approach” , Book Chapter, AAAI Press-2003.
- [13] S. Roy, D. Gevry, and W. M. Pottenger. “Methodologies for trend detection in textual data mining”. In the Textmine '02 Workshop, Second SIAM International Conference on Data Mining, Pages 85-94. Jun 2002
- [14]C Fraley, AE Raftery , “Model-Based Clustering, Discriminant Analysis, and Density Estimation” , Journal of the American Statistical Association, Vol 5, Pages 435-448 Aug-2002.
- [15] J. Kleinberg. “Bursty and Hierarchical Structure in Streams”. In Proceedings of KDD-Pages 345-350, canada -2002.

- [16] Soucy P.; Mineau G.W, “A simple KNN algorithm for text categorization”, Proceedings IEEE International Conference on Data Mining, 2001. Pages 647-652, Dec 2001.
- [17] D Hiemstra , “A probabilistic justification for using $tf \times idf$ term weighting in information retrieval” , International Journal on Digital Libraries, 2000 – Springer, Volume 3, Pages131-139 , August, 2000
- [18] Brun, A.; Smaili, K.; Haton , “Experiment analysis in newspaper topic detection.”, Proceedings. Seventh International Symposium on String Processing and Information Retrieval, Page(s):55 – 64 , April 2000.
- [19] A. Popescul, G. W. Flake, S. Lawrence, L. H. Ungar, and C. L. Giles. “ Clustering and Identifying Temporal Trends in Document Databases”. In IEEE Advances in Digital Libraries ADL-2000, pages 173–182, 2000.
- [20] M. A. Hearst. “Untangling text data mining”. In Proceedings of the 37th conference on Association for Computational Linguistics (ACL 1999), pages 3–10, 1999.
- [21] P. Bradley, U. Fayyad, and C. Reina. , “Scaling clustering algorithms to large databases”. In ACM KDD Conference, 1998.
- [22] C Fraley, C. Reina , K.; Haton, ”Algorithms for model-based Gaussian hierarchical clustering” , SIAM Journal on Scientific Computing, vol-4 , Pages 145-156, April -1998
- [23]J Sander, M Ester, HP Kriegel, X Xu , “Density-Based Clustering in Spatial Databases: The Algorithm GDBSCAN and Its Applications”, Proceedings of Data Mining and Knowledge Discovery, Pages- 87-95, 1998 – Springer.

- [24] M Ester, HP Kriegel, J Sander, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise", Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining, Pages 346-357, May -1996.
- [25] Chinrungrueng, C.; Sequin, "Optimal adaptive k-means algorithm with dynamic adjustment of learning rate."; IEEE transactions on neural networks, Volume 6, Issue 1, Page(s):157 – 169, Jan. 1995.
- [26] R. Feldman and I. Dagan., "Knowledge discovery in textual databases (kdt)". In proceedings of KDD, pages 112-117, 1995.
- [27] G. Salton and C. Buckley. "Weighting Approaches in Automatic Text Retrieval." International conference on Information Processing and Management, Pages :513–523, 1988.
- [28] wh.Day, H.E. delsbrenner, "Efficient algorithms for agglomerative hierarchical clustering methods", Journal of Classification, Springer, vol 3, Pages 56-64, july 1984
- [29] A Griffiths, I. Dagan, "Hierarchic Agglomerative Clustering Methods for Automatic Document Classification", Journal of Documentation, Vol 31, Pages 155-169, June 1984.
- [30] http://www.Ksagresearch.com/pdf_files/mapcalc%20training2.pdf

