4-2008

Clustering and its Application in Requirements Engineering

Chuan Duan
DePaul

Clustering and its Application in Requirements Engineering

Chuan Duan
School of Computing
DePaul University

duanchuan@cti.depaul.edu

Abstract

Large scale software systems challenge almost every activity in the software development life-cycle, including tasks related to eliciting, analyzing, and specifying requirements. Fortunately many of these complexities can be addressed through clustering the requirements in order to create abstractions that are meaningful to human stakeholders. For example, the requirements elicitation process can be supported through dynamically clustering incoming stakeholders' requests into themes. Cross-cutting concerns, which have a significant impact on the architectural design, can be identified through the use of fuzzy clustering techniques and metrics designed to detect when a theme cross-cuts the dominant decomposition of the system. Finally, traceability techniques, required in critical software projects by many regulatory bodies, can be automated and enhanced by the use of cluster-based information retrieval methods. Unfortunately, despite a significant body of work describing document clustering techniques, there is almost no prior work which directly addresses the challenges, constraints, and nuances of requirements clustering. As a result, the effectiveness of software engineering tools and processes that depend on requirements clustering is severely limited. This report directly addresses the problem of clustering requirements through surveying standard clustering techniques and discussing their application to the requirements clustering process.

1. Introduction

Software requirements specify the goals, functionalities, and constraints of a software system [Zave97]. The discipline of systematically managing requirements is known as requirements engineering (RE). To decompose the problem of managing requirements, RE defines a set of basic tasks, such as elicitation, analysis and validation, and documentation of the requirements within a software requirement specification (SRS). The effectiveness of these tasks, namely the extent to which they improve the overall quality of the software product, depends a good deal on the supporting tools and characteristics of the software project itself. Many manually executed tasks work well in small or medium projects, but are ineffective in large projects. This is illustrated by the failure of FBI Virtual Case File (VCF) project [Gold05]. This was a 170 million dollar project whose functionality was documented in an 800 page requirements specification. As a specialist involved in the VCF project once pointed out, the problems in eliciting, managing, and prioritizing requirements significantly contributed to the disaster. In particular, during the requirement elicitation, significant effort was expended to manually discover and understand the requirements from hundreds of stakeholders but, unfortunately, this huge effort did not translate into a successful product. RE tasks, especially when related to very large projects, are in need of automated support. The crux of the problem is how to automatically and efficiently coordinate large numbers of stakeholders' requests, and to arrange the subsequent requirements into meaningful structures.

Clustering, or cluster analysis, provides a potential solution to help address this problem. Clustering is defined as the automatic division of the data or population into cohesive subsets or clusters. Despite its long history of study, the importance of clustering has become more obvious since the emergence of the internet, with the onslaught of huge volumes of data, generated and accumulated through the exchange of information. Methods needed to be developed to organize the data and to mine useful information. Clustering has been employed widely in text retrieval and mining to address several issues such as retrieval performance improvement [Kowalski97], document browsing [Cutting92], topics discovery [Ertz01], organization of search results [Zamir97], and concept decomposition [Dhillon01].

Clustering methods can be classified according to the nature of the data, such as spatial data, time series data,

and document data. Given that most software requirements are specified as documents in natural language, it is reasonable to adopt theories, methods, and tools from the document clustering discipline.

Admittedly, the two areas of document clustering and requirements clustering, have a lot in common. For example, as mentioned previously, both cases deal with textual information, suggesting that the basic framework of document clustering, including preprocessing techniques, similarity calculations between two documents, clustering algorithms, and validation of clusters, can be adopted in requirements clustering. Both problem domains also share a number of challenges, such as high dimensionality of the data, significant background noise, and the need for scalability.

The clustering of requirements, however, is significantly more difficult than the clustering of ordinary documents in a number of ways. First, the cluster granularity, i.e. the number of clusters, needs to be determined automatically in requirements clustering at a very fine level of granularity. Whereas document clustering stems from the need to sort or filter large collections of texts, such as books, patent articles, or web pages, the purpose, which is usually the sole purpose, of information clustering, is to organize the documents into a limited number of categories to ease a few basic tasks such as browsing and searching. The number of the categories is typically small, and is usually known in advance. On the other hand, the purposes for clustering of requirements are highly variable and are dependent on the tasks for which the generated clusters will be put to use. Many tasks rely upon very fine-grained clusters, and have no existing reference categories, meaning that the granularity and themes of the clusters must be determined automatically.

Second, each domain makes different assumptions about the membership of each datum. Document clustering usually assumes each document comes from one of the fixed numbers of categories; in other words, each document belongs to one and only one cluster. In contrast, crisp clustering assignments are insufficient for requirements, and a single requirement may need to be placed into multiple clusters. Furthermore a significant number of requirements may be outliers which do not belong in any cluster.

Third, the two domains differ in the distribution of topics among the data set. The documents studied in typical document clustering tend to contain rich textual information and exhibit only one dominant significant topic in each document. Software requirements, on the other hand, are typically documented tersely and their common segments tend to be short and sometimes appear trivial. Consequently, the requirements clusters generated by traditional document clustering algorithms are often formed around a dominant topic, while critical cross-cutting concerns are dispersed across multiple clusters. For example, the three requirements shown below, are all related to the topic of login and could reasonably be placed into a login cluster, however they were actually scattered across three more dominant clusters of local display, unique ID's, and informing the employee. A better approach would have been to place them each into two distinct clusters, representing each of their themes.

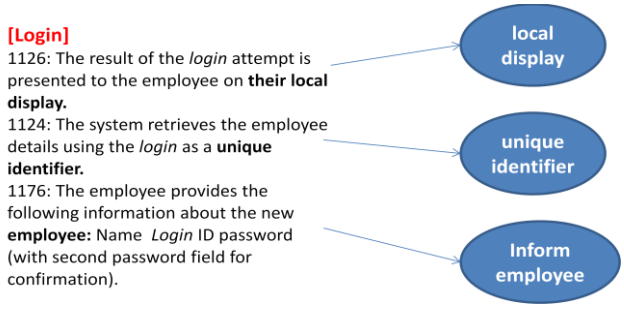

Figure 1.1 Trivial topics distort more meaningful clustering

Finally, document clustering has a wealth of available data sets to support empirical evaluation of clustering algorithms. These include carefully selected large scale document data sets, such as TREC [TREC], UCI KDD data sets [Hettich99], which have been manually cleaned up and classified as reference answer sets for clustering. The fact that requirement clustering has no such answer set to serve the purpose of evaluation, is a significantly non-trivial problem that is addressed later in this proposal. In fact one of the secondary contributions of this research will be the placement of three requirements datasets into the public domain for use in ongoing comparative studies for requirements clustering.

These three problems related to the dynamic determination of granularity, dealing with terse multi-topic requirements, and a lack of a standard answer set pose serious challenges to the requirements clustering problem, which have not been fully addressed and well tackled in existing research. The objective of the research in this proposal is to incorporate these unique challenges into the design and validation of clustering algorithms in order to identify, enhance, or develop clustering algorithms capable of generating cohesive and loosely coupled clusters that provide efficient automated support for a wide variety of RE related tasks.

2. Document clustering

This chapter first introduces the general process of document clustering. Each of the following primary components is discussed: preprocessing, weighting, similarity computation, clustering algorithms, and cluster validation. While the survey focuses on crisp clustering

algorithms, a fuzzy clustering algorithm based on correlation metrics and the neural method of self-organizing maps (SOM), are also discussed. The chapter then reviews the author's prior work on applying traditional clustering algorithms to the requirements domain.

2.1 Definition and Notation

Clustering is defined as the division of a set of objects into K clusters or groups for which the intra-cluster cohesion is maximized and the inter-cluster coupling is minimized. This proposal is devoted to the discussion of clustering on documents and textual software requirements. For a more comprehensive review of various clustering research fields, see [Jain88, Jain99, Berkhin02, Theodoridis06, Duda01].

In this proposal, the name "artifact" is used to refer to a document, which in the requirements domain is synonymous with either a requirement or a raw statement of stakeholder's needs. It is usually represented by an artifact vector whose components correspond to the terms that have been extracted from the artifact collection. Let the set of terms be denoted as T = $\{t_1, t_2, \ldots t_d\}$, then the whole artifact collection can be represented as a term-by-document matrix $A = [a_1, a_2, \ldots, a_N]$ where each column corresponds to an artifact. The component value $a_{ik}, k = 1,2, \ldots, d$ in artifact $a_i, i = 1,2, \ldots, N$ denotes the weight of term t_k for a_i. This weight could simply be the number of occurrences of t_k in a_i, but is more typically a score computed by taking additional factors into account, a procedure called "weighting" that is to be discussed in section 2.4.

In the remainder of the discussion the following notation is adopted. Regular letters denote scalars, small-bold letters such as **x** or **y** denote any of the artifact vectors, and capital-bold letters such as **A**, **B** denote matrices.

2.2 Components of document clustering process

For systematic studies, the process of document clustering is generally described by decomposing the clustering process into the following components: preprocessing, weighting, similarity calculations, grouping by use of a clustering algorithm, and cluster validation. As shown in Figure 2.1, the sequence of these components includes two feedback paths. The first one represents the output of the grouping algorithm fed back into the next round of computations, while second feedback path represents the feedback from evaluating the clusters, which impacts the next iteration of grouping.

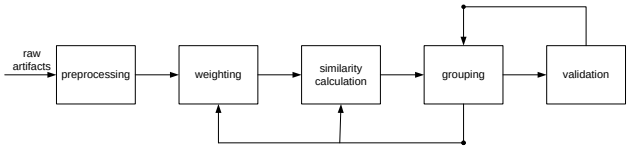

Figure 2.1 Components of a document clustering process

2.3 Preprocessing of raw artifacts

The purpose of preprocessing is to remove any information that is either insignificant, or detrimental to the clustering. First the text is split into meaningful tokens, which are generally referred to as ords. Next, stop words i.e. extremely common words including articles, pronouns, and any other frequently occurring term such as *do* and *make*, are eliminated since they do not provide useful information for helping to differentiate between different documents. Remaining words are then stemmed to their root forms. These stemmed words are typically referred to as terms. Finally, it is possible, although not yet demonstrated to return consistently improved results [REF], to use a thesaurus or explicitly constructed matching word listto unify the occurrence of synonyms or other forms of domain equivalencies. Following this step, any term that occurs only once in the entire document collection can also be removed, as these terms are not useful for clustering purposes. After these preprocessing steps, a raw artifact is represented by a vector whose components correspond to the terms determined to be significant in the collection.

2.4 Weighting of terms

The term weights represent values attached to each term to indicate their importance within an artifact. Three main components that are used to compute a term weight include: the term frequency (*tf*), the inverse document frequency factor (*idf*), and a document length normalization (*dl*) factor [Salton86]. The most frequently used term weighting is the product of *tf* and *idf*, referred to as *tf-idf* and computed as:

$$w_{ij} = tf * idf = f_{ij} * \log\frac{N}{N_j}$$

where f_{ij} is the occurrence of term j in document i, N is the total number of documents, and N_j is the number of documents in which term j appears at least once. This simple weighting scheme is very widely used because it is

intuitively sound – the more a term appears in a collection, the less useful information it provides for computing similarity between two documents.

Despite the success of *tf-idf* and its variations, a number of additional weighting schemes have been proposed and empirically proven to be more efficient. One of them, pivoted document length normalization weighting [Singhal96], is defined as

$$w_{ij} = \frac{l_j / l_j^{'}}{(1-s) \times p + s \times u}$$

where $l_j = \sum_i f_{ij}$, $l_j^{'} = N_j$, s is the slope constant, p is the average number of distinct terms throughout the collection, and u is the number of distinct terms in document i. In a study by Singhal [Singhal96] this weighting was demonstrated to obtain 13.7% more relevant documents [Singhal96].

Latent Semantic Analysis

Another type of term weighting, more commonly called indexing, attempts to capture the semantic relationship between documents by the reduction or transformation of term dimensions. It can be viewed as an implicit domain thesaurus. Among many such schemes of term indexing, Latent Semantic Indexing (LSI), or Latent Semantic Analysis (LSA) has been shown to be able to filter noisy data and absorb synonymy i.e. the use of two different terms that share the same meaning, and polysemy i.e. the use of a single term to mean to distinct things, in large corpus [Deerwester90, Dumais93, Dumais95, Berry05]. The basic derivation of LSI is as follows. Let **X** be the term by document matrix

$$\begin{bmatrix} x_{1,1} & \cdots & x_{1,n} \\ \vdots & \ddots & \vdots \\ x_{m,1} & \cdots & x_{m,n} \end{bmatrix}$$

$\boldsymbol{t}_i = [x_{i,1}, \dots, x_{i,m}]$ is the occurrence vector of term i, and $\boldsymbol{d}_j^T = [x_{1,j}, \dots, x_{m,j}]$ is the vector of document j. The dot-product $\boldsymbol{t}_i \boldsymbol{t}_p^T$ then gives the correlation between terms, and matrix $\boldsymbol{X}\boldsymbol{X}^T$ contains all of the correlations. Likewise, $\boldsymbol{d}_j^T \boldsymbol{d}_q$ represents the correlation between documents, and matrix $\boldsymbol{X}^T\boldsymbol{X}$ stores all such correlations.

Singular Value Decomposition (SVD) is applied to $\boldsymbol{X}$ to produce three components:

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$$

where $\boldsymbol{U}$ and $\boldsymbol{V}$ are orthonormal matrices and $\sum$ is a diagonal square. Applying this factorization to $\boldsymbol{X}\boldsymbol{X}^T$ and $\boldsymbol{X}^T\boldsymbol{X}$:

$$\begin{aligned} \boldsymbol{X}\boldsymbol{X}^T &= (\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T)(\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T)^T = (\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T)(\boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{U}^T) \\ &= \boldsymbol{U}\boldsymbol{\Sigma}^2\boldsymbol{U}^T \\ \boldsymbol{X}^T\boldsymbol{X} &= (\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T)^T(\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T) = (\boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{U}^T)(\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T) \\ &= \boldsymbol{V}\boldsymbol{\Sigma}^2\boldsymbol{V}^T \end{aligned}$$

In other words, the columns of $\boldsymbol{U}$ are the eigenvectors of matrix $\boldsymbol{X}\boldsymbol{X}^T$, the columns of the $\boldsymbol{V}$ are the eigenvectors of matrix $\boldsymbol{X}^T\boldsymbol{X}$, and $\sum$ is the square root of the eigenvalues of matrix $\boldsymbol{X}\boldsymbol{X}^T$ or $\boldsymbol{X}^T\boldsymbol{X}$. This can also be denoted as follows:

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T = [\boldsymbol{u}_1, \boldsymbol{u}_2, \dots, \boldsymbol{u}_l] \begin{bmatrix} \sigma_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_l \end{bmatrix} \begin{bmatrix} \boldsymbol{v}_1^T \\ \boldsymbol{v}_2^T \\ \dots \\ \boldsymbol{v}_l^T \end{bmatrix}$$

The selection of k largest singular values, and the corresponding singular vectors from $\boldsymbol{U}$ and $\boldsymbol{V}$, constitutes a rank K approximation to $\boldsymbol{X}$, $\boldsymbol{X}_k = \boldsymbol{U}_k\boldsymbol{\Sigma}_k\boldsymbol{V}_k^T$, with the smallest error in terms of Frobenius Norm, where each artifact $\boldsymbol{d}_i$ can be represented by K weights $\boldsymbol{v}_i$. Furthermore, this approximation transforms the original purely physical occurrence into the relationship in the concept space, leading to a new similarity calculation between terms or documents.

However, LSA has three serious problems. First, the purely matrix factorization derivation of LSI makes the resulting dimensions difficult to interpret. Second, and more important, LSA assumes that words and documents form a joint Gaussian model, while is against the commonly observed Poisson distribution. And last, the optimal k must be empirically determined by numerous trials.

