Improving Requirements Clustering in an Interactive and Dynamic Environment

BY

CHUAN DUAN

A DISSERTATION SUBMITTED TO THE SCHOOL OF COMPUTING,

COLLEGE OF COMPUTING AND DIGITAL MEDIA OF DEPAUL

UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE

OF

DOCTOR OF PHILOSOPHY

DEPAUL UNIVERSITY

CHICAGO, ILLINOIS

2009

DePaul University

College of Computing and Digital Media

Dissertation Defense Report

I have read the dissertation written by:

Name Chuan Duan Student No: 0904012

(*To the advisor*): The following dissertation title is identical to the one on the title page of the draft returned to the student. This title is approved by me and it is to be used when the final copies of the dissertation are prepared.

Title of dissertation:

Improving Requirements Clustering in an Interactive and Dynamic Environment

Advisor's Initials JH

☑ Acceptable. Candidate may proceed to make final copies

☐ Pass, with revisions stated below:

☐ Not Acceptable. Please explain:

Dr. Jane Cleland- Huang	Jane Huang	7/13/09
Advisor (Print Name)	Signature	Date
Dr. Andrian Marcus		4/20/09
1st Reader (Print Name)	Signature	Date
Dr. Bamshad Mobasher		4/20/09
2nd Reader (Print Name)	Signature	Date
Dr. Daniella Raicu	Daniela Raicu	4/20/2009
3rd Reader (Print Name)	Signature	Date
4th Reader (Print Name)	Signature	Date

IMPROVING REQUIREMENTS CLUSTERING IN AN INTERACTIVE AND DYNAMIC ENVIRONMENT

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. 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. Domain analysis in product line development can be partially automated through clustering features to identify shared and variable components.

Unfortunately, despite a significant body of work describing document clustering techniques, there is almost no prior work which systematically evaluates the challenges, constraints, and nuances of requirements clustering. As a result, developers and researchers select requirements clustering algorithms with little understanding of their appropriateness for

the task, and the effectiveness of software engineering tools and processes that depend on requirements clustering is severely limited.

This research directly addresses the problem of improving requirements clustering. The main contribution includes the extensive study of existing clustering methods and their combinations, as well as development of new algorithms that are able to deliver high quality clusters adaptively in various application contexts. Among a set of popular basic hierarchical and partitioning algorithms, the two-stage spherical K-means was identified as the most effective one. Indicated by further experiments with a number of enhancement clustering methods, consensus clustering, a hybrid clustering approach that combines the knowledge from multiple clusterings, was found to be effective and robust in clustering requirements.

Two new approaches are proposed to address unique domain characteristics. The first approach takes advantage of the human-centric environment of many software engineering activities, including online requirements gathering forums or automated traceability tools, in which users provide feedback about the quality of each cluster. To utilize this feedback, a consensus-based constrained clustering approach is proposed and empirically shown to be effective in choosing informative feedback and generating improved clusterings. The second domain problem addresses the need to balance change versus stability in the incremental clustering process. Clusters change as requirements evolve and users' feedback is elicited, yet users desire stability. For example, when clustering is used to automatically organize requirements in a visible context such as a feature gathering forum or an automated trace tool, then the set of feature requests is dynamic and clustering is incremental, yet stakeholders do not want clusters to constantly change. This is particularly evident in a web-based forum, where the generated clusters are used to anchor discussion threads and users do now want their posts to

move significantly between multiple threads. A seed-preserving clustering method was therefore designed to maintain stability without sacrificing clustering quality.

This work developed a deeper insight into effective requirements clustering techniques and specific enhancements that deliver higher quality clusters. These findings are anticipated to be useful for supporting requirements related activities.

ACKNOWLEDGEMENTS

I would like to take this opportunity to thank my advisor, Dr. Jane Huang for all her support during my PhD studies. She helped me locate and refine my research topic, carefully and patiently edited every paper I wrote, and provided me numerous opportunities of paper presentation to improve my speech techniques. She knows my strengths and trusts my potential to make contribution to the RE area. I would also like to thank my lovely wife Ming, who brings real joy to my life – I will always love you. I would like to express my gratitude to my dear parents and parents-in-law, who love us so much and have offered us so much. Last but not least, I want to thank my best friends, two sweet couples: Yong Chen and Hui Wang, Fang Fang and Jingye Xu. They took great care of me and Ming over the last five years and we have had lots of fun spending time with them.

TABLE OF CONTENTS

NOTATIONS AND ABBREVIATIONS

AHC	Agglomerative Hierarchical Clustering
EM	Expectation-Maximization
FMM	Finite Mixture Model
i.i.d.	Identically and Independently Distributed
IR	Information Retrieval
LDA	Latent Dirichlet Allocation
MAP	Maximum a Posteriori
ML	Maximum Likelihood
MLE	Maximum Likelihood Estimate
MPCA	Multinomial Principal Component Analysis
NMI	Normalized Mutual Information
NMF	Nonnegative Matrix Factorization
PCA	Principal Component Analysis
p.d.f.	Probability Density Function
PLSA	Probabilistic Latent Semantic Analysis
RE	Requirements Engineering
SOM	Self-organizing Map
SPK	Spherical K-means
SRS	Software Requirement Specification
SSE	Sum of Squared Errors
SVD	Singular Value Decomposition

LIST OF TABLES

LIST OF FIGURES

LIST OF EXPERIMENTS

SECTION A DISSERTATION INTRODUCTION

CHAPTER 1. INTRODUCTION

Software system requirements specify the goals, functionalities, and the constraints of a software system [Zave97]. Software requirements are often classified as functional requirements (FR) and non-functional requirements (NFR). FRs relate to the functionalities or services the system is expected to provide, whereas NRFs describe emergent system properties such as dependability and security. The process of identifying, analyzing, documenting, and validating requirements is called Requirements Engineering (RE).