Source Code Listing

```
// finds all the features in the given dataset.  
// All the text file should be in the folder "C:\Input"
```

```
import java.io.*;  
import java.io.File;  
import java.util.StringTokenizer;  
import java.util.*;
```

```
class tokenize{
```

```
    public static void main(String args[]) {  
        try{
```

```
            Hashtable words=new Hashtable();  
            Object frequency;
```

```
            int i=0,f=0;
```

```
            String str;  
            String dirname="C:\Input";
```

```
            FileWriter fout = new FileWriter("C:\features.txt");
```

```
            File dir= new File(dirname);           // dir working
```

```
            if(dir.isDirectory()){
```

```
                String s[]=dir.list();
```

```
                for(int fi=0;fi<s.length;fi++){
```

```
                    File file=new File(dirname+"/"+s[fi]);
```

```
                    if(!file.isDirectory()){           // file operations
```

```
                        FileReader fin = new FileReader(file.getAbsolutePath());
```

```
                        BufferedReader bin=new BufferedReader(fin);
```

```
                        StringTokenizer st;
```

```
                        while(( str=bin.readLine())!=null) {
```

```
                            str=str.toLowerCase();
```

```
                            st=new StringTokenizer(str,"_-$%!#&*+,()[]\\"= ;:\n");
```

```
                            while(st.hasMoreTokens()) {
```

```
                                str=st.nextToken();
```

```
                                if(str.length()<=3)
```

```
                                    continue;
```

```
                                else{
```

```
                                    if((frequency=words.get(str))==null)
```

```
                                        words.put(str,new Integer(1));
```

```
                                    else
```

```
                                        words.put(str,new Integer(((Integer)frequency).intValue()+1));
```

```
                                } // end of else
```

```
                            } // end of while
```

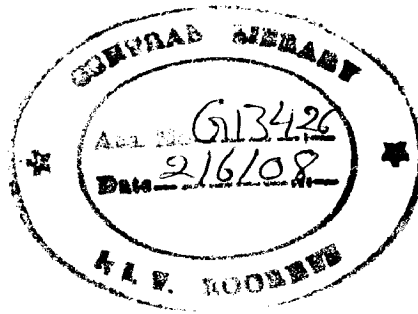
```
                        }
```

```
                            bin.close();
```

```
                        }// file
```

```
                    }//for all files
```

```
                } // if directory
```



```

        Enumeration kys = words.keys();
        while(kys.hasMoreElements()){
            str=(String) kys.nextElement();
            if((((Integer)(frequency=words.get(str))).intValue())>3)
                fout.write(str+"\t"+(Integer)frequency+"\n");
        }

        fout.close();
    }catch(Exception e){ System.out.println(e);}
}

```

.....

// This program removes the stop words that are present in the fearutes list

```

import java.io.*;
import java.io.File;
import java.util.StringTokenizer;
import java.util.*;

class stopwordsremover{

    public static void main(String args[]) {
        try{

            Hashtable words=new Hashtable();
            Object frequency;
            int i=0,f=0,nowords;
            String str,str1,str2;
            FileReader findct = new FileReader("C:\\english.stop");
            BufferedReader bindct=new BufferedReader(findct);
            StringTokenizer st;
            while(( str=bindct.readLine())!=null) {
                str.trim();
                words.put(str,new Integer(0));
            } // end of readline while
            bindct.close();
            String dirname="C:\\tokens";
            FileWriter fout = new FileWriter("C:\\stop.txt");
            File dir= new File(dirname); // dir working
            if(dir.isDirectory()){
                String s[]=dir.list();
                for(int fi=0;fi<s.length;fi++) {
                    File file=new File(dirname+"/"+s[fi]);
                    if(!file.isDirectory()){ // file operations
                        FileReader fin = new FileReader(file.getAbsolutePath());
                        BufferedReader bin=new BufferedReader(fin);
                        while(( str=bin.readLine())!=null) {
                            str.trim();
                            st=new StringTokenizer(str,"\t");
                            str1=st.nextToken();
                            if((frequency=words.get(str1))==null)
                                fout.write(str+"\n");
                            else{}
                        } // end of readline while
                        bin.close();
                    }
                }
            }
        }
    }
}

```



```

        fout.write("\n");
    } // end for each file
} // for all files
fout.close();
} // if directory
} catch(Exception e){ System.out.println(e);}
}
}

```

.....