Non-negative matrix factorization

Non-negative matrix factorization (NMF) represents a similar dimensional transformation technique to LSA. NMF factorizes a matrix $\boldsymbol{X}$ into two matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ with the constraints that the elements in $\boldsymbol{U}$ and $\boldsymbol{V}$ are non-negative, namely, $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{V}^T$, $u_{ij} \geq 0$ and $v_{ij} \geq 0$. NMF for a matrix is not exclusive and the choice of factorization depends on the divergence of resulting factorization $\boldsymbol{U}\boldsymbol{V}^T$ from the original matrix $\boldsymbol{X}$. For example, as discussed in

[Xu03], the Frobenius norm [1] can be used as the divergence criterion, whose optimization involves the minimization of the objective function $\|\boldsymbol{X}-\boldsymbol{U}\boldsymbol{V}^T\|_F$ or equivalently $\frac{1}{2}\|\boldsymbol{X}-\boldsymbol{U}\boldsymbol{V}^T\|_F^2$:

$$J=\frac{1}{2}\|\boldsymbol{X}-\boldsymbol{U}\boldsymbol{V}^T\|_F^2=\frac{1}{2}tr\big((\boldsymbol{X}-\boldsymbol{U}\boldsymbol{V}^T)(\boldsymbol{X}-\boldsymbol{U}\boldsymbol{V}^T)^T\big)$$
$$=[tr(\boldsymbol{X}\boldsymbol{X}^T)-2tr(\boldsymbol{X}\boldsymbol{V}\boldsymbol{U}^T)+tr(\boldsymbol{U}\boldsymbol{V}^T\boldsymbol{V}\boldsymbol{U}^T)]$$

with the constraints that $u_{ij}\geq 0\ and\ v_{ij}\geq 0$, and by introducing proper Lagrange multipliers, the following iterative estimation of $\boldsymbol{U}$ and $\boldsymbol{V}$ is reached:

$$u_{ij}^m=u_{ij}^{m-1}\frac{(\boldsymbol{X}\boldsymbol{V})_{ij}}{(\boldsymbol{U}\boldsymbol{V}^T\boldsymbol{V})_{ij}}$$
$$v_{ij}^m=v_{ij}^{m-1}\frac{(\boldsymbol{X}^T\boldsymbol{U})_{ij}}{(\boldsymbol{V}\boldsymbol{U}^T\boldsymbol{U})_{ij}}$$

The whole iteration has time complexity O(tKN), where t is the number of the iterations.

Similarly to LSA, NMF discovers a latent semantic space from the data, in which each axis captures the base topic of a candidate document cluster. Each document is then represented as an addictive combination of the base topics. NMF differs noticeably from LSI in two aspects. First, the latent space found by NMF does not need to be orthogonal. Second, and more important to clustering, the projection values are all positive, so that the clusters could be directly derived from $\boldsymbol{V}$ – i.e. the cluster membership of each document is determined by finding the base topic or topics with which the document has the largest projection value. Nevertheless the dimensions found by NMF can still be hard to interpret.

2.5 Similarity calculation between artifact vectors

In most heuristic algorithms, requirement clustering is strongly dependent upon computing the similarity between pairs of documents. Therefore the similarity computation of two artifact vectors can significantly impact the quality of the resulting clustering. The concepts of distance and similarity are complementary, with distance representing the level of dissimilarity between two documents, and similarity denoting the level to which they resemble each other. The more general word "proximity" is therefore sometimes used to denote a certain metric between two artifacts which can be expressed either as similarity or distance. The commonly used proximity metrics for documents include:

Correlation. For two artifact represented as column vectors $\mathbf{x} = (x_1, x_2, \ldots, x_d)^T$ and $\mathbf{y} = (y_1, y_2, \ldots, y_d)^T$, their un-normalized correlation is their dot product:

$$c(\boldsymbol{x},\boldsymbol{y})=(\boldsymbol{x},\boldsymbol{y})=\sum_{i=1}^{d}x_i\,y_i$$

Euclidean Distance. Euclidean distance measures the distance between vector **x** and **y** in d-dimensional space:

$$d_e(\boldsymbol{x},\boldsymbol{y})=\|\boldsymbol{x}-\boldsymbol{y}\|=\sqrt{\sum_{i=1}^{d}(y_i-x_i)^2}$$

Euclidean distance is a special case of the Minkowski metric when θ is set to 2:

$$d_{mks}(\boldsymbol{x},\boldsymbol{y})=\left(\sum_{i=1}^{d}|y_i-x_i|^{\theta}\right)^{1/\theta}$$

Direction Cosine. While Euclidean distance is concerned with the absolute distance between vectors, the direction cosine measures the similarity purely based on the relative magnitudes of the features:

$$s_c(\boldsymbol{x},\boldsymbol{y})=\cos<\boldsymbol{x},\boldsymbol{y}>=\frac{(\boldsymbol{x},\boldsymbol{y})}{\|\boldsymbol{x}\|\|\boldsymbol{y}\|}$$

where $\|x\|$ and $\|y\|$ are the Euclidean norms of the vector, defined as $\sqrt{\sum_{i=1}^{d}(x_i)^2}$. Their distinction can be observed from Figure 2.2. It should be noted that if vectors x and y have been normalized with regard to the norm, $\|\boldsymbol{x}\|=\|\boldsymbol{y}\|=1$, then the Euclidean distance is completely complementary to the dot-product, since $\|\boldsymbol{x}-\boldsymbol{y}\|^2=(\boldsymbol{x}-\boldsymbol{y})^T(\boldsymbol{x}-\boldsymbol{y})=\|\boldsymbol{x}\|^2+\|\boldsymbol{y}\|^2-2\boldsymbol{x}^T\boldsymbol{y}=-2cos<\boldsymbol{x},\boldsymbol{y}>$.

[1] The Frobenius norm of a matrix $\mathbf{A} = (a_{ij})$ is the sum of square of all the elements of **A**: $\|\boldsymbol{A}\|_F=\sqrt{\sum_{i,j}a_{ij}^2}$.

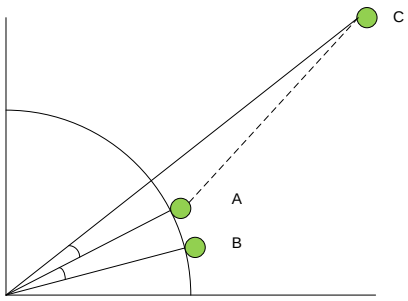

Figure 2.2 Normally Euclidean distance and cosine direction give unrelated scores for raw vectors, demonstrated by the fact that $\angle AC$ is incompatible to the measure of $\overline{AC}$; on the other hand, for normalized vectors, the two metrics function equally.

Hamming distance. Originally defined for binary codes, Hamming distance can be used to compare any ordered sets that consist of discrete-valued elements. It defines the dissimilarity of two vectors of the same lengths to be the number of different symbols in them normalized by the length of the vector:

$$d_H(\boldsymbol{x},\boldsymbol{y}) = \frac{count\{x_i \neq y_i\}}{|\boldsymbol{x}|}, i = 1,2,\dots,d$$

Probabilistic similarity. In information retrieval, the similarity between two documents can be formulated as the inference probability from one document to another. Formally, the inference of document **x** given query **y** in a d-term space can be defined as the posterior probability:

$$s_b(\boldsymbol{x},\boldsymbol{y}) = p(\boldsymbol{x}|\boldsymbol{y}) = \frac{p(\boldsymbol{x},\boldsymbol{y})}{p(\boldsymbol{y})} = \frac{\sum_{i=1}^{d} p(\boldsymbol{x}|t_i)p(\boldsymbol{y}|t_i)p(t_i)}{\sum_{i=1}^{d} p(\boldsymbol{y}|t_i)p(t_i)}$$

There are many alternate methods for calculating $p(\boldsymbol{x}|t_i)$. Some assume a specific distribution of terms such as Poisson or multinomial [Zaragoza03] and then integrate the probability density function in the inference, while others assume no known parametric distributions and proceed in an ad hoc way. One of the latter techniques uses the frequency of term t_i in the **x** $f_{x,i}$ to estimate $p(\boldsymbol{x}|t_i)$, an approximation that leads to $p(\boldsymbol{x}|t_i) = f_{x,i}/\sum_k f_{x,k}$ and $p(\boldsymbol{x}|t_i) = f_{x,i}/\sum_k f_{x,k}$. The estimation of $p(t_i)$ usually follows the idea of reversed term frequency discussed in the last section on weighting, namely, $p(t_i) = n_i/N$, where n_i is the total occurrence of t_i, and N is the total number of the artifacts. Notice that unlike the three proximities just discussed, s_b is asymmetric, i.e., the belief acquired from **x** to **y** is usually different from the one acquired from **y** to **x** given that x and y are different.

The choice of proximity calculation technique depends on the nature of the data, representation of the data, and other requirements or constraints. For example, in spatial data clustering, Euclidean distance is a natural choice, while in document clustering, which depends on the frequencies of components rather than on their absolute scores, Cosine distance is more appropriate and therefore more widely used, as reported in [Feature Projection]. However if the speed of calculation is important, asymmetric metrics should be used with great caution because of the extra time needed to normalize averages such as $p(\boldsymbol{x}|\boldsymbol{y})$ and $p(\boldsymbol{y}|\boldsymbol{x})$.

2.6 Crisp document clustering algorithms

For an artifact collection with size m the number of possible K-clusterings has been proven to be the Stirling number of the second kind [Anderberg73]

$$S_m^{(K)} = \frac{1}{K!}\sum_{k=0}^{K}(-1)^k C_K^k (K-k)^m$$

Even for a very small requirement collection, this number would be huge. So instead of brute-force evaluation of each possible clustering, a heuristic search algorithm must be designed to converge towards an optimal solution quickly.

A high quality clustering requires clusters to be internally cohesive and to exhibit low inter-cluster coupling, and this goal can be expressed as an objective function. A clustering algorithm can then be viewed as an optimization process that either implicitly or explicitly satisfies designated objective functions either at a local or global level. In structure, the optimization can be bottom-up, top-down, or flat iterative. The taxonomies of crisp clustering algorithms, which for completeness purposes are not limited to the ones used in clustering requirements, are shown in Figure 2.3.

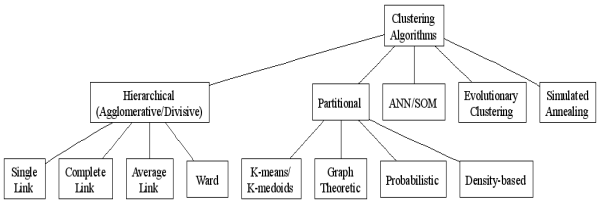

Figure 2.3 Classification of clustering algorithms

Before reviewing each of these algorithms, one important point must be made: just like a deck of poker cards may be arranged effectively but in different ways by different players, it is also true that no universally optimal clustering algorithm exists. Each clustering algorithm makes implicit assumptions about the shape of clusters the data should exhibit, has different capabilities for handling high dimensionality and large scale data, consumes various lengths of time, adopts different strategies to tackle outliers, and so forth.

Hierarchical algorithms have been frequently used for clustering clustering documents. They are divided into agglomerative (bottom-up) and divisive (top-town) [Hartigan75, Jain88] approaches. The agglomerative algorithms start from singleton clusters and continuously merge the most similar clusters, while divisive ones begin with a large cluster containing all of the data and recursively split the least cohesive cluster. The decisions that traditional hierarchical clustering algorithms make at merging or splitting are based on the linkage metric i.e. the similarity or distance between two clusters. Common approaches include the single link, which calculates the shortest distance between objects in each of the two clusters, complete link, which calculates the two farthest objects, and average link which computes the average. A slightly different approach is adopted in the Ward algorithm, which uses an objective function similar to the one used in K-means. The complete link, average link, and Ward work well only in finding tightly bound or compact clusters. In contrast, the single-link algorithm is more versatile – it can not only extract concentric clusters, as shown in Figure 2.4, but can also find the clusters that are mixed with noise patterns. However, it suffers from a chaining effect [Nagy68], which means it has a tendency to produce clusters that are straggly or elongated. These traditional algorithms are used less nowadays because their typical time complexity is $O(N^2)$ which is not cost-effective when clustering a large amount of data, and also they are not able to revisit clusters that have already been formed in order to perform additional optimizations or reclustering. Despite these limitations, they may still prove useful within the RE domain, as many of the initial clustering tasks can be performed offline as batch processes, meaning that running time is not as significant as cluster quality.

Some sophisticated hybrid hierarchical algorithms have been proposed to alleviate these weaknesses. The algorithm CURE, described in [Guha98], represents each cluster by a fixed number of points instead of simply by a centroid or medoid. This algorithm is therefore able to identify non-spherical shapes and to dampen the effect of outliers. It also improves scalability through using random sampling and partitioning. The 2-phase algorithm CHAMELEON [Karypis99] uses dynamic modeling to measure the similarity between two clusters. In the first phase, a K-nearest neighbor connectivity graph is generated and produces small tight clusters. In the second phase, these tight clusters, as well as processed interim clusters, are recursively merged only if the mutual inter-connectivity and closeness are relatively high in comparison to the internal inter-connectivity and closeness. This dynamic modeling was found to be able to identify the clusters that CURE and DBSCAN [Sander98] failed to identify. Another agglomerative hierarchical algorithm proposed by Chiu et al. [Chiu01] adopted a probabilistic technique

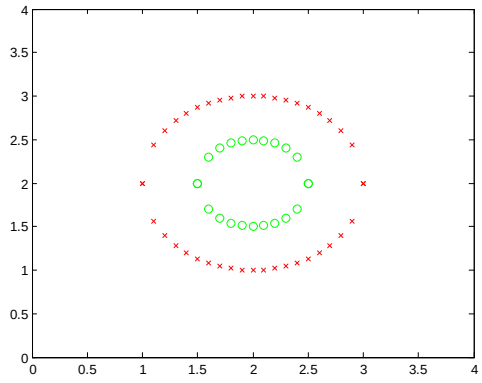

Figure 2.4 The versatility of single-link clustering [Jain99]

for measuring similarity between Gaussian distributed clusters. For each cluster c, its log likelihood is $l_c = \sum_{x \in c} \log p(x|c)$, and then the distance between two clusters is defined as a descent likelihood $d(c_1, c_2) = l_{(c_1)} + l_{(c_2)} - l_{(c_1 \cup c_2)}$. By using this model-based measure, the merging is able to effectively filter out outliers. Unfortunately, these effective algorithms are usually applied to spatial data where shape and density is often geometrically clear. They become inept in dealing with the documents, since documents reside in very high dimensional space where similarity is calculated using correlation instead of Euclidean distance.

In contrast to the hierarchical clustering algorithms which construct hierarchies, partitional clustering algorithms iteratively optimize a flat separation structure. The subclasses of partitional algorithms are relocation method, probabilistic method, density-based method, and graph-theoretic method. The relocation method reassigns the data points to its nearest cluster usually guided by an objective function. It has two variations differentiated by different choices for representing a cluster. K-Medoids methods, such as PAM [Kaufman90], CLARA [Kaufman90], and CLARANS [Ng02], choose a data point within a cluster as a cluster representative, whereas K-means methods, such as Forgy's algorithm [Forgy65] and incremental K-means [Duda01], calculate the arithmetic mean of a cluster as its representative. K-medroids methods present no limitation on data types and are less sensitive to the outliers. Although affected by outliers, K-means has the advantage of clear geometric and statistical meaning [Dhillon01a]. Probabilistic methods, which will be elaborated further in Chapter 4, assume closed form statistical models $p(c_i|x)$ for each cluster, learn this model from the data, and classify the data by a Bayesian discriminate:

$$p(c_i|x) = \frac{p(x|c_i)p(c_i)}{p(x)}$$

Although often suffering from the initialization problem that a bad initial configuration will lead to local optima, relocation and probabilistic clustering methods are used widely in clustering documents thanks to their O(N) time efficiency. For example, the series of papers by Dhillon present an excellent, thorough introduction to K-means' application in document clustering, discussing basics [Dhillon01b], scalability [Dhillon01b], heuristic improvement [Dhillon02], and utilization [Dhillon01a].