// This program finds the document_frequency (idf value)for dictionary of words

```

import java.io.*;
import java.io.File;
import java.util.StringTokenizer;
import java.util.*;

class document_frequency {

    public static void main(String args[]) {
        try{

            Hashtable words=new Hashtable();
            Object frequency;
            int i=0,f=0,nowords;
            String str,str1,str2;
            FileReader findct = new FileReader("C:\\stop.txt");
            BufferedReader bindct=new BufferedReader(findct);
            StringTokenizer st;
            while(( str=bindct.readLine())!=null) {
                words.put(str.trim(),new Integer(0));
            } // end of readline while
            bindct.close();
            Hashtable words_doc=new Hashtable();
            Object frequency_doc;
            String dirname="C:\\corpus";
            FileWriter fout = new FileWriter("C:\\docfreq.txt");
            File dir= new File(dirname); // dir working
            if(dir.isDirectory()){
                String s[]=dir.list();
                for(int fi=0;fi<s.length;fi++){
                    File file=new File(dirname+"/"+s[fi]);
                    if(!file.isDirectory()){ // file operations
                        FileReader fin = new FileReader(file.getAbsolutePath());
                        BufferedReader bin=new BufferedReader(fin);
                        while(( str=bin.readLine())!=null) {
                            str=str.toLowerCase();
                            st=new StringTokenizer(str,"_-$%!#&*+,()[]\`= :;.^n");
                            while(st.hasMoreTokens()) {
                                str=st.nextToken();
                                if(str.length()<3)
                                    continue;

```

```

else{
if((frequency_doc=words_doc.get(str))==null)
words_doc.put(str,new Integer(1));
else
words_doc.put(str,new Integer(((Integer)frequency_doc).intValue()+1));
        } // end of else
    } //end of more tokens while
} // end of readline while
bin.close();
Enumeration kys = words_doc.keys();
while(kys.hasMoreElements()){
str=(String) kys.nextElement();
if(((Integer)(frequency_doc=words_doc.get(str))).intValue())>0)
if((frequency=words.get(str))!=null)
words.put(str,new Integer(((Integer)frequency).intValue()+1));
words_doc.put(str,new Integer(0));
        } // end of has more elements while
    } // file
} //for all files
} // if directory

Enumeration kys = words.keys();
while(kys.hasMoreElements()){
str=(String) kys.nextElement();
frequency=words.get(str);
fout.write(str+"\t"+(Integer)frequency+"\n");
    } // end of has more tokens while
fout.write("\n");
fout.close();
}catch(Exception e){ System.out.println(e);}
}
}

```

// This represents whole data set in a vector representaion.

```

import java.io.*;
import java.io.File;
import java.util.StringTokenizer;
import java.util.*;

class gettrainingdata {

    public static void main(String args[]) {
        try{
            int once=0,files=0;
            Hashtable words=new Hashtable();
            Hashtable words1=new Hashtable();
            Object frequency;
            int i=0,f=0,nowords;
            String str,str1,str2,tmp;

```

```

StringTokenizer st;
FileWriter fout1=new FileWriter("C:\\features2.txt");
FileReader findct = new FileReader("c:\\docfreq.txt");
BufferedReader bindct=new BufferedReader(findct );
while(( str=bindct.readLine())!=null) {
st=new StringTokenizer(str.trim(),"\t");
tmp=st.nextToken();
    words.put(tmp,new Integer(0));
    words1.put(tmp,new Integer(st.nextToken()));
    } // end of read line while

bindct.close();
String dirname="C:\\corpus";
FileWriter fout = new FileWriter("C:\\output.arff");
fout.write("@relation thesis"+"\\n");
Enumeration kys = words.keys();
while(kys.hasMoreElements()){
    str=(String) kys.nextElement();
    fout.write("@attribute "+str+" real"+"\\n");
    } // end of has more elements while
fout.write("@data"+"\\n");
File dir= new File(dirname); // dir working
if(dir.isDirectory()){
    String s[]=dir.list();
    for(int fi=0;fi<s.length;fi++) {
    File file=new File(dirname+"/"+s[fi]);
    if(!file.isDirectory()){ // file operations
        files++;
        FileReader fin = new FileReader(file.getAbsolutePath());
        BufferedReader bin=new BufferedReader(fin);
        nowords=0;
        while(( str=bin.readLine())!=null) {
st=new StringTokenizer(str,"_@$%!#&*+,()[\\\"'= ;:.\n");
        while(st.hasMoreTokens()) {
            str=st.nextToken();
            if(str.length()<=3)
                continue;
            else{
                if(!((frequency=words.get(str))!=null))
                    words.put(str,new Integer(((Integer)frequency).intValue()+1));
                nowords++;
            } // end of else

        } // end of has more tokens while
    } // end of read line while
        bin.close();
        Vector kysvector = new Vector(words.keySet());
        Object array[]=new Object[kysvector.size()];
        kysvector.copyInto(array);
        Arrays.sort(array);
        for(i=0;i<array.length;i++){
            str=(String) array[i];
            fout1.write(str+"\\n");
            int df=((Integer)(frequency=words1.get(str))).intValue();
            if(((Integer)(frequency=words.get(str))).intValue()>0) {
                words.put(str,new Integer(0));
            }
        }
    }
}

```

```

    }
    int tf=((Integer)(frequency)).intValue();
    if(df==0) df=1;
    double tfidf=tf*Math.log(674/df)*100;
    if(i<array.length-1)
    fout.write(((int)tfidf)/100.0+",");
    else
    fout.write(((int)tfidf)/100.0+"\n");
    } // end of readline while
    once++;
    } // file
  } //for all files
  fout.close();
} // if directory
fout1.close();
} catch(Exception e){ System.out.println(e);}
}
}

```

// Divides each document into respective folders, each folder represents a cluster

```

import java.io.*;
import java.io.File;
import java.util.StringTokenizer;
import java.util.*;

class folders {

    public static void main(String args[]) {
        try{
            int once=0,i;
            String str;
            int noofclusters=4;
            boolean status;
            for(i=0;i<noofclusters;i++)
                status = new File("C:\\cluster"+i).mkdir();
            StringTokenizer st;
            FileReader findct = new FileReader("c:\\four.arff");
            BufferedReader bindct=new BufferedReader(findct );
            String dirname="C:\\corpus";
            File dir= new File(dirname); // dir working
            String s[]=dir.list();
            for(int fi=0;fi<s.length;fi++){
                File file=new File(dirname+"/"+s[fi]);
                str=bindct.readLine();
            }
            st=new StringTokenizer(str,"/");
            while(st.hasMoreTokens())
                str=st.nextToken();
            String str1=str;
            char ch=str1.charAt(str1.length()-1);

```

```

        int tmp=Integer.parseInt(""+ch);
        File outputFile = new File("C:"+"\\"+"cluster"+tmp+"\\"+file.getName());
        FileReader in = new FileReader(file);
        FileWriter out = new FileWriter(outputFile);
        int c;
        while ((c = in.read()) != -1)
            out.write(c);
        in.close();
        out.close();
    } // end of for
    findct.close();
} catch (Exception e){ System.out.println(e);}
}
}

```

// Describes each cluster

```

import java.io.*;
import java.io.File;
import java.util.StringTokenizer;
import java.util.*;

class clster{
    static double arr[][]=new double[5][2];
    public static void main(String args[]) {
        try{
            double temp;
            String str;
            int i=0,j=0,k=0,l=0;
            for(i=0;i<5;i++)
                for(j=0;j<2;j++)
                    arr[i][j]=-1.0;
            int count=0;
            FileWriter fout=new FileWriter("C:\\clusters.txt");
            File file=new File("C:\\result4.txt");
            FileReader fin = new FileReader(file.getAbsolutePath());
            BufferedReader bin=new BufferedReader(fin);
            StringTokenizer st;
            while(( str=bin.readLine())!=null) {
                str=str.toLowerCase();
                st=new StringTokenizer(str," \t,\n");
                while(st.hasMoreTokens()) {
                    count++;
                    str=st.nextToken();
                    temp= Double.parseDouble(str);
                    if(count<=5)
                    {
                        arr[count-1][1]=temp;
                        arr[count-1][0]=count;
                    }
                }
            }
        }
    }
}