The third subclass of partitional methods are density-based algorithms, including DBSCAN [Sander98], DBCLASD [Xu98], etc, while the fourth subclass of graph-theoretic methods are gaining in popularity because of their success in image segmentation [Shi00, Meila01]. They will not be discussed in this proposal because of the strong reliance on spatial relationships for density-based methods and high computational cost ($O(N^2)$) for Graph-theoretic methods.

It is far from trivial to determine the best clustering algorithm for a specific clustering task, however some hybrid hierarchical clustering algorithms, such as PDDP [Boley98] and bisecting 2-means [Zhao02] have been shown in document clustering to outperform purely hierarchical methods or purely partitional methods.

2.7 Fuzzy K-means based on correlation metrics

This fuzzy algorithm, also known as FCM, is a generalized K-means clustering, with an extended objective function defined over correlation metric space:

$$J_\alpha(U, M) = \sum_{i=1}^{N} \sum_{j=1}^{K} u_{ji}^{\alpha}(1 - x_i \cdot m_j)$$

where u_{ji} is the membership assignment of artifact i to the cluster j, and m_i is the centroid of clustering j, and α is a hyper-parameter to control the magnitude of calculated norm. To minimize $J_\alpha(U, M)$, the updating of membership scores and centroids follow as [Rodrigues04]:

$$u_{ji} = \left[\sum_{h=1}^{K} \left(\frac{1 - x_i \cdot m_j}{1 - x_i \cdot m_h} \right)^{\frac{1}{\alpha - 1}} \right]^{-1}$$

and

$$m_j = \sum_{i=1}^{N} u_{ji}^{\alpha} x_i \cdot \left[\sum_{j=1}^{d} \left(\sum_{h=1}^{N} u_{ji}^{\alpha} \cdot x_{hj} \right)^2 \right]^{-\frac{1}{2}}$$

where as usual, N is the number of artifacts, and d is the number of terms.

2.8 Self-organizing maps (SOM)

A self-organizing Map (SOM) is a Vector Quantization (VQ) and neural data projection technique, often used in clustering and visualizing high-dimensional data [Kaski97, Kohonen01, Vesanto97]. The high-dimensional input data are mapped into 2-dimensional or 3-

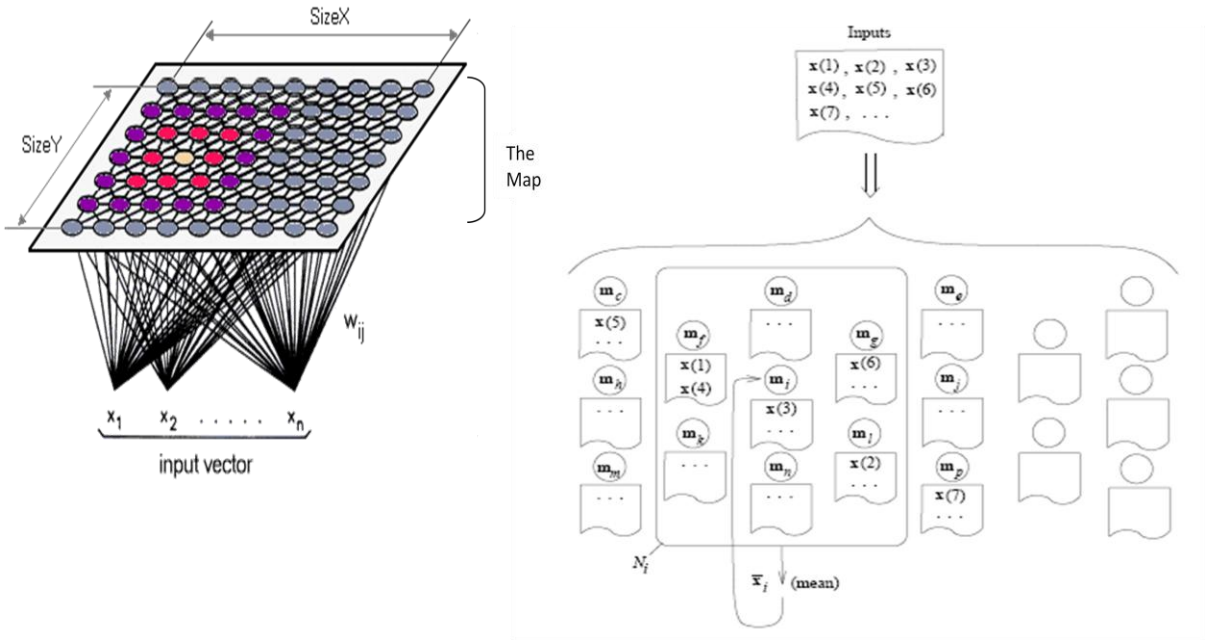

Figure 2.5 The structure and training of an SOM [Kohonen01]

dimensional lattices to approximate the density distribution of the input. On one hand, similarly to K-means and principal curves projection [Hastie89], SOM is a VQ method, which finds a series of vectors called codebooks or models, so as to represent a large collection of vectors. On the other hand, SOM tries to preserve the topological relationship among the input vectors so that the adjacent map units resemble each other coherently. Combining these two typically contradictory facets, SOM achieves a tradeoff between VQ resolution and topological preservation.

In a nutshell, with input vectors x(t), the time t = 0,1,…,N, as shown in Figure 2.5, the SOM array or lattice is comprised of an ordered set of codebook units m_i that act as representatives of similar inputs. The array is updated nonlinearly through a number of training iterations. Each iteration goes through two steps: first every input is attached to the best matched unit (BMU), i.e. the model that is most related; then after all the inputs are classified, each model m_i is updated as the mean or median of the N_i neighbor associated inputs within a certain radius. The process of iterations stops when the values of codebook converge.

The cluster structure of a trained SOM can be viewed by a U-matrix. It stores the similarity scores between adjacent array units in an orderly manner and can be visualized as a "bordered" SOM, where a group of darker cells represents a possible cluster and a series of brighter cells represents a possible cluster border. Figure 2.6 is the U-matrix visualization of Iris data [Anderson35], a classical dataset consisting of 50 samples from each of three species of Iris flowers (Iris setosa, Iris virginica and Iris versicolor). From U-matrix it is easy to identify two clusters distributed vertically on the map although the formation of the third one is not very sharp.

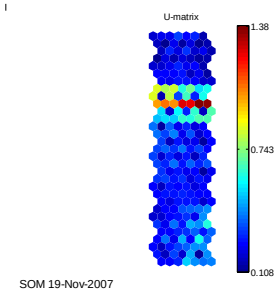

Figure 2.6 The U-matrix of Iris data

2.8.1 Formal definition of SOM training

An incremental SOM training process using Euclidean distance measures is formally described as follows [Kohonen01]:

Initialization

The coordinates of models could be initialized randomly, or linearly, where vectors are calculated in an orderly fashion along the linear subspace spanned by the two

principal eigenvecters of input data set using Gram-Schmidt orthogonalization.

Iteration in time t+1

(1) With each input *x*, the best-matching unit $c = argmin_i\{\|\boldsymbol{x} - m_i\|\}$ is identified

(2) Each model j is updated as
$m_i(t+1) = m_i(t) + h_{ci}(t)[x(t) - m_i(t)]$
Where $h_{ci}(t)$ is a neighborhood function, a smoothing kernel, usually defined as $h_{ci}(t) = h(\|r_c - r_i\|, t)$.

In practice, there are two simple choices for $h_{ci}(t)$. In the first one, $h_{ci}(t) = \alpha(t)$ if $i \in N_i$, and 0 otherwise. $0 < \alpha(t) < 1$ can be any function that decreases monotonically in time, such as $\alpha(t) = 0.9(1 - \frac{t}{1000})$. The second choice takes into account the distance of unit j and BMU in $h_{ci}(t) = \alpha(t) \cdot exp\,(-\frac{\|r_c - r_i\|^2}{2\sigma^2(t)})$, where both $0 < \alpha(t) < 1$ and $0 < \sigma(t) < 1$ are decreasing monotonically in time.

Due to the high time complexity of the incremental version, SOM can also be trained in batch mode. Apparently in the equilibrium of SOM, since $m_i(t+1) = m_i(t)$, then $E[h_{ci}(x - m_i^*)] = 0$, where x is one of the closest data point to m_i. By expansion,

$$\int h_{ci}(x - m_i^*)p(x)dx = 0 \overset{\Delta}{\rightarrow} m_i^* = \frac{\int xp(x)dx}{\int p(x)dx}$$

which means each m_i^* must coincide with the centroid of the respective influence region. This observation leads to batch SOM training, where the models are updated by all of the input vectors simultaneously, as Figure 2.5 has shown.

The training of SOM depends on several parameters which must be explicitly specified. These include the shape and model number of the SOM array, radius of a neighbor, and iteration times. Among them, the size of the SOM array significantly influences the training time. To achieve a good result, the number of iterations must normally be at least 500 × (array size).

2.8.2 Quality measurement

The quality of SOM is usually measured in terms of topology preservation, VQ resolutions, or a combination of both of them.

Since SOM is the mapping of the original data density, it should not exhibit significant differences between adjacent units, meaning that it should be smooth. This smoothness can be calculated as

(a) $\Phi(k) = \sum_j \left|\{i \,|\, \|m_i - m_j\| > k; C_{ij} = 1\}\right|$, C_{ij} = 1 if unit i and j are the two closest BMUs of ANY input vector x ;

(b) $\epsilon_t = \frac{1}{N}\sum_{k=1}^{N} u(x_k)$, where $u(x_k) = 1$ if the corresponding closest two BMUs are not adjacent.

In the perspective of data clustering, the quantization error over the whole testing data: $\epsilon_q = \frac{1}{N}\sum_{k=1}^{N}\|x_k - m_c\|$, should be minimized. So a combined quality measure is appropriate, such as

$$\epsilon_{tq} = E\left[\|x - m_i\| + min\left\{\sum_{k=i}^{k=j}\left\|m_{k' \in N_{k,l}} - m_k\right\|\right\}\right]$$

where the second term calculates the minimum path from 1st BMU to 2nd BMU.

2.9 Granularity determination

In some clustering tasks the number of clusters to be generated, also referred to as the 'stopping criterion' is predefined. However in many cases, the granularity needs to be determined automatically at runtime by the clustering process. There are five common approaches to estimating the granularity of a clustering. These include cross-validation, penalized likelihood estimation, permutation tests, re-sampling, and finding the significant turning point of a metric curve [Salvador04]. Model-based methods, such as cross-validation and penalized likelihood estimation, are computationally expensive and often require the clustering algorithm to be run several times. Permutation tests and re-sampling are extremely inefficient, since they require the entire clustering algorithm to be re-run hundreds or even thousands of times. Even worse, many of the evaluation functions that are used to evaluate a set of clusters run in $O(N^2)$ time. This means that it may take longer just to evaluate a set of clusters than it does to generate them.

The fifth approach, which searches for a significant turning point, is more widely used in practice. A statistical based validation metric is computed during the sequence of clustering and then all the scores of this metric are composed into a score curve. The appropriate

number of clusters is determined by locating a significant point, which can either be a maximum or minimum, or turning points represented as a "knee" or "elbow" of the score curve. There are methods that statistically evaluate each point in the score curve and pick the significant points automatically. Such methods include the gap statistic [Tibshirani03], prediction strength [Tibshirani01], and "L" method [Salvador04]. These methods generally require the entire clustering algorithm to be run for each potential number of clusters. However, for hierarchical algorithms computation is inexpensive, because the only difference between two successive clusters numbered K and K-1 is one additional merge or split.

Generally these validation metrics are categorized into internal metrics which measure cohesion, external metrics which measure coupling, and hybrid metrics.

Cohesion metrics

For a clustering comprised of cluster set $c_1, c_2, \dots, c_K$, the internal metrics evaluate the cohesion of clusters by considering either the similarity between artifacts and the centroid or the similarity between each possible pair of artifacts. For the convenience of formulation, for cluster c_k, let a composite vector $D_k = \sum_{x \in c_k} x$ represent the sum of its contained artifact vectors, and $m_k = \frac{D_k}{n_k} = \frac{\sum_{x \in c_k} x}{n_k}$ represent the vector of the centroid, cohesion metrics are then defined as

$$I_1 = \sum_{k=1}^{K} \sum_{x \in c_k} \frac{x^T m_k}{\|m_k\|} = \sum_{k=1}^{K} \sum_{x \in c_k} \frac{x^T D_k}{\|D_k\|}$$
$$= \sum_{k=1}^{K} \sum_{x \in c_k} x^T \frac{D_k}{\|D_k\|} = \sum_{k=1}^{K} D_k^T \frac{D_k}{\|D_k\|}$$
$$= \sum_{k=1}^{K} \|D_k\|$$

$$I_2 = \sum_{k=1}^{K} \frac{\sum_{x,y \in c_k} x^T y}{n_k} = \sum_{k=1}^{K} \frac{\sum_{x \in c_k} x^T \sum_{y \in c_k} y}{n_k}$$
$$= \sum_{k=1}^{K} \frac{D_k^T D_k}{n_k} = \sum_{k=1}^{K} \frac{\|D_k\|^2}{n_k}$$

Coupling metrics

External metrics estimate the level of coupling between clusters. One way of defining the total coupling is to calculate the size weighted sum of similarity to the centroid of the entire collection:

$$E_1 = \sum_{k=1}^{K} n_k \frac{m_k^T m}{\|m_k\|\|m\|} = \sum_{k=1}^{K} n_k \frac{D_k^T D}{\|D_k\|\|D\|}$$
$$\propto \sum_{k=1}^{K} n_k \frac{D_k^T D}{\|D_k\|}$$

Another commonly used coupling measurement is computed as the average pair-wise similarity between cluster centroids:

$$E_2 = \frac{1}{K^2} \sum_{m_i, m_j} \frac{m_i^T m_j}{\|m_i^T\|\|m_j\|} = \frac{1}{K^2} \sum_{D_i, D_j} \frac{D_i^T D_j}{\|D_i^T\|\|D_j\|}$$

A recent work by Kulkarni discussed the clustering stopping criteria in the bisecting clustering algorithm [Kulkarni06]. Her method focused on the study on the curve of the objective function I_1 (called *cf* in the paper). Three metrics PK1(m) which transforms the metric score into a normalized z-score, PK2(m) which measures the ratio of two consecutive scores, and PK3(m) which normalizes the score by the sum of scores from adjacent steps , are shown below:,

$$PK1(m) = \frac{\text{cf(m)} - \text{mean(cf[1..max])}}{\text{std(cf[1..max])}}, PK2(m) =$$
$$cf(m)/cf(m-1), PK3(m) = \frac{2cf(m)}{[cf(m-1)+cf(m+1)]}$$

were considered to decide whether granularity m is an optimal stopping point. The limitation of these metrics is that they only consider cohesion of the clusters, but ignore the coupling between the clusters. This is problematic because coupling and cohesion tend to trade-off against each other. Some widely used metrics that take into account both cohesion and coupling are reviewed next.

Hybrid metrics

Hybrid metrics combine both internal and external metrics; in other words, they attempt to maximize cohesion while simultaneously minimizing coupling. Obviously the ratio of internal and external metrics can serve naturally as hybrid metrics, such as $\frac{I_1}{E_1}, \frac{I_1}{E_2}, \frac{I_2}{E_1}, \frac{I_2}{E_2}$. Other commonly used hybrid metrics are described as follows.