```

```

else if(count>5)
placein(temp,count);
if(count==5)
{
for(k=4;k>=0;k--)
{
for(l=1;l<=k;l++)
{
if(arr[l-1][1] > arr[l][1])
{
double temp3,temp4;
temp3=arr[l-1][1];
arr[l-1][1]=arr[l][1];
arr[l][1]=temp3;
//System.out.println( arr[k][l] + " "+ arr[l][1] + " "+temp3);
temp4=arr[l-1][0];
arr[l-1][0]=arr[l][0];
arr[l][0]=temp4;
// System.out.println( arr[k][0] + " "+ arr[l][0] + " "+temp4);
}
}
}
}
} // end of if count==5
} // end of more tokens
File file1=new File("C:\\output.txt");
FileReader fin1 = new FileReader(file1.getAbsolutePath());
BufferedReader bin1=new BufferedReader(fin1);
String tmpstr=" ";
String str2=" ";
int m;
int temparr[]=new int[5];
for( m=0;m<5;m++)
temparr[m]=(int)arr[m][0];
Arrays.sort(temparr);
int temp2=0;
for( m=0;m<5;m++)
{
int temp1=temparr[m];
for(int n=1;n<=temp1-temp2;n++)
str2=bin1.readLine();
temp2=temp1;
str2=str2.trim();
tmpstr=tmpstr+" "+str2;
} // endof for m

fout.write(tmpstr+"\n");
count=0;
} // end of while
fout.close();

```



```

int day, month, year;
String filename;
FileWriter fout = new FileWriter("C:"+"\\"+"graph4.txt");
File dir= new File(dirname);
if(dir.isDirectory()){
String s[]=dir.list();
for(int fi=0;fi<s.length;fi++){
File file=new File(dirname+"/"+s[fi]);
if(file.isDirectory()){ // file operations
String s1[]=file.list();
for(i=0;i<13;i++)
arr[i]=0;
for(int fj=0;fj<s1.length;fj++){
File file1=new File(dirname+"/"+file+"/"+s1[fj]);
filename=file1.getName();
StringTokenizer st;
st=new StringTokenizer(filename,"-");
day=Integer.parseInt(st.nextToken());
month=Integer.parseInt(st.nextToken());
year=Integer.parseInt(st.nextToken());
if(year==4)
arr[0]++;
else
arr[month]++;

//end of for fj
} // end of if file is directory
for(i=0;i<13;i++)
fout.write(arr[i)+"\t");
fout.write("\n");
} //end of for fi
} // end of if directory is directory
fout.close();
} catch(Exception e){ System.out.println(e);}
}
}

```

//Finds the k-nearest neighbours for all the documents in the data set.

```

import java.io.*;
import java.io.File;
import java.util.StringTokenizer;
import java.util.*;
import java.lang.*;
class influential1{

static double arr[][]=new double[5][2];
public static void main(String args[]) {
try{

```



```

File file=new File("C:\\document.txt");
StringTokenizer st;
int noofdocuments=65;
int counti=0,countj=0;
String str="";
int i,j;
FileWriter fout = new FileWriter("C:\\influential.txt");
for(i=0;i<noofdocuments;i++) {
for(int ii=0;ii<5;ii++)
for(int jj=0;jj<2;jj++)
arr[ii][jj]=-1.0;
int count=0;
counti=0;
Vector v1=new Vector();
FileReader fin = new FileReader(file.getAbsolutePath());
BufferedReader bin=new BufferedReader(fin);
do{
str=bin.readLine();
} while((counti++)!=i);
st=new StringTokenizer(str," ',/t,/n");
while(st.hasMoreTokens()) {
str=st.nextToken();
double tem1= Double.parseDouble(str);
v1.addElement(new Double(tem1));
} // end of more tokens

bin.close();
fin.close();
for(j=0;j<noofdocuments;j++) {
countj=0;
Vector v2=new Vector();
} // end of read line

```

```

FileReader fin1 = new FileReader(file.getAbsolutePath());
BufferedReader bin1=new BufferedReader(fin1);
do{
str=bin1.readLine();
}while((countj++)!=j);
st=new StringTokenizer(str," ',/t,/n");
while(st.hasMoreTokens()) {
str=st.nextToken();
double tem= Double.parseDouble(str);
v2.addElement(new Double(tem));
} // end of more tokens
double temp=finddistance(v1,v2);
fin1.close();
bin1.close();
count++;
if(count<=5)
{
arr[count-1][1]=temp;
arr[count-1][0]=j;
}
}

```

```

        else if(count>5)
            placein(temp,j);
        if(count==5) {
            for(int k=4;k>=0;k--) {
                for(int l=1;l<=k;l++) {
                    if(arr[l-1][1] > arr[l][1]) {
                        double temp3,temp4;
                        temp3=arr[l-1][1];
                        arr[l-1][1]=arr[l][1];
                        arr[l][1]=temp3;

                        temp4=arr[l-1][0];
                        arr[l-1][0]=arr[l][0];
                        arr[l][0]=temp4;
                    }
                }
            }
        }

    }

} // end of for loop j
for(int kk=0;kk<5;kk++)
    fout.write((arr[kk][0]+1)+" ");
fout.write("\n");
} // end of for loop i
fout.close();
}catch(Exception e){ System.out.println(e);}
} // end of main
//} // end of class

```

```

static double finddistance(Vector v1,Vector v2)
{
    int nooffeatures=303;
    int i=0;
    double distance=0;
    for(i=0;i<nooffeatures;i++)
    {
        double x= ((Double)v1.elementAt(i)).doubleValue();
        double y= ((Double)v2.elementAt(i)).doubleValue();
        distance+=(Math.pow((y-x),2));
    }

    return distance;
} // end of find distance

```

```

static void placein(double temp,double count)
{

```

```

int i,j;
double temp1;
if(temp< arr[0][1])
    return;
for(i=1;i<5;i++) {
if(arr[i][1] > temp){
for(j=1;j<=i-1;j++)
    {
        arr[j-1][0]=arr[j][0];
        arr[j-1][1]=arr[j][1];
    }

arr[i-1][0]=count;
arr[i-1][1]=temp;
return;
}
}
for(j=1;j<=4;j++)

    {
        arr[j-1][0]=arr[j][0];
        arr[j-1][1]=arr[j][1];
    }

arr[4][0]=count;
arr[4][1]=temp;
return;
}
} // end of main

```

// finds the scaled lead/lag index for each document in the dataset.

```

import java.io.*;
import java.io.File;
import java.util.StringTokenizer;
import java.util.*;

```

```

class document_frequency {
    public static void main(String args[]) {
        try{

```

```

            FileWriter fout=new FileWriter("C:\\draw.txt");
            PrintWriter pout=new PrintWriter(fout);
            int i=0,f=0,nowords;
            String str,str1,str2;
            double arr[]=new double[5];
            int docno=0;
            File file=new File("C:\\influential.txt");
            FileReader fin = new FileReader(file.getAbsolutePath());
            BufferedReader bin=new BufferedReader(fin);
            while(( str=bin.readLine())!=null) {
                str=str.toLowerCase();