MinMaxCut [Zhao01]

$$MMC = \sum_{k=1}^{K} \frac{\sum_{x \in c_k, y \notin c_k} x^T y}{\sum_{x,y \in c_k} x^T y} = \sum_{k=1}^{K} \frac{D_k^T (D - D_k)}{\|D_k\|^2}$$
$$\propto \sum_{k=1}^{K} \frac{D_k^T D}{\|D_k\|^2}$$

This metric simply calculates the average of the coupling-cohesion ratio, and so lower scores represents better clustering.

Davies-Bouldin (DB) index [Davies79]
DB measures the goodness of a clustering by its average dispersion and cluster coupling. In a partition of n objects into K clusters, for all pairs of clusters c_j and c_k, the within-to-between cluster spread is defined as

$$R_{j,k} = \frac{e_j + e_k}{dist_{j,k}}$$

where e_j and e_k are the standard deviation of point-to-centroid distances for c_j and c_k respectively, and $dist_{j,k} = \min_{x \in c_j, y \in c_k} d(x,y)$. Then the spread for individual cluster c_k is defined as

$$R_k = \max\{R_{j,k}\}, j \neq k$$

Finally, the DB index for K-cluster clustering is

$$DB(K) = \frac{\sum_{k=1}^{K} R_k}{K}$$

Intuitively, DB index considers the average distance of clusters to their nearest clusters respectively, therefore the smaller DB (K) indicates the better clustering.

Dunn's index [Dunn74]
This metric is built on the notion of $dist_{j,k}$ which was just defined, and also cluster diameter $diam_k = \max_{x,y \in c_k} d(x,y)$. It attempts to capture both the mutual distance between clusters and the inner span of a cluster simultaneously by using the formula:

$$\min_{1 \le i \le K-1} \{ \min_{1 \le j \le K} \{ \frac{dist_{j,i}}{\max_{1 \le k \le K} diam_k} \} \}$$

The larger Dunn (K) score is an indicator of a better clustering.

Hubert's Γ statistic [Halkidi01]

This metric measures the quality of clustering by considering the correlation between the partitioning and the original proximity matrix. Denoting the proximity matrix as **X** = [x_{ij}] and cluster labeling matrix **Y** = [y_{ij}], where y_{ij}=1 when requirement i and j are in the same cluster, and y_{ij}=0 otherwise, Hubert Γ statistic is defined as the point correlation between **X** and **Y**

$$\Gamma = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} x_{i,j} y_{i,j}$$

A normalized Hubert's Γ statistic can also be defined as:

$$\Gamma = (\frac{1}{M}) \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} [x_{i,j} - m_x][y_{i,j} - m_y] / s_x s_y$$

where $M = n(n-1)/2$, and m_x, m_y, s_x, s_y are the mean and standard deviation of two matrices respectively. A direct application of Pearson's linear correlation, means that the normalized Hubert Γ is always between -1 and 1. Unusually large absolute values of Γ suggest that two matrices agree with each other. But since the index increases monotonically as K increases, one can determine the optimal clustering granularity by identifying significant turning points.

As an example, of the Hubert index applied to the IBS data set, the scores of these metrics, with a small modification that uses correlation instead of Euclidean distance to measure similarity, are plotted in Figure 2.7 at successive values of K.

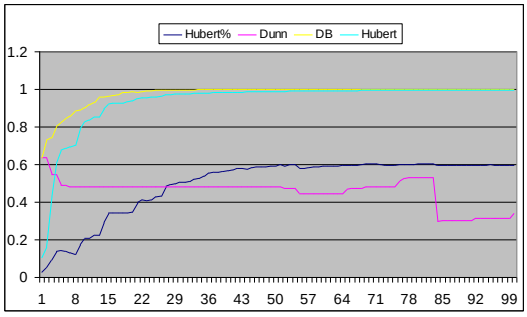

Figure 2.7 Score curves of DB, Dunn, Hubert, and normalized Hubert for IBS

Three main problems make the direct application of these metrics in requirements clustering dubious:

(1) They require substantial computation. Most of them have time complexity of $O(N^2)$.

(2) They usually return different answers and it is not clear which one is best. For example, in the Figure 2.7, the knees or elbows for Hubert, DB, Dunn, and Normalized Hubert are 6, 3, 4, and 5 respectively. As IBS is a relatively small dataset, these differences represent significantly different granularity strategies. Additional factors are therefore needed to select the optimal granularity from these solutions

(3) They do not take into account the purposes of particular clustering tasks which place additional constraints on the granularity. For example, if the 214 requirements of IBS data are divided into 10 clusters, the average cluster size will be 22. Although in theory, this granularity represents a best fit to the data, when used in tasks for which clusters are used directly by humans, 22 requirements per cluster is not ideal and would be hard for a human analyst to work with.

2.10 Correlation measure of partitions

Correlation metrics between partitions are necessary for validating and comparing clusters against a priori clustering, often an answer set that has been manually produced and scrutinized. Three types of correlation metrics exist in the literature of cluster analysis. They include metrics based on binary vector comparison, those based on information theory, and finally those based on retrieval performance evaluation.

Binary vector comparison

In a clustering, for all artifact pairs $x, y \in D$, an indicator is set to 1 if the pair belongs together and 0 otherwise. Thus in two clusterings, there are 4 possible combinations: 11, 00, 10, and 01, as shown in the table below:

	1	**0**
1	a	B
0	c	D

Three commonly used basic correlation coefficients known as Rand, Jaccard, and Folkes and Mallows index have been derived from this confusion table:

$$Rand = \frac{a+d}{a+b+c+d}$$

$$Jaccard = \frac{a}{a+b+c}$$

$$FM = \sqrt{m_1 m_2} = \sqrt{\left(\frac{a}{a+b}\right)\left(\frac{a}{a+c}\right)}$$

Some modifications are proposed, most of which apply certain weightings on the "a" in the Jaccard index, such as:

$$Dice = \frac{2a}{2a+b+c}$$

Intuitively, Rand takes into account both commonalities, whereas Jaccard focuses on the togetherness. In practice, these two metrics normally differ significantly: Rand index is too high, and Jaccard index is too low, and unfortunately no great guidelines exist in index choice.

One problem with Rand index is that its expected value for two random partitions does not take a constant value. The adjusted Rand index assumes the generalized hyper-geometric distribution as the model of randomness, i.e., the two partitions are picked at random such that the number of objects in the classes and clusters are fixed. With the extended confusion table shown below:

U/V	v_1	v_2	….	v_C	*Sums*
u_1	n_{11}	n_{12}	…	n_{1C}	$n_{1.}$
u_2	n_{21}	n_{22}	…	n_{2C}	$n_{2.}$
….	…	…	…	…	…
u_R	n_{R1}	n_{R2}	…	n_{RC}	$n_{R.}$
Sums	$n_{.1}$	$n_{.2}$	…	$n_{.C}$	$n_{..}=n$

and the adjusted Rand index [Hubert85] as:

$$Rand_a = \frac{\sum_{i,j}\binom{n_{ij}}{2} - \left[\sum_i \binom{n_{i.}}{2}\sum_j \binom{n_{.j}}{2}\right]/\binom{n}{2}}{\frac{1}{2}\left[\sum_i \binom{n_{i.}}{2} + \sum_j \binom{n_{.j}}{2}\right] - \left[\sum_i \binom{n_{i.}}{2}\sum_j \binom{n_{.j}}{2}\right]/\binom{n}{2}}$$

where

$$E\left[\sum_{i,j}\binom{n_{ij}}{2}\right] = \left[\sum_i \binom{n_{i.}}{2}\sum_j \binom{n_{.j}}{2}\right]/\binom{n}{2}$$

Typically the adjusted Rand index is much lower than the Rand index.

Information theoretical measures

Unlike binary vector comparison, which makes a hard pair-wise examination of two partitions, information theoretical measures the extent to which knowledge of one partition reduces uncertainty of the other. The agreement between two partitions P^a and P^b is expressed by the mutual information:

$$I(P^a, P^b) = \sum_{X \in P^a} \sum_{Y \in P^b} P(X,Y) \log \frac{P(X,Y)}{P(X)P(Y)}$$
$$= \sum_{i=1}^{k_a} \sum_{j=1}^{k_b} \frac{n_{ij}^{ab}}{n} \log \left(\frac{\frac{n_{ij}^{ab}}{n}}{\frac{n_i^a}{n} * \frac{n_j^b}{n}} \right)$$

where k_a and k_b are the cluster numbers of two partitions, n is the total number of artifacts, n_{ij}^{ab} is the number of shared artifacts in cluster a of clustering P^a and cluster b of clustering P^b (similar explanation to $\frac{n_i^a}{n}$ and $\frac{n_j^b}{n}$). To make the value bounded between 0 and 1, the normalization can be added by arithmetic [Fred05] or geometric average [Strehl03]:

$$NMI_a(P^a, P^b) = \frac{I(P^a, P^b)}{H(P^a)H(P^b)/2}$$

$$NMI_g(P^a, P^b) = \frac{I(P^a, P^b)}{\sqrt{H(P^a)H(P^b)}}$$

where *H(P)* is the entropy of a clustering $H(P) = -\sum_{j=1}^{k} \frac{n_i}{n} \log\left(\frac{n_i}{n}\right)$.

Maximum F-measure

This metric considers the agreement between two clusterings as a retrieval evaluation, for c_i^a and c_j^b, for which the recall, precision, and F-measure are defined as:

$$recall(i,j) = \frac{n_{ij}^{ab}}{n_j^b} \qquad precision(i,j) = \frac{n_{ij}^{ab}}{n_i^a}$$

$$F(i,j) = 2 * \frac{recall(i,j)precision(i,j)}{recall(i,j) + precision(i,j)}$$

$$FM = \sum_j \frac{n_j^b}{n} \max_i F(i,j)$$

Since NMI and F-measure have monotonous dependency on the cluster number and cluster size, they are typically used when two clusterings have comparable granularity.

2.11 Related work on requirement clustering

Despite the large body of literature on clustering, there has not yet been a substantial body of research focused on the clustering of requirements. This section discusses the available literature discussing clustering in requirements engineering.

Hsia et al [Hsia88, Hsia96] realized that although functional decomposition of design is mature, it is hard to map these functional parts onto customer-recognizable components. They therefore proposed the idea of decomposing requirements into a certain number of useful, usable, and semi-independent partitions that would facilitate incremental delivery (ID). The proximity matrix is constructed indirectly from the references requirements make to a set of system components; the clustering algorithm is a simplified hierarchical clustering technique in which requirements are segmented by continuous application of a series of proximity thresholds. This approach is reasonable in their example because the size of the requirements set is not large. However, they did not provide a convincing method or empirical evidence to validate their choice of clustering methods. Yaung [Yaung92] has the similar motivation and applies hierarchical clustering to explore the analogy between design modularity and requirements modularity; but again no rigorous evaluation of experimental results is presented.

Chen et al. [Chen05] used requirements clustering to automatically construct feature models for software product line analysis. They calculated the proximity between requirements by their various access modes to system resources, constructed a graph whose edge weights were based on the proximities, and utilized an iterative graph-splitting approach to cluster the requirements. They evaluated the individual cluster quality using an independency metric (IM), which is a graph theoretical metric that computes the ratio of the sum of outer edge weights over the sum of inner edges weights.

These limited studies have focused on very specific and unique clustering applications, but have not addressed the challenges described in Chapter 1, which are critical to successful clustering of real-life large scale requirement repositories. For example, most of the studies use trivial-sized data sets for concept proving, picked a clustering algorithm without empirical evaluation to compare different techniques, and did not address the issue of comprehensive cluster validation. Motivated by the flourish of clustering research in many other areas and the necessity of introducing robust automated methods to deal with large requirement collections, the author has done some earlier work to investigate the use of traditional term-based clustering algorithms within the requirements domain.

2.12 Author's earlier work using crisp document clustering methods

Duan et al [Duan07b] described a process for using traditional clustering algorithms and validation metrics to support automated tracing. Clustering algorithms are used to organize the candidate links in ways which would be more intuitive to the analyst, and would facilitate the analyst's task of evaluating the correctness of each link.

Three clustering algorithms of average link agglomerative hierarchical, K-means, and bisecting 2-means hierarchical, were evaluated. Two validation metrics, Hubert index and CC, were used to determine cluster granularity. Based on the observation that these two metrics did not return results that fully accommodated the tasks the clusters were intended to support, a new metric, named theme metric, along with a heuristic that constrained the average cluster size, was proposed to achieve optimal cluster granularity.

2.12.1 Data sets in these prior experiments

The three datasets included in the experiment were the Ice Breaker System (IBS), Event Based Traceability (EBT), and Public Health Watcher (PHW).

IBS was initially described in [Robertson99] and enhanced with requirements mined from documents obtained from the public work departments of Charlotte, Colorado; Greeley, Colorado; and the Region of Peel, Ontario. IBS manages de-icing services to prevent ice formation on roads. It receives inputs from a series of weather stations and road sensors within a specified district, and uses this information to forecast freezing conditions and manage dispersion of de-icing materials. The system consists of 180 functional requirements, 72 classes, and 18 packages.

EBT, which was initially developed at the International Center for Software Engineering at the University of Illinois at Chicago, provides a dynamic traceability infrastructure based on the publish-subscribe scheme for maintaining artifacts during long-term change maintenance. It is composed of 54 requirements, 60 classes, and 8 packages.

Finally PHW represents a health complaint system developed to improve the quality of the services provided by health care institutions [Soares02]. The specification is mainly structured as use cases, and in this paper, each use-case step is extracted as a requirement, resulting in 241 requirements.

2.12.2 Clustering algorithms

For completeness, the three algorithms are briefly described as follows.

Average-link agglomerative hierarchical clustering (AHC)

Initialization Each requirement is assigned to an individual cluster, and the similarities between requirements are calculated as the similarities between clusters.

Iterations - Merge the most similar pair of clusters
- Calculate the similarity between the new cluster c_i and each existing cluster c_j by

$$S(c_i, c_j) = \frac{1}{|c_i \| c_j|} \sum_{a_1 \in c_i, a_2 \in c_j} s_c(a_1, a_2)$$

Termination The target granularity K is met.

K-means clustering

Initialization Define a set of centroids $M = \{m_1, m_2, \ldots, m_K\}$ for clusters $\{c_1, c_2, \ldots, c_K\}$. To avoid poor quality clusters, pick K artifacts from D to serve as initial centroids such that these artifacts exhibit as little mutual similarity as possible.

Iterations - For each artifact a_i, compute the similarity scores between a_i and each centroid. Identify the centroid m_j that is most similar to a_i, and assign or reassign a_i to cluster c_j.
- For each cluster c_j, recompute the newly formed center m_j as the mean of all the artifacts contained in c_j.

Termination No membership reassignment occurs during an iteration.

Bisecting Hierarchical Clustering (BHC)

The bisecting Hierarchical clustering algorithm relies on K-means (K=2 specifically) clustering to consecutively bisect a larger cluster into two smaller ones. It runs as follows:

Initialization Assign all the artifacts to a single cluster

Iterations - For each c_i in the present clustering C, bisect it using 2-means clustering and

then compute the score of the objective function E over the resulting clusters
- Select the cluster c_p that exhibits the highest E score. For this cluster, commit the splitting of c_p, by removing c_p, and adding the two new clusters into the clustering.

Termination The target granularity K is met.

Comparison

In time complexity, both K-means and BHC exhibit a time complexity of O(N), although BHC is usually slower than K-means since its complexity has a much larger constant. AHC has a $O(N^2)$ complexity, thus is much slower when clustering large scale data sets.

The biggest advantage of bisecting clustering over K-means clustering is that it tends to produce relatively uniformly sized clusters, a nice property especially useful in supporting applications where the average size of the clusters is relatively small. Actually, through the empirical comparison with answer set by using the partition comparison metrics just introduced, bisecting clustering produces better document clusters than K-means most of the time, as already demonstrated by Steinbach et al. [Steinbach00] and Zhao [Zhao02]. However, this balance cluster assumption in bisecting algorithms may embed outliers in clusters, therefore reducing the cohesion of the clustering.

2.12.3 Granularity determination

Hubert index and CC, the ratio between intra-cluster cohesion and inter-cluster coupling, were adopted to determine the right number of clusters for the bisecting algorithm. Figure 2.8 and 2.9 show their score curves and highlight the significant turning points for 3 data sets.

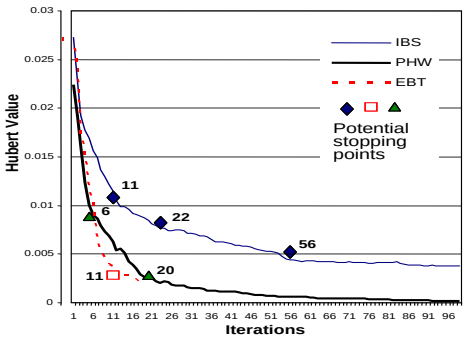

Figure 2.8 Hubert's index against three datasets

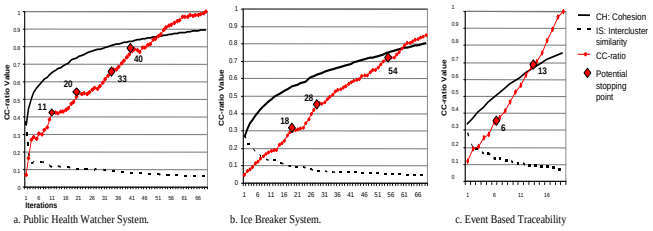

Figure 2.9 CC index against three datasets

The first problem with these two metrics is that they return very different answers to the granularity question. The second problem is that human evaluation of the generated clusters suggested that several of the resulting clusters contained multiple meaningful topics or themes, and that it was not the case that every requirement in a cluster belonged to a single, and clearly identifiable theme. Furthermore, many of the resulting clusters were not cohesive enough for practical use by human analysts. To locate the stopping point that produced truly cohesive clusters and to measure the quality of the generated clusters i.e. clusters with a single dominant theme, a set of theme-based metrics were designed.

The basic idea of the theme-based metric is that a cohesive cluster should have only one dominant theme.

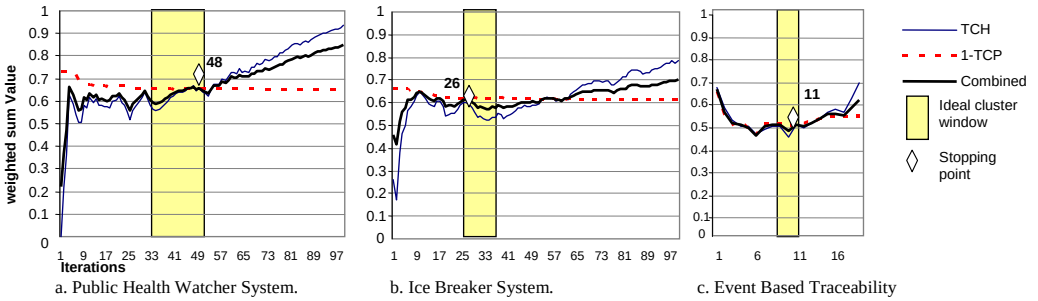

Figure 2.10 Theme cohesion and coupling for three datasets showing ideal cluster window.

This dominant theme within a cluster is represented by a

set of dominant terms $Dt = \{dt_1, dt_2, dt_3, \ldots dt_m\}$, each of which satisfies $\frac{N_R(dt_i)}{N_R} \geq p$, where $N_R(dt_i)$ is the number of requirements in the cluster containing term dt_i, N_R is the total number of artifacts in the cluster, and $0 \leq p \leq 1$ is a threshold. Based on the definition of dominant terms, theme cohesion (TCH) is computed as $TCH = \frac{N_R(I(Dt) \geq \alpha)}{N_R}$ where $N_R(I(Dt) \geq \alpha)$ represents the number of requirements containing at least percentage α of all dominant terms, and theme coupling (TCP) is the normalized correlation between dominant terms $TCP = \frac{\sum_{c_i, c_j \in C} [Dt_i \cdot Dt_j / |Dt_i \| Dt_j|]}{k(k-1)/2}$. Then the theme metric (TM) is defined as the weighted combination of TCH and TCP: $TM = \alpha TCH + (1-\alpha)TCP$. For the practical use of TM in granularity determination of bisecting clustering, a target range has to be first estimated since the TM is not compatible with the objective function that guides the bisecting algorithm. This range is based on George Miller's research which showed that an average person can handle around seven chunks, plus or minus five, of information in working memory at a time [Miller56]. Our approach therefore targeted an average cluster size within the range of seven to twelve, and used this to compute the target number of clusters needed. The granularity that optimized theme cohesion within the target number of clusters, as shown by a maximum on the TM curve, in Figure 2.10, was then selected.

3. Probabilistic topic-based modeling of textual artifacts

The clustering of documents or requirements can also be studied from the perspective of model learning. In that view, the artifacts in a cluster are similar in that they conform to the same parametric distribution, and accordingly, the clustering is a process of identifying those distributions and classifying artifacts according to their most related distributions. Because these approaches have a sound probabilistic interpretation, adapt flexibly to data of different characteristics, and are empirically proven to exhibit good performance in both supervised and un-supervised learning, they are becoming prevalent in machine learning and information retrieval. Topic-based modeling is one example of such state-of-the-art document modeling methods. Intuitively, a topic is represented by a set of terms that are most closely related to that topic. In the language of statistics, a topic z corresponds to a distribution of terms, in most cases a multinomial
distribution $P(w_i|z), i = 1,2, \ldots, T, and \sum_i w_i = 1$. A number of topic models exist in literature, differentiated by their assumption on the relationship between documents and topics. Some of them, such as multinomial mixture and Dirichlet compound multinomial mixture, assume that a document is associated with only one topic, while others, such as PLSA and LDA, assume that a document can be a mixture of multiple topics. The latter has been demonstrated to be more flexible, performing better in document modeling, document categorization, and collaborative filtering [Blei03], etc. This chapter provides a survey of a number of these models and the inference of them.

3.1 Maximum likelihood estimates of mixture model and EM framework

Before further discussing the modeling of topics, the basic finite mixture model and some techniques for estimating the parameters of mixture models are reviewed as the former is the skeleton of almost all the probabilistic latent topic models and the latter is indispensible in model inference.

Maximum likelihood estimates (MLE)

Among various estimation techniques, MLE is the most widely used for its simplicity. The basic idea is to choose the parameters that maximize the likelihood function of the samples. Suppose n i.i.d. (identically and independently distributed) samples $X = \{\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_n}\}$, let $L(\boldsymbol{\theta}) = \ln p(X|\boldsymbol{\theta}) = \ln \prod_{k=1}^{n} p(\boldsymbol{x_k}|\boldsymbol{\theta})$ be the log-likelihood function of the samples, then parameter $\boldsymbol{\theta}$ is estimated as $\widehat{\boldsymbol{\theta}} = argmax_{\theta} L(\boldsymbol{\theta})$. Numerically the MLE is typically given by the solution of the linear equation $\nabla_{\boldsymbol{\theta}} L = 0$. The MLE solution could represent a true global maximum, a local maximum or minimum, or an inflection point of $L(\boldsymbol{\theta})$.

As an example, in cases where the samples conform to a multivariate Gaussian distribution $p(\mathbf{x}) \sim N(\boldsymbol{\mu}, \sum)$

$$p(\boldsymbol{x}) = \frac{1}{(2\pi)^{\frac{d}{2}}\sqrt{|\boldsymbol{\Sigma}|}} exp\left\{-\left(\frac{1}{2}\right)(\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right\}$$

where $\boldsymbol{\mu} = E(\boldsymbol{x}) = \int \boldsymbol{x} p(\boldsymbol{x}) d\boldsymbol{x}$ and $\Sigma = E[(\boldsymbol{x} - \boldsymbol{\mu x})(\boldsymbol{x} - \boldsymbol{\mu x})^T] = \int (\boldsymbol{x} - \boldsymbol{\mu x})(\boldsymbol{x} - \boldsymbol{\mu x})^T p(\boldsymbol{x}) d\boldsymbol{x}$, the MLE of $\boldsymbol{\theta}$ calculated by the vanishing gradient of L are:

$$\hat{\boldsymbol{\mu}} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{x}_i$$

$$\hat{\Sigma} = \frac{1}{n}\sum_{i=1}^{n} (\boldsymbol{x}_i - \hat{\boldsymbol{\mu}})(\boldsymbol{x}_i - \hat{\boldsymbol{\mu}})^T$$

Expectation-Maximization (EM)

The MLE of models with simple closed form of likelihood such as Gaussian can be directly derived, while this is more complicated for other models such as multinomial mixture models (see the next section). A particularly important method of estimating complicated models is the EM framework. EM was originally proposed as a computational framework to cope with the problem of missing data [Dempster77], and it has also been applied in problem domains where, even though the observable data are complete, the problem can be reformulated into one with missing latent variables. This section introduces EM in its most general form.

First, by Jensen's inequality, if $x_1, x_2, \dots, x_n \in I$ and $\lambda_1, \lambda_2, \dots, \lambda_n \geq 0$ with $\sum_i \lambda_i = 1$, then

$$\ln\left(\sum_i \lambda_i \boldsymbol{x}_i\right) \geq \lambda_i \left(\sum_i \ln(\boldsymbol{x}_i)\right)$$

EM tries to maximize the difference between samples' likelihoods of two iterations:

$$L(\boldsymbol{\theta}) - L(\boldsymbol{\theta_n}) = \ln P(\boldsymbol{X}|\boldsymbol{\theta}) - \ln P(\boldsymbol{X}|\boldsymbol{\theta_n})$$

where $L(\boldsymbol{\theta})$ is the likelihood of desired parameters $\boldsymbol{\theta}$, and $L(\boldsymbol{\theta_n})$ is the likelihood of desired parameters $\boldsymbol{\theta_n}$. Denoting by $\boldsymbol{Z}$ the unobserved or missing variables, likelihood can be un-marginalized as

$$P(\boldsymbol{X}|\boldsymbol{\theta}) = \sum_z P(\boldsymbol{X}|z, \boldsymbol{\theta}) P(z|\boldsymbol{\theta})$$

then the difference of two likelihoods is:

$$L(\boldsymbol{\theta}) - L(\boldsymbol{\theta_n}) = \ln P(\boldsymbol{X}|\boldsymbol{\theta}) - \ln P(\boldsymbol{X}|\boldsymbol{\theta_n}) = \ln \sum_z P(\boldsymbol{X}|z, \boldsymbol{\theta}) P(z|\boldsymbol{\theta}) - \ln P(\boldsymbol{X}|\boldsymbol{\theta_n}) \quad (2)$$

By introducing $P(z|\boldsymbol{X}, \boldsymbol{\theta_n})$,

$$\ln \sum_z P(\boldsymbol{X}|z, \boldsymbol{\theta}) P(z|\boldsymbol{\theta}) = \ln \sum_z \left(P(z|\boldsymbol{X}, \boldsymbol{\theta_n}) \frac{P(\boldsymbol{X}|z, \boldsymbol{\theta}) P(z|\boldsymbol{\theta})}{P(z|\boldsymbol{X}, \boldsymbol{\theta_n})} \right) \geq \sum_z P(z|\boldsymbol{X}, \boldsymbol{\theta_n}) \ln \left[\frac{P(\boldsymbol{X}|z, \boldsymbol{\theta}) P(z|\boldsymbol{\theta})}{P(z|\boldsymbol{X}, \boldsymbol{\theta_n})} \right]$$

and writing $\ln P(\boldsymbol{X}|\boldsymbol{\theta_n}) = \ln \sum_z \left(P(z|\boldsymbol{X}, \boldsymbol{\theta_n}) P(\boldsymbol{X}|\boldsymbol{\theta_n}) \right)$

$$(2) = \sum_z P(z|\boldsymbol{X}, \boldsymbol{\theta_n}) \ln \left[\frac{P(X|z, \boldsymbol{\theta}) P(z|\boldsymbol{\theta})}{P(z|\boldsymbol{X}, \boldsymbol{\theta_n})} \right] - \ln \sum_z \left(P(z|\boldsymbol{X}, \boldsymbol{\theta_n}) P(\boldsymbol{X}|\boldsymbol{\theta_n}) \right)$$

$$= \sum_z P(z|\boldsymbol{X}, \boldsymbol{\theta_n}) \ln \left[\frac{P(\boldsymbol{X}|z, \boldsymbol{\theta}) P(z|\boldsymbol{\theta})}{P(z|\boldsymbol{X}, \boldsymbol{\theta_n}) P(\boldsymbol{X}|\boldsymbol{\theta_n})} \right] = \Delta(\boldsymbol{\theta}|\boldsymbol{\theta_n})$$

Any $\boldsymbol{\theta}$ that increases $\Delta(\boldsymbol{\theta}|\boldsymbol{\theta_n})$ in turn increases $L(\boldsymbol{\theta})$, the parameter of which is denoted as $\boldsymbol{\theta_{n+1}}$. So by dropping the terms that are constant with respect to $\boldsymbol{\theta}$, :

$$\begin{aligned}
&\boldsymbol{\theta_{n+1}} \\
&= argmax_{\boldsymbol{\theta}} \left\{ \sum_z P(z|\boldsymbol{X}, \boldsymbol{\theta_n}) \ln P(\boldsymbol{X}|z, \boldsymbol{\theta}) P(z|\boldsymbol{\theta}) \right\} \\
&= argmax_{\boldsymbol{\theta}} \left\{ \sum_z P(z|\boldsymbol{X}, \boldsymbol{\theta_n}) \ln \frac{P(z, \boldsymbol{X}, \boldsymbol{\theta})}{P(z, \boldsymbol{\theta})} \frac{P(z, \boldsymbol{\theta})}{P(\boldsymbol{\theta})} \right\} \\
&= argmax_{\boldsymbol{\theta}} \left\{ \sum_z P(z|\boldsymbol{X}, \boldsymbol{\theta_n}) \ln P(\boldsymbol{X}, z|\boldsymbol{\theta}) \right\} \\
&= argmax_{\boldsymbol{\theta}} \left\{ E_{(z|\boldsymbol{X}, \boldsymbol{\theta_n})} \ln P(\boldsymbol{X}, z|\boldsymbol{\theta}) \right\}
\end{aligned}$$

The intuitive way to understand this is: now that the values of $\mathbf{X}$ and $\boldsymbol{\theta_n}$ are known and $\boldsymbol{\theta}$ is the parameter to be adjusted, then distribution of $\mathbf{Z}$ is governed by $p(z|\mathbf{X}, \boldsymbol{\theta_n})$; with this Z distribution, the next value of θ will be the one that maximized expected value of $\ln p(z|\mathbf{X}, \boldsymbol{\theta_n})$ w.r.t. $\mathbf{Z}$.

Based on the derivation above, the two iterative steps of EM are:

- E-step: Determine the conditional expectation $E_{(z|\boldsymbol{X}, \boldsymbol{\theta_n})} \ln P(\boldsymbol{X}, z|\boldsymbol{\theta})$
- M-step: Maximize the expression with respect to $\boldsymbol{\theta}$

So EM provides a framework for parameter estimate while taking into account the unobserved or missing data $\mathbf{Z}$.