```

```

        i=0;
        StringTokenizer st=new StringTokenizer(str," ',/t,/n");
        while(st.hasMoreTokens()) {
            str=st.nextToken();
            arr[i++]= Double.parseDouble(str);
        } // end of has more tokens while

        int id=0;
        id=findrank(arr,docno);
        docno++;
        pout.println(id);

    } // end of read line

    fout.close();
    pout.close();
    bin.close();
    fin.close();
    findscaledrank();
    }catch(Exception e){ System.out.println(e);}

}

static int findrank(double arr[],int docno)
{
    int i=0;
    String str1,str2;
    String dirname="C:\\cluster0";
    int sday,smonth,syear,dday,dmonth,dyear;
    int klater=0,kearlier=0;
    File dir= new File(dirname);
    String s[]=dir.list();
    File sfile=new File(dirname+"/"+s[docno]);
    str1=sfile.getName();
    StringTokenizer st;
    st=new StringTokenizer(str1,"-");
    sday=Integer.parseInt(st.nextToken());
    smonth=Integer.parseInt(st.nextToken());
    syear=Integer.parseInt(st.nextToken());
    for(i=0;i<5;i++)
    {
        File dfile=new File(dirname+"/"+s[((int)arr[i])]);
        str2=dfile.getName();
        st=new StringTokenizer(str2,"-");
        dday=Integer.parseInt(st.nextToken());
        dmonth=Integer.parseInt(st.nextToken());
        dyear=Integer.parseInt(st.nextToken());
        if(dyear>syear)
            klater++;
        else if(dyear==syear && dmonth>smoth)
            klater++;
        else if(dmonth==smoth && dday>sday)
            klater++;
        else
            kearlier++;
    }
}

```

```

        } //end of for
        int idraw=klater-kearlier;
        return idraw;
    } // end of find rank

```

```

static void findscaledrank()
{
    try
    {
        String str;
        int docno=0,idsum=0,doccount=0,count=0;
        FileWriter fout1=new FileWriter("C:\\idscaled.txt");
        PrintWriter pout=new PrintWriter(fout1);
        FileWriter fout=new FileWriter("C:\\filenames.txt");
        String str1="",str2;
        int sday,smmonth,syear,dday,dmonth,dyear;
        String dirname="C:\\cluster0";
        File dir= new File(dirname);
        String s[]=dir.list();
        for(int t=0;t<s.length;t++)
        {
            File sfile=new File(dirname+"/"+s[t]);
            str1=sfile.getName();
            fout.write(str1);
            fout.write("\n");
        }
        fout.close();
        for(int j=0;j<s.length;j++) {
            int idraw=getidraw(j);
            File sfile=new File(dirname+"/"+s[docno++]);
            str1=sfile.getName();
            StringTokenizer st;
            st=new StringTokenizer(str1,"-");
            sday=Integer.parseInt(st.nextToken());
            smmonth=Integer.parseInt(st.nextToken());
            syear=Integer.parseInt(st.nextToken());
            for(int i=0;i<s.length;i++) {
                File dfile=new File(dirname+"/"+s[i]);
                str2=dfile.getName();
                st=new StringTokenizer(str2,"-");
                dday=Integer.parseInt(st.nextToken());
                dmonth=Integer.parseInt(st.nextToken());
                dyear=Integer.parseInt(st.nextToken());
                if(sday==dday && smmonth==dmonth && syear==dyear)
                {
                    doccount++;
                    idsum+=getidraw(i);
                } //end of if
            } // end of for i
            double idscaled= (double)((double)(1/5.0)*((double)idraw-((double)doccount/(double)
                idsum)));
            pout.println(idscaled+" "+str1);
            doccount=0;idsum=0;
        } //end of for j
    }
}

```

```
    pout.close();
    fout1.close();
} catch (Exception e) { System.out.println(e);}
} //end of find scaled
```

```
static int getidraw(int x)
{
    int idra=0;
try {
    String str;
    int i=-1;
    File file2=new File("C:\\idraw.txt");
    FileReader fin2 = new FileReader(file2.getAbsolutePath());
    BufferedReader bin2=new BufferedReader(fin2);
    do
    {
        str=bin2.readLine();
        i++;
    }while(i!=x) ;
    str=str.trim();
    idra= Integer.parseInt(str);
    return idra;
    } catch (Exception e) { System.out.println(e);}
    return idra;
} //end of get id raw
} // end of main
```