3.2 Finite mixture model and unsupervised learning of mixture model

The finite mixture model assumes the probability density of a sample as the weighted mixture of a certain number of component densities, namely,

$$p(\boldsymbol{x}|\boldsymbol{\theta}) = \sum_{i=1}^{K} \pi_i p(\boldsymbol{x}|\omega_i, \boldsymbol{\theta_i})$$

where K is the number of components, $p(\mathbf{x}|\omega_i, \boldsymbol{\theta_i})$ is the p.d.f. of the component i, π_i is the prior of the component i, and $\boldsymbol{\theta}$ includes all the parameters of Kcomponents.

By ML,

$$l = \sum_{k=1}^{n} \ln p(\boldsymbol{x_k}|\boldsymbol{\theta}),$$

In practice, since the log of the sum of densities is not easy to handle, EM is typically used to simplify the learning of the models [McLachlan00]. Let Z_i be a random discrete variable with value among 1, 2 , …, and K, namely an indicator which component actually issues the sample x_i. then the likelihood can be simplified as

$$l = \sum_{k=1}^{n} \left[\ln \pi_{z_k} p(\boldsymbol{x_k}|\theta z_k)\right]$$

According to the EM framework just introduced, the E-step (as defined in section 4.1.2) will be

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta^g}) = E_{(\boldsymbol{Z}|\boldsymbol{X},\boldsymbol{\theta^g})} P(\boldsymbol{X}, \boldsymbol{Z}|\boldsymbol{\theta})$$

$$= \sum_{z_1,z_2,\ldots,z_n=1,2,\ldots,K} \left(\sum_{k=1}^{n} \left[\ln \pi_{z_k} p(\boldsymbol{x_k}|\theta z_k)\right] \prod_{j=1}^{n} p(z_k|\boldsymbol{x_k}, \boldsymbol{\theta^g}) \right)$$

$$= \sum_{g=1}^{K} \sum_{k=1}^{n} \left[\ln \pi_g p(\boldsymbol{x_k}|\theta z_g)\right] p(g|\boldsymbol{x_k}; \boldsymbol{\theta^g})$$

The parameters can then be calculated in the M-step where $\nabla_{\theta_i} Q(\boldsymbol{\theta}, \boldsymbol{\theta^g}) = 0$ and $\nabla_{\pi_i} Q(\boldsymbol{\theta}, \boldsymbol{\theta^g}) = 0$. It has been proven that the iterative calculation of parameters by EM never decreases the likelihood, so in many cases an optimal estimate can be achieved once certain difference tolerances of the likelihood difference is met.

3.3 Related work on un-supervised learning of finite mixture document model

Fraley et al discussed the use of a mixture model in clustering multivariant normal data and Bayesian Information Criterion (BIC) in model selection [Fraley98, Fraley02]. The mixture model of a sample is:

$$p(\boldsymbol{x}) = \sum_{i=1}^{K} \pi_i p(\boldsymbol{x}|\omega_i, \theta_i)$$

where a Gaussian distribution is assumed: $p(\boldsymbol{x}|\omega_i, \theta_i) = p(\boldsymbol{x}) = \frac{1}{(2\pi)^{\frac{d}{2}}\sqrt{|\boldsymbol{\Sigma}|}} exp\left\{-\left(\frac{1}{2}\right)(\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right\}$. In addition, they modeled the noise and outliers as a constant rate Poisson process, resulting in the mixture log likelihood of

$$l = \sum_{k=1}^{n} \ln\left(\frac{\tau_0}{V} + \sum_{i=1}^{K} \pi_i p(\boldsymbol{x}|\omega_i, \theta_i)\right)$$

where V is the hypervolume of the data region. In this modeling, if an observation is noisy, it contributes 1/V to the likelihood, and a normal mixture likelihood otherwise. Their experiment on Diabetes diagnosis and minefield detection suggested that for their data, the Gaussian mixture model clustering outperformed K-means and single link hierarchical clustering, whether noise was present or not.

For modeling documents where supposedly a topic is in fact a multinomial distribution over terms, mixture multinomial models are proposed for supervised and un-supervised learning [Nigam00, Rigouste07], where the probability of document **x** of length n is:

$$p(\boldsymbol{x}|\boldsymbol{\theta}) = \frac{n!}{\prod_w x_w!} \prod_w \theta_w^{x_w}$$

Another type of mixture model, Dirichlet compound multinomial (DCM) model and EDCM model, addressed in [Madsen05, Elkan06], adds one more degree of freedom by modeling the generation of a document in a Polya process which first performs a Dirichlet drawing:

$$p(\boldsymbol{\theta}|\boldsymbol{\alpha}) = \frac{\Gamma(\sum_{w=1}^{W} \alpha_w)}{\sum_{w=1}^{W} \Gamma(\alpha_w)} \prod_{w=1}^{W} \theta_w^{\alpha_w - 1}, where \sum_w \theta_w = 1$$

and then a multinomial drawing:

$$p(\boldsymbol{x}|\boldsymbol{\theta}) = \frac{n!}{\prod_w x_w!}\prod_w \theta_w^{x_w}$$

Integrating the parameter $\boldsymbol{\theta}$, a DCM

$$p(x|\alpha) = \int_\theta p(x|\theta)p(\theta|\alpha)d\theta$$
$$= \frac{n!}{\prod_{w=1}^W x_w!}\frac{\Gamma(\sum_{w=1}^W \alpha_w)}{\Gamma(\sum_{w=1}^W (x_w+\alpha_w))}\prod_{w=1}^W \frac{\Gamma(x_w+\alpha_w)}{\Gamma(\alpha_w)}$$
$$= \frac{n!}{\prod_{w=1}^W x_w!}\frac{\Gamma(s)}{\Gamma(s+n)}\prod_{w=1}^W \frac{\Gamma(x_w+\alpha_w)}{\Gamma(\alpha_w)}$$

where n is the length of the artifact x and s is the sum of Dirichlet parameter vector. The DCM mixture model learned using EM is shown to outperform the multinomial mixture in clustering NIPS document sets [Elkan06].

A serious problem in these mixture models is that they all assume a document exhibits only a single dominant topic, which results in overfitting especially when the collection has insufficient samples [Blei03]. This limitation can only be overcome by assuming the existence of multiple topics in a document.

3.4 Topic-based modeling of documents

Based on the assumption that semantic information can be derived from a word-document co-occurrence matrix, the topic-based modeling methods claim that documents are composed of a mixture of topics, where a topic is a probability distribution over words. For example, in a requirements data set PCS, the topic of database construction will be mainly comprised of words such as database, server, backup, microsoft, configure, oracle, SQL. For the purpose of clustering, the desired outcome of topic models includes not only the topics, but also the topic distribution over documents, two sets of parameters denoted in $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$ here. Two widely used models, PLSA and LDA, will be described as follows.

3.4.1 Probabilistic LSA (PLSA)

Probabilistic LSA [Hoffman99] is a pioneer in probabilistic topic modeling of documents. Although strictly speaking not a generative model, it achieves a document decomposition and topic extraction with sound probabilistic interpretation. A description and fitting of the model using EM [Hoffman99] is now described.

The model used by PLSA to model documents is called the aspect model, a latent variable model which associates an un-observed topic $z \in Z = \{z_1, \dots, z_k\}$ with each observation of occurrence of a document d and a term w, whose joint probability can be written as:

$$P(d,w) = P(d)P(w|d) = P(d)\sum_{z\in Z} P(w|z)P(z|d)$$

One important assumption underlying this modeling is the **conditional independence** – the d and w are independent conditioned on the state of the associated latent variable. Putting $P(d)$ inside the summation leads to the following symmetric expression of joint probability:

$$P(d)\sum_{z\in Z} P(w|z)P(z|d) = \sum_{z\in Z} P(w|z)P(z,d)$$
$$= \sum_{z\in Z} P(d|z)P(w|z)P(z)$$

The distinction is illustrated in Figure 3.1.

By defining $\boldsymbol{U} = \left(P(d_i|z_k)\right)_{i,k}, \boldsymbol{V} = \left(P(w_j|z_k)\right)_{j,k}, and\ \boldsymbol{\Sigma} = diag\left(P(z_k)\right)_k$, the joint model P could be written as $\boldsymbol{P} = \boldsymbol{U\Sigma V^T}$. Although this factorization resembles the LSA, PLSA differs from LSA primarily in the objective function used to determine the optimal approximation. LSA uses Frobenius norm, which corresponds to an implicit additive Gaussian noise assumption on counts; in contrast, PLSA relies on the likelihood function of multinomial sampling, a well-defined probability distribution and factors that have a clear probabilistic meaning.

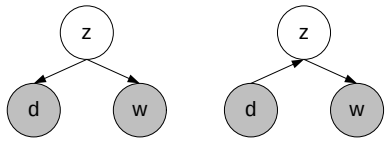

Figure 3.1 Graphical Model representation of aspect model in the symmetric (left) and asymmetric (right) parameterization.

Learning of the PLSA model using EM

PLSA uses EM to find the MLE of parameters, so it does not guarantee to find the global maximum. The joint probability and log likelihood are:

$$P(z, w_i, d_i) = P(z)P(w_i|z)P(d_i|z)$$

$$l = \log P(W, D) = \log\left(\prod_{i=1}^{n} P(w_i, d_i)\right)$$
$$= \sum_{i=1}^{n} \log\left(\sum_{z} P(z, w_i, d_i)\right)$$
$$= \sum_{i=1}^{n} \log \sum_{z} P(z)P(w_i|z)P(d_i|z)$$

Applying the EM estimate of the mixture model, letting the targeted parameters $\boldsymbol{\theta}$ be all $P(z)$, $P(w_i|z)$, and $P(d_i|z)$,

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^g)$$
$$= \sum_{z}\sum_{i=1}^{n} [\log(P(z)P(w_i|z)P(d_i|z))]\, P(z|w_i, d_i; \boldsymbol{\theta}^g)$$

The posterior

$$P(z|w_i, d_i; \boldsymbol{\theta}^g) = \frac{P(z; \boldsymbol{\theta}^g)P(w_i|z; \boldsymbol{\theta}^g)P(d_i|z; \boldsymbol{\theta}^g)}{P(w_i, d_i; \boldsymbol{\theta}^g)}$$
$$= \frac{P(z; \boldsymbol{\theta}^g)P(w_i|z; \boldsymbol{\theta}^g)P(d_i|z; \boldsymbol{\theta}^g)}{\sum_{z'} P(z'; \boldsymbol{\theta}^g)P(w_i|z'; \boldsymbol{\theta}^g)P(d_i|z'; \boldsymbol{\theta}^g)}$$

After the introduction of Lagrange multipliers and solving $\nabla_\theta Q(\theta, \theta^g)$, the iterative estimate of the parameters are:

$$P(z) = \frac{1}{N}\sum_{j=1}^{n} P(z|w_j, d_j)$$

$$P(w_i|z) = \phi_{iz} = \frac{\sum_{w_j = w_i} P(z|w_j, d_j))}{\sum_{j=1}^{n} P(z|w_j, d_j)}$$

$$P(d_i|z) = \frac{\sum_{d_j = d_i} P(z|w_j, d_j))}{\sum_{j=1}^{n} P(z|w_j, d_j)}$$

Among them, $P(w_i|z)$ represents the term distribution over topics **φ**. Further, **θ**, the topic distribution specific to a document d_i can be calculated using Bayes rule:

$$\theta_{zi} = P(z|d_i) = \frac{P(z|d_i)P(z)}{P(d_i)}$$

Having been demonstrated to outperform many other semantic methods such as LSA, PLSA has two weaknesses. First over-fitting and local optima estimation occurs in the learning of the model. Second, as it is not a generative model for documents, the likelihood of a new document **w** can only be represented heuristically by marginalizing over all the existing documents:

$$p(\boldsymbol{w}) = \sum_{d}\prod_{w}\sum_{z} p(w|z)p(z|d)p(d)$$

3.4.2 Latent Dirichlet Allocation (LDA)

LDA is a three-level hierarchical generative model for documents, constrained with two set of corpus parameters , Dirichlet priors $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, .., \alpha_c]^T$, and term distributions over topics $\beta = (\beta)_{ij}$, where $\beta_{ij} = p(w_i|z_j)$. A document represented by vector **w** of length W is generated in the following steps:

1. Draw a topic distribution **θ** from Dirichlet distribution Dir(α) with prior $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, .., \alpha_c]^T$
2. For each term position in **w**
   draw a topic z from Multinomial(**θ**)
   draw a word w from Multinomial(z, β)

The relationship between observed and latent variables is shown in the plate diagram below:

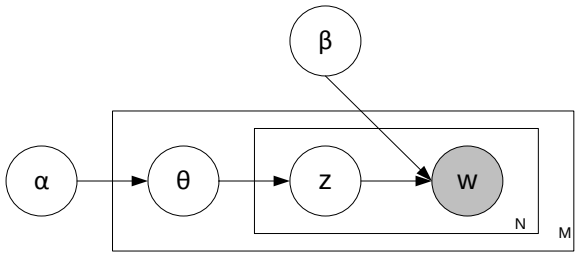

Figure 3.2 Graphical model representation of LDA

Therefore, the joint distribution of a document **w** with latent topics z under topic distribution θ is:

$$p(\boldsymbol{w}, \boldsymbol{z}, \boldsymbol{\theta}|\boldsymbol{\alpha}, \boldsymbol{\beta}) = p(\boldsymbol{\theta}|\boldsymbol{\alpha})\prod_{i=1}^{K} p(z_i|\boldsymbol{\theta})p(w_i|z_i, \boldsymbol{\beta})$$

By integrating over **θ** and summing over topics z, the above leads to the marginal distribution of a document:

$$p(\boldsymbol{w}|\boldsymbol{\alpha}, \boldsymbol{\beta}) = \int_{\theta} p(\boldsymbol{\theta}|\boldsymbol{\alpha})\prod_{i=1}^{N}\sum_{j=1}^{K} p(z_i|\boldsymbol{\theta})p(w_i|z_i, \boldsymbol{\beta})\, d\theta$$

Model learning using Gibbs Sampling

Since the form of document distribution is intractable, in Blei's original LDA paper [Blei03], Variational Bayesian

(VB) was used for model inference. The method described here is a Gibbs sampling based estimation method described in [Griffiths04, Steyvers07] as it's easy to implement, and competitive in speed and performance with other methods.

The Gibbs sampling LDA inference adds a Dirichlet prior on the term distributions over topics, denoted as β in [Blei03]. To unify the denotation in clustering which uses both PLSA and LDA, β here is switched to represent Dirichlet prior and **φ** is used for term distributions. The probabilistic model of LDA with Dirichlet prior is:

$$\theta \sim Dir(\alpha)$$

$$\phi \sim Dir(\beta)$$

$$z_i|\theta \sim Multinomial(\theta)$$

$$w_i|z_i, \phi \sim Multinomial(\phi_i)$$

And their graphical model plate is shown in Figure 3.3.

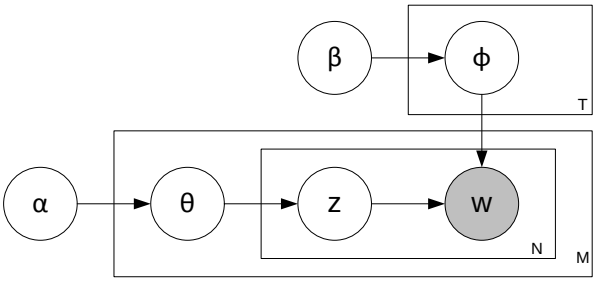

Figure 3.3 Graphical model representation of LDA with Dirichlet Prior on term distributions

Given a plausible assumption of uniform priors for α and β, since Dirichlet distributions $Dir(\alpha)$ and $Dir(\beta)$ are conjugate to the multinomial distributions $Multinomial(\theta)$ and $Multinomial(\phi_i)$, the distributions $p(\boldsymbol{z})$ and $p(\boldsymbol{w}|\boldsymbol{z})$ can be directly expressed in α and β, leading to a convenient expression of posterior $p(\boldsymbol{z}|\boldsymbol{w})$, that is:

$$p(\boldsymbol{w}|\boldsymbol{z}) = \left(\frac{\Gamma(W\beta)}{\Gamma(\beta)^W}\right)^T \prod_{j=1}^{T} \frac{\prod_w \Gamma(n_j^w + \beta)}{\Gamma(n_j + W\beta)}$$

$$p(\boldsymbol{z}) = \left(\frac{\Gamma(T\alpha)}{\Gamma(\alpha)^T}\right)^D \prod_{d=1}^{D} \frac{\prod_j \Gamma(n_j^d + \alpha)}{\Gamma(n_\cdot^d + T\alpha)}$$

and

$$p(\boldsymbol{z}|\boldsymbol{w}) = \frac{p(\boldsymbol{w}|\boldsymbol{z})p(\boldsymbol{z})}{\sum_z' p(\boldsymbol{w}|\boldsymbol{z}')p(\boldsymbol{z})}$$

Where W is the number of the terms, D is the number of artifacts, n_j^w is the number of times word w is assigned to topic j, and n_j^d is the number of times a word from document d is assigned to topic j. Then, a Gibbs sampling process is carried over this posterior distribution until a stable set of samples is obtained. Finally the statistics that are independent of individual topics can be computed by integrating across the whole set of samples, and **φ** and **θ** can be estimated using samples from the converged chain:

$$\phi_{wj} = \frac{n_j^w + \beta}{n_j + W\beta}$$

Theme # 1	
Term	**Score**
Email	0.042
Messag	0.027
Address	0.022
contact	0.015
associ	0.015
account	0.014
link	0.013
histori	0.012
automat	0.012
send	0.011

Theme # 2	
Term	**Score**
calendar	0.06
meet	0.048
abil	0.025
appoint	0.024
dai	0.021
event	0.021
schedul	0.02
dashlet	0.017
displai	0.016
time	0.016

Theme # 3	
Term	**Score**
campaign	0.066
target	0.055
list	0.041
email	0.037
send	0.019
contact	0.016
lead	0.014
mail	0.014
distribut	0.013
sent	0.011

Theme # 4	
Term	**Score**
print	0.033
territori	0.029
assign	0.028
team	0.022
pro	0.014
manag	0.01
option	0.01
record	0.009
sale	0.008
state	0.008

Theme # 5	
Term	**Score**
case	0.016
document	0.016
text	0.015
opportun	0.012
layout	0.011
lead	0.011
second	0.011
project	0.011
descript	0.01
search	0.009

Figure 3.4 A sample of themes extracted from SugarCRM feature requests showing top terms. Terms over a threshold >= 0.15 are shaded.

$$\theta_{jd} = \frac{n_j^d + \alpha}{n_\cdot^d + T\alpha}$$

It has been proven in [Girolami03] that PLSA is a *maximum a posteriori* estimated LDA model under a uniform Dirichlet prior, which is exactly the setting that the Gibbs sampling based inference takes. However, as they have different numerical inferences and those inferences highly rely on the characteristic of data, the performance must be evaluated within specific domain applications.

As an illustration of extracted topics from requirements using topic probabilistic models, Figure 3.4 shows five topics or themes from mining SUGAR data sets.

It can seen that not all the topics are equally meaningful and strong, so in general, when these algorithms are applied, the identified topics will be rigorously analyzed along with other parameters inferred from model fitting, producing sets of parameters that are used to derive the clustering by classifying each requirement according to the topics that it primarily contains. For comparison purposes, the popular fuzzy K-means algorithm will be implemented and compared with the topic-based clustering in the coverage of found topics.

4. Conclusion

The first part of this report extensively surveys the document clustering methods, and presents the experiments results using several popular heuristic-based crisp clustering algorithms on requirements. Based on the observation that typical requirements have terse representation and multiple topics, which causes the loss of significant topics, the second part proposes using probabilistic topic models, such as PLSA and LDA, to directly extract topics from requirements and then to derive clusters from significant topics. The future work will include the vigorous validation of topic-based clustering of requirements, investigating whether it can produce more cohesive clusters and wider range of topics to facilitate various requirement engineering tasks.

References

[Ambroise00] Ambroise, C., Seze, G., Badran, F., and Thiria, S. 2000. Hierarchical clustering of self-organizing maps for cloud classification. *Neurocomputing*, 30(1):47–52.

[Anderberg73]Anderberg, M. R. 1973. *Cluster Analysis for Applications*. Academic Press.

[Anderson35] Anderson, E. 1935. The irises of the Gaspe Peninsula. *Bulletin of the American Iris Society* 59: 2–5.

[Antoniol02] Antoniol, G., Canfora, G., Casazza, G., De Lucia A., and Merlo, E. 2002. Recovering traceability links between code and documentation. *IEEE Transactions on Software Engineering*, Vol. 28, No. 10, 2002, pp. 970-983.

[Bacao05] Bacao, F., Lobo, V., and Painho, M. 2005. Self-organizing Maps as Substitutes for K-Means Clustering. *Lecture Notes in Computer Science*, Volume 3516/2005, pp. 476-483, 2005.

[Baniassad04] Baniassad, E. and Clarke, S. 2004. Finding Aspects in Requirements with Theme/Doc. In *Proceedings of Early Aspects 2004*.

[Baniassad06] Baniassad, E., Clements, P., Araujo, J., Moreira, A., Rashid, A., and Tekinerdogan, B. 2006. Discovering early aspects. *IEEE Software*, Vol 23, No 1, Jan/Feb 2006, pp. 61-70.

[Berkhin02] Berkhin, P. 2002. Survey of Clustering Data Mining Techniques. Accrue Software.

[Berry05] Berry, M. W. and Browne, M. 2005. *Understanding Search Engines: Mathematical Modeling and Text Retrieval (Software, Environments, Tools)*, Second Edition. Society for Industrial and Applied Mathematics.

[Blei03] Blei, D. M., Ng, A. Y., and Jordan, M. I. 2003. Latent dirichlet allocation. *J. Mach. Learn. Res.* 3 (Mar. 2003), 993-1022.

[Boehm04] Boehm, B. and Turner, R. 2004. *Balancing Agility and Discipline: A Guide for the Perplexed*, Addison Wesley.

[Boley] Boley, D.L. 1998. Principal direction divisive partitioning. *Data Mining and Knowledge Discovery*, 2, 4, 325-344.

[Bradley98] Bradley, P. S. and Fayyad, U. M. 1998. Refining initial points for k-means clustering. In J. Shavlik, editor, *Proceedings of the Fifteenth International Conference on Machine Learning (ICML '98)*, pages 91--99, San Francisco, CA, 1998.

[Can90] Can, F. and Ozkarahan, E. A. 1990. Concepts and effectiveness of the cover-coefficient-based clustering methodology for text databases. *ACM Trans. Database Syst.* 15, 4 (Dec. 1990), 483-517.

[Castro-Herrera08] Castro-Herrera, C., Duan, C., Cleland-Huang, J., and Mobasher, B. 2008. Using Data Mining and Recommender Systems to Facilitate Large-Scale, Open, and Inclusive Requirements Elicitation Processes. submitted to RE'08.

[Chen05] Chen, K., Zhang, W., Zhao, H., and Mei, H. 2005. An Approach to Constructing Feature Models Based on Requirements Clustering. In *Proceedings of the 13th IEEE international Conference on Requirements Engineering (Re'05)*, Volume 00, IEEE Computer Society, Washington, DC, 2005.

[Chiu01] Chiu, T., Fang, D., Chen, J., Wang, Y., and Jeris, C. 2001. A robust and scalable clustering algorithm for mixed type attributes in large database environment. In *Proceedings of the Seventh ACM SIGKDD international Conference on*

Knowledge Discovery and Data Mining (San Francisco, California, August 26 - 29, 2001). KDD '01.

[Ciampi00] Ciampi, A. and Lechevallier, Y. 2000. Clustering large, multi-level data sets: an approach based on kohonen self-organizing maps. In *Principles of Data Mining and Knowledge Discovery. 4th European Conference, PKDD 2000*. Proceedings (Lecture Notes in Artificial Intelligence Vol.1910). Springer-Verlag, Berlin, Germany, pages 353–8.

[Cleland-Huang05a] Cleland-Huang, J., Settimi, R., BenKhadra, O., Berezhanskaya, E., and Christina, S. 2005. Goal Centric Traceability for Managing Non-Functional Requirements. Intn'l Conf on Software Engineering, (ICSE'05), (St Louis, USA, May 2005), ACM Press, 362-371.

[Cleland-Huang05b] Cleland-Huang, J., Settimi, R., Duan, C., and Zou, X. 2005. Utilizing supporting evidence to improve dynamic requirements traceability. In proceeding of *International Requirements Engineering Conference*, Paris, France, 2005. pp. 135-144.

[Cleland-Huang06] Cleland-Huang, J., Settimi, R., Zou, X., and Solc, P. 2006. The detection and classification of non-functional requirements with application to early aspects. *IEEE Intn'l Conf. on Reqs Engineering*, Minneapolis, MN, 2006, pp. 39-48.

[Cleland-Huang07a] Cleland-Huang, J., Settimi, R., Zou, X., and Solc, P. 2007. Automated Classification of Non-Functional Requirements. *Requirements Engineering Journal*, August, 2007.

[Clelend-Huang07b] Cleland-Huang, J., Berenbach, B., Clark, S., Settimi, R., and Romanova, E. 2007. Best Practices for Automated Traceability. *IEEE Computer*, 40, 5, (June, 2007), 24-32.

[Cleland-Huang08] Cleland-Huang, J., and Mobasher, B. 2008. Using Data Mining and Recommender Systems to Scale up the Requirements Process. 3rd International Workshop on Ultra Large Software Systems, Leipzig, Germany, May, 2008.

[Cutting92] Cutting, D. R., Karger, D.R., Pedersen, J.O., and Tukey, J. W. 1992. Scatter/Gather: A Cluster-based Approach to Browsing Large Document Collections. *Conf. on Research and Development in information Retrieval*, (Copenhagen, Denmark, 1992), pp. 318-329.

[Davies79] Davies, D. L. and Bouldin, W. 1979. A cluster separation measure. *IEEE PAMI*, 1:224–227, 1979.

[Davis06] Davis, A., Dieste, O., Hickey, A., Juristo, N., and Moreno, A. 2006. Effectiveness of Requirements Elicitation Techniques. *IEEE Intn'l Requirements Engineering Conf.*, Minneapolis, MN, Sept. 2006, pp. 179-188.

[Deerwester90] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. 1990. Indexing by latent semantic analysis. Journal of the American Society for Information Science, 41(6), 391-407.

[Dekhtyar07] Dekhtyar, A., Hayes, J. H., Sundaram, S., Holbrook, A., and Dekhtyar, O. 2007. Technique Integration for Requirements Assessment. RE'07. (Oct. 2007), 141-150.

[DeLucia06] De Lucia, A., Fasano, F., Oliveto, R., and Tortora, G. 2006. ADAMS: advanced artefact management system. *10th European Conference on Software Maintenance and Reengineering, (CMSR'06)*, (2006), 349-350.

[Dempster77] Dempster, A., Laird, N., and Rubin, D. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, Series B, 39(1):1–38, 1977.

[Dhillon01a] Dhillon, I. S. and Modha, D. S. 2001. Concept decompositions for large sparse text data using clustering. *Machine Learning*, 42, ½, (Jan. 2001), 143-175.

[Dhillon01b] Dhillon, I., Fan, J., and Guan, Y. 2001. Efficient Clustering of Very Large Document Collections. In R. Grossman, G. Kamath, and R. Naburu, editors, *Data Mining for Scientific and Engineering Applications*, Kluwer Academic Publishers, 2001.

[Dhillon01c] Dhillon, I. S. 2001. Co-clustering documents and words using bipartite spectral graph partitioning. In Proceedings of *the Seventh ACM SIGKDD international Conference on Knowledge Discovery and Data Mining* (San Francisco, California, August 26 - 29, 2001).

[Dhillon02] Dhillon, I. S., Guan, Y., and Kogan, J. 2002. Iterative Clustering of High Dimensional Text Data Augmented by Local Search. In Proceedings of *the 2002 IEEE international Conference on Data Mining (Icdm'02)* (December 09 - 12, 2002).

[Duan07a] Duan, C. and Cleland-Huang, J. 2007. A Clustering Technique for Early Detection of Dominant and Recessive Cross-Cutting Concerns. In Proceedings of *the Early Aspects At Icse: Workshops in Aspect-Oriented Requirements Engineering and Architecture Design* (May 20 - 26, 2007). International Conference on Software Engineering.

[Duan07b] Duan, C. and Cleland-Huang, J. 2007. Clustering Support for Automated Tracing. International Conference on Automated Software Engineering, Atlanta, Georgia, November, 2007.

[Duda01] Duda, R. O., Hart, P. E., and Stork, D. G. 2001. *Pattern classification* (2nd edition), Wiley, New York.

[Dumais93] Dumais, S. 1993. LSI Meets TREC: A Status Report, In Proc. *First Text Retrieval Conference (TREC1)*, pp. 137--152, NIST Special Publication 500-207.

[Dumais95] Dumais, S. 1995. Latent semantic indexing (LSI): TREC-3 report. In D. Hartman, editor, *The Third Text REtrieval Conference*, NIST special publication 500-225, pages 219-230,1995.

[Dunn74] Dunn, J. C. 1974. Well separated clusters and optimal fuzzy partitions. *Journal of Cybernetics*, 4:95-104, 1974.

[Elkan06] Elkan, C. 2006. Clustering documents with an exponential-family approximation of the Dirichlet compound multinomial distribution. In Proceedings of *the 23rd international Conference on Machine Learning* (Pittsburgh, Pennsylvania, June 25 - 29, 2006). ICML '06, vol. 148.

[Ertz01] Ertz, L., Steinbach, M., and Kumar, V. 2001. Finding Topics in Collections of Documents: A Shared Nearest Neighbor Approach. Text Mine '01 at SIAM Intn'l. Conf. on Data Mining, (Chicago, IL, 2001).

[Fern03] Fern, X. Z. and Brodley, C. E. 2003. Random projection for high dimensional data clustering: A cluster ensemble approach. In *Proc. of ICML*, Washington, DC (2003) 186—193.

[Fern04] Fern, X. Z. and Brodley, C. E. 2004. Solving cluster ensemble problems by bipartite graph partitioning. In *Proceedings of the Twenty-First international Conference on Machine Learning* (Banff, Alberta, Canada, July 04 - 08, 2004).

[Forgy65] Forgy, E. 1965. Cluster analysis of multivariate data: Efficiency versus interpretability of classification. *Biometrics*, 21, 768-780.

[Fraley98] Fraley, C. and Raftery, A. E. 1998. How many clusters? Which clustering method? -Answers via model-based cluster analysis. *The Computer Journal* 41, 578–588.

[Fraley02] Fraley, C. and Raftery, A. E. 2002. Model-based clustering, discriminant analysis and density estimation. *Journal of the American Statistical Association* 97, 611–631.

[Fred05] Fred, A. L. and Jain A. K. 2005. Combining Multiple Clusterings Using Evidence Accumulation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 6, pp. 835-850, June, 2005.

[Flexer01] Flexer, A. 2001. On the use of self-organizing maps for clustering and visualization. *Intelligent Data Analysis*, 5:373–84.

[Girolami03] Girolami, M. and Kabán, A. 2003. On an equivalence between PLSI and LDA. In *Proceedings of the 26th Annual international ACM SIGIR Conference on Research and Development in informaion Retrieval* (Toronto, Canada, July 28 - August 01, 2003). SIGIR '03.

[Gold05] Goldstein, H. 2005. Who Killed the Virtual Case File? *IEEE Spectrum*, Vol. 42, No. 9, 2005, pp. 24-35.

[Gotel94] Gotel, O. C. Z. and Finkelstein A. C. W. 1994. An Analysis of the Requirements Traceability Problem. *Proc.of the 1st Intn'l Conf. on Requirements Engineering (ICRE '94)*, (Colorado Springs, CO, 1994), IEEE Computer Society Press, 94-101.

[Gotel95] Gotel, O. 1995. Contribution Structures for Requirements Traceability. London, England: Imperial College, Department of Computing, 1995.

[Griffiths04] Griffiths, T. and Steyvers, M. 2004. Finding Scientific Topics. *Proceedings of the National Academy of Sciences*, 101 (suppl. 1), 5228-5235.

[Guha98] Guha, S., Rastogi, R., and Shim, K. 1998. CURE: An efficient clustering algorithm for large databases. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pages 73--84, New York, 1998.

[Halkidi01] Halkidi, M., Batistakis, Y., and Vazirgiannis, M. 2001. On Clustering Validation Techniques. *Journal of Intelligent Information Systems*, 17,2-3,. (Dec. 2001), 107-145.

[Hartigan75] Hartigan, J. 1975. *Clustering Algorithms*. John Wiley & Sons, New York, NY.

[Hastie89] Hastie, T. and Stuetzle, W. 1989. Principal curves. *Journal of the American Statistical Association*, 84:502-516.

[Hayes06] Huffman Hayes, J., Dekhtyar, A., and Karthikeyan Sundaram, S. 2006. Advancing Candidate Link Generation for Requirements Tracing: The Study of Methods. *IEEE Transactions on Software Engineering*, 32, 1, (2006), IEEE Computer Society Press, 4-19.

[Hettich99] Hettich, S. and Bay, S. D. 1999. The UCI KDD Archive [http://kdd.ics.uci.edu]. Irvine, CA: University of California, Department of Information and Computer Science.

[Hofmann99] Hofmann, T. 1999. Probabilistic latent semantic indexing. In *Proceedings of the 22nd Annual international ACM SIGIR Conference on Research and Development in information Retrieval* (Berkeley, California, United States, August 15 - 19, 1999).

[Hsia88] Hsia, P. and Yaung, A. T. 1988. Another Approach to System Decomposition: Requirements Clustering. In *Proceedings of COMPSAC '88*, Chicago, IL, October 3-6, 1988.

[Hsia96] Hsia, P., Hsu, C. T., Kung, D. C., and Holder, L. B. 1996. User-Centered System Decomposition: Z-Based Requirements Clustering. In *Proceedings of the 2nd international Conference on Requirements Engineering (ICRE '96)* (April 15 - 18, 1996). ICRE. IEEE Computer Society, Washington, DC, 126.

[Jain88] Jain, A. K. and Dubes, R. C. 1988. *Algorithms for Clustering Data*. Prentice-Hall, Inc.

[Jain99] Jain, A. K., Murty, M. N., and Flynn, P. J. 1999. Data Clustering: A Review. *ACM Computing Surveys*, Vol 31, No. 3, 264-323.

[Kaski97] Kaski, S. 1997. Data Exploration Using Self-Organizing Maps. PhD thesis, Helsinki University of Technology, 1997.

[Karypis99] Karypis, G., Han, E., and Kumar, V. 1999. Chameleon: Hierarchical Clustering Using Dynamic Modeling. *Computer* 32, 8 (Aug. 1999), 68-75.

[Kaufman90] Kaufman, L. and Rousseeuw, P. 1990. *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley and Sons, New York, NY.

[Kit06] Kit, L., Man, C., and Baniassad, E. 2006. Isolating and relating concerns in requirements using latent semantic analysis. SIGPLAN Not. 41, 10 (Oct. 2006).

[Kohonen01] Kohonen, T. 2001. *Self-organizing Maps*. Springer-Verlag, Berlin Heidelberg New York.

[Kowalski97] Kowalski, G. 1997. *Information Retrieval Systems – Theory and Implementation*. Kluwer Academic Publishers.

[Kulkarni06] Kulkarni, A. 2007. Unsupervised Context Discrimination and Automatic Cluster Stopping. Master Thesis.

[Laerhoven01]Van Laerhoven, K. 2001. Combining the self-organizing map and k-means clustering for on-line classification of sensor data. In *Artificial Neural Networks-ICANN* 2001, Proceedings, pages 464–469.

[Lange05] Lange, T. and Buhmann, J. M. 2005. Combining partitions by probabilistic label aggregation. In *Proceeding of the Eleventh ACM SIGKDD international Conference on*

Knowledge Discovery in Data Mining (Chicago, Illinois, USA, August 21 - 24, 2005).

[Li04] Li, T., Ogihara, M., and Ma, S. 2004. On combining multiple clusterings. In *Proceedings of the Thirteenth ACM international Conference on information and Knowledge Management* (Washington, D.C., USA, November 08 - 13, 2004).

[Lin06] Lin, J., Lin, C.C., Cleland-Huang, J., Settimi, R., Amaya, J., Bedford, G., Berenbach, B., Ben Khadra, O., Duan, C., and Zou, X. 2006. Poirot: A Distributed Tool Supporting Enterprise-Wide Traceability. IEEE International Conference on Requirements Engineering, (September, 2006), 356-357.

[Madsen05] Madsen, R. E., Kauchak, D., and Elkan, C. 2005. Modeling word burstiness using the Dirichlet distribution. In *Proceedings of the 22nd international Conference on Machine Learning* (Bonn, Germany, August 07 - 11, 2005). ICML '05, vol. 119.

[Marcus03] Marcus A. and Maletic, J. 2003. Recovering Documentation-to-Source-Code Traceability Links using Latent Semantic Indexing. International Conference on Software Engineering, 2003, pp. 125-137.

[Marcus05] Marcus, A., Maletic, J. I., and Sergeyev, A. 2005. Recovery of Traceability Links Between Software Documentation and Source Code. International Journal of Software Eng. and Knowledge Eng., 15, 4, (October 2005), World Scientific Publishing Co. 811-836.

[McLachlan00] McLachlan, G. J. and D. Peel. 2000. *Finite Mixture Models*. Wiley.

[Meila98] Meila, M. and Heckerman, D. 1998. An Experimental Comparison of Several Clustering and Initialization Methods. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence* (Morgan Kaufmann, Inc., San Francisco, CA, 1998) 386-395.

[Meila01] Meila, M. and Shi, J. 2001. Learning Segmentation with Random Walk. *Neural Information Processing Systems*, NIPS, 2001.

[Miller56] Miller, G.A. 1956. The Magical Number Seven, Plus or Minus Two: Some Limits on our Capacity for Processing Information. *The Psychological Review*, 63, (1956), 81-97.

[Monti03] Monti, S., Pablo, T., Mesirov, J. and Golub, T. 2003. Consensus Clustering: A Resampling-Based Method for Class Discovery and Visualization of Gene Expression Microarray Data. *Machine Learning*, 52, pp. 91-118, 2003.

[Nigam00] Nigam, K., McCallum, A. K., Thrun, S., and Mitchell, T. 2000. Text Classification from Labeled and Unlabeled Documents using EM. *Mach. Learn.* 39, 2-3 (May. 2000), 103-134.

[Ng02] Ng, R. T. and Han, J. 2002. CLARANS: A Method for Clustering Objects for Spatial Data Mining. *IEEE Transactions on Knowledge and Data Engineering*, vol. 14, no. 5, pp. 1003-1016, September/October, 2002.

[Pena99] Pena, J.M., Lozano, J.A. and Larran~aga P. 1999. An empirical comparison of four initialization methods for the k-means algorithm. *Pattern Recognition Letters*, 20, 1999, 1027-1040. 50.

[Rashid02] Rashid, A., Sawyer, P., Moreira, A. M., and Araújo, J. 2002. Early Aspects: A Model for Aspect-Oriented Requirements Engineerin. In *Proceedings of the 10th Anniversary IEEE Joint international Conference on Requirements Engineering* (September 09 - 13, 2002). RE. IEEE Computer Society, Washington, DC, 199-202.

[Rashid03] Rashid, A., Moreira, A., and Araújo, J. 2003. Modularisation and composition of aspectual requirements. In *Proceedings of the 2nd international Conference on Aspect-Oriented Software Development* (Boston, Massachusetts, March 17 - 21, 2003). AOSD '03. ACM, New York, NY, 11-20.

[Rigouste07] Rigouste, L., Cappé, O., and Yvon, F. 2007. Inference and evaluation of the multinomial mixture model for text clustering. *Inf. Process. Manage.* 43, 5 (Sep. 2007), 1260-1280.

[Robertson99] Robertson, S. and Robertson, J. 1999. *Mastering the Requirements Process*. Addison-Wesley, 1999.

[Rodrigues04] Mendes Rodrigues, M. E. S. and Sacks, L. 2004. A scalable hierarchical fuzzy clustering algorithm for text mining. In: *Proc. of the 4th International Conference on Recent Advances in Soft Computing*, RASC'2004, pp. 269-274, Nottingham, UK, Dec. 2004.

[Rosenhaine04] Rosenhaine, L. 2004. Identifying Crosscutting Concerns in Requirements Specifications.

[Roth02] Roth, V., Lange, T., Braun, M., and Buhmann, J. 2002. A Resampling Approach to Cluster Validation. In *Intl. Conf. on Computational Statistics*, pp. 123-129, 2002.

[Salton86] Salton, G. and McGill, M. J. 1986. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc.

[Salvador04] Salvador, S. and Chan, P. 2004. Determining the Number of Clusters/Segments in Hierarchical Clustering/Segmentation Algorithms. In *Proceedings of the 16th IEEE international Conference on Tools with Artificial intelligence (Ictai'04)* - Volume 00 (November 15 - 17, 2004).

[Sander98] Sander, J., Ester, M., Kriegel, H. P., and Xu, X. 1998. Density-based clustering in spatial databases: the algorithm GDBSCAN and its applications. In *Data Mining and Knowledge Discovery*, 2, 2, 169-194.

[Sampaio05] Sampaio, A., Loughran, N., Rashid, A., and Rayson, P. 2005. Mining Aspects in Requirements. Workshop on Early Aspects 2005.

[Schapire03] Schapire, R. E. 2003. The boosting approach to machine learning: An overview. In D. D. Denison, M. H. Hansen, C. Holmes, B. Mallick, B. Yu, editors, *Nonlinear Estimation and Classification*. Springer.

[Shi00] Shi J. and Malik, J. 2000. Normalized Cuts and Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8), 888-905, August 2000.

[Siersdorfer04] Siersdorfer, S. and Sizov, S. 2004. Restrictive clustering and metaclustering for self-organizing document collections. In *Proceedings of the 27th Annual international ACM SIGIR Conference on Research and Development in*

information Retrieval (Sheffield, United Kingdom, July 25 - 29, 2004).

[Singhal96] Singhal, A., Buckley, C., and Mitra, M. 1996. Pivoted document length normalization. In *Proceedings of the 19th Annual international ACM SIGIR Conference on Research and Development in information Retrieval* (Zurich, Switzerland, August 18 - 22, 1996). SIGIR '96.

[Smyth96] Smyth, P. 1996. Clustering Using Monte-Carlo Cross-Validation. In *Proc. 2nd KDD*, pp.126-133, 1996.

[Soares02] Soares, S., Laureano, E., and Borba, P. 2002. Implementing Distribution and Persistence Aspects with AspectJ. In *Proc. of Object Oriented Programming, Systems, Languages, and Applications (OOPSLA' 02)*, (November, 2002), 174–190.

[Steinbach00] Steinbach, M., Karypis, G., and Kumar, V. 2000. A comparison of document clustering techniques. Workshop on Text Mining at Intn'l Conf on Knowledge Discovery and Data Mining.

[Steyvers07] Steyvers, M. and Griffiths, T. 2007. Probabilistic topic models. In T. Landauer, D McNamara, S. Dennis, and W. Kintsch (eds), *Latent Semantic Analysis: A Road to Meaning*. Laurence Erlbaum.

[Strehl03] Strehl, A. and Ghosh, J. 2003. Cluster ensembles – a knowledge reuse framework for combining multiple partitions. *J. Mach. Learn.* Res. 3 (Mar. 2003), 583-617.

[Theodoridis06] Theodoridis, S. and Koutroumbas, K. 2006. *Pattern Recognition* (3rd edition), Elsevier.

[TREC] TREC Data collection from Text REtrieval Conference (TREC), URL http://trec.nist.gov/.

[Tibshirani01] Tibshirani, R., Walther, G., Botstein, D., and Brown, P. 2001. Cluster Validation by Prediction Strength, Technical Report, 2001-21, Dept. of Biostatistics, Stanford Univ, 2001.

[Tibshirani03] Tibshirani, R., Walther, G., and Hastie, T. 2003. Estimating the number of clusters in a dataset via the Gap statistic. In *JRSSB* 2003.

[Topchy03] Topchy, A., Jain, A. K., and Punch, W. 2003. Combining Multiple Weak Clusterings. In *Proceedings of the Third IEEE international Conference on Data Mining* (November 19 - 22, 2003).

[Vasko02] Vasko, K. and T. Toivonen. 2002. Estimating the number of segments in time series data using permutation tests. In *Proc. IEEE Intl. Conf. on Data Mining*, pp. 466-473, 2002.

[Vesanto97] Vesanto, J. Data Mining Techniques Based on the Self-Organizing Map, Master thesis.

[Vesanto00] Vesanto, J. and Alhoniemi, E. 2000. Clustering of the self-organizing map. *IEEE Transactions on Neural Networks*, 11(3):586–600.

[Wiegers99] Wiegers, K. E. 1999. *Software Requirements*, Microsoft Press, Redmond, WA, 1999.

[Wu03] Wu, W., Xiong, H., and Shekhar. S., (Eds.) 2003. *Clustering and Information Retrieval*. Kluwer.

[Xu98] Xu, X., Ester, M., Kriegel, H. P., and Sander, J. 1998. A distribution-based clustering algorithm for mining in large spatial databases. In *Proceedings of the 14th ICDE*, 324-331, Orlando, FL.

[Xu03] Xu, W., Liu, X., and Gong, Y. 2003. Document clustering based on non-negative matrix factorization. In *Proceedings of the 26th Annual international ACM SIGIR Conference on Research and Development in informaion Retrieval* (Toronto, Canada, July 28 - August 01, 2003).

[Yaung92] Yaung, A. T. 1992. Design and implementation of a requirements clustering analyzer for software system decomposition. In *Proceedings of the 1992 ACM/SIGAPP Symposium on Applied Computing: Technological Challenges of the 1990's*, Kansas City, Missouri, 1992.

[Zaragoza03] Zaragoza, H., Hiemstra, D., and Tipping, M. 2003. Bayesian extension to the language model for ad hoc information retrieval. In *Proceedings of the 26th Annual international ACM SIGIR Conference on Research and Development in informaion Retrieval* (Toronto, Canada, July 28 - August 01, 2003). SIGIR '03.

[Zamir97] Zamir, O., Etzioni, O., Madani, O., and Karp, R. M. 1997. Fast and Intuitive Clustering of Web Documents. In *Proc. of the Intn'l Conf on Knowledge Discovery and Data Mining*, (August 14-17, 1997), 287-290.

[Zave97] Zave, P. 1995. Classification of Research Efforts in Requirements Engineering. In *Proc. RE'95 - 2nd IEEE Int. Symposium on Requirements Engineering*, March 1995, 214-216.

[Zhao01] Zhao, Y. and Karypis, G. 2001. Criterion functions for document clustering: Experiments and analysis. Technical Report TR #01--40, Department of Computer Science, University of Minnesota.

[Zhao02] Zhao, Y, and Karypis, G. 2001. Evaluation of hierarchical clustering algorithms for document datasets. In *Proceedings of the Intn'l Conf. on information and Knowledge Management*, (McLean, Virginia, Nov 4-9, 2002), 515-524.