A RE process is comprised of a collection of related activities. Four primary RE activities are requirements elicitation, documentation, validation, and management. Requirements elicitation lies at the beginning of a typical RE process, in which system boundaries and stakeholders are identified, and system goals, use cases, and scenarios are elicited from the stakeholders. The process of elicitation is supported by a variety of techniques, such as questionnaire, reviews, group brainstorming, prototyping, and the choice of these techniques depends on available resources and the nature of information that needs to be collected. Usually following elicitation, requirements documentation specifies the requirements in clear language and notations in order to ensure that these requirements can be effectively communicated among the stakeholders. Requirements validation then checks that the requirements actually specify the functionality and behavior that the stakeholders need. A few common checks include validity checks, consistency checks, and completeness checks. Lastly, requirements management is needed to understand and control requirements evolution. During the requirements management stage a particularly important decision is traceability policies, which define the relationships between requirements and between requirements and designs, as well as how these relationships should be recorded and maintained.

Several different studies have highlighted the importance of the requirements phase. For example, the Chaos report, generated by the Standish research group, studied factors that contributed to failed and successful projects. They reported that requirements related problems such as incomplete and changing requirements and specifications as one of the leading causes of project failure [Standish94]. Requirements problems are exacerbated by the increasing scale and complexity of ongoing projects, suggesting that new approaches are needed to mitigate the subsequent risks and provide active support for managing projects containing tens of thousands of requirements.

For example of increasing sizes of software projects, in a case study performed at Sony Ericsson [Wnuk09], the requirements database contained around 30,000 system requirements regarding the company's mobile phone. In another example, NASA engineer, Kristin Farry,

stated that the paperwork produced whilst eliciting Space Station requirements, could almost have been used to build a stairway all the way into space and thereby eliminating the need for a rocket launch [Hook01]. In yet another example, the FBI Virtual Case File (VCF) project [Gold05], which was a 170 million dollar project, had a huge number of features that were documented in an 800 page requirements specification. These kinds of large projects with tens of thousands of requirements are becoming almost common place.

With the increase in project size, a significant amount of effort is expended to discover and analyze the requirements. However, practice has shown that this effort does not necessarily translate into a successful project. For example, the VCF project failed despite substantial money and labor spent in eliciting and prioritizing requirements, and subsequent post mortem analysis laid the blame partially on a bloated requirements specification in which key requirements were not clearly differentiated from design specifications. Methods that can automatically extract, abstract, and organize ideas can greatly alleviate the burden on human labor in such large projects. For example, in a current project which the Center for Systems and Requirements Engineering is currently consulting on, human users are manually mapping 14K's worth of regulatory constraints to over 30,000 requirements. Ideally these highly intensive user-centric tasks would be at least semi-automated.

Automated methods include clustering, the automatic division of data into similar groups. Clustering methods have been employed widely and rather successfully 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]. These same needs also exist in the domain of RE. Therefore, it sounds reasonable to adopt theories, methods, and tools from the document clustering discipline to provide automated support for working with requirements on large projects.

The two domains of document clustering and requirements clustering are similar in several respects. For example, both domains rely on the decomposition of textual information, suggesting the basic framework of document clustering can be adopted in requirements clustering. Furthermore, both areas share a number of challenges, such as high dimensionality of data, significant background noise, and the need for scalability.

However, the clustering of requirements differs significantly from the clustering of ordinary documents in a number of ways. First, they differ in the cluster granularity, i.e. the number and size of generated clusters. Document clustering usually aims at organizing the documents into a limited number of categories to facilitate a few basic tasks such as browsing and searching. The number of the categories is typically small, and is usually known in advance. In contrast, much finer-grained requirement clusters are often required in RE tasks because the clusters are often visible to human users and are designed to help users actually understand the requirements.

Worse still, there are no existing reference categories for requirement clusters, meaning that the granularity and themes of the clusters must be determined automatically.

Second, the characteristics of the requirement data set make requirement clustering more difficult. The documents studied in typical document clustering tend to contain rich textual information and tend to focus on one dominant significant topic in each document. Software requirements, especially the early-stage feature requests, on the other hand, are typically documented tersely and contain more noisy information. Terseness and noise have several consequences. One is that requirements clusters generated by traditional document clustering algorithms are often formed around a trivial dominant topic. Another consequence, associated with the trivial topic problem, is that 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 each of the requirements into two distinct clusters so that they were grouped together in respect to each of their conceptual themes. For example, requirement 1126 should have been placed in both the "login" theme as well as a 'local display' theme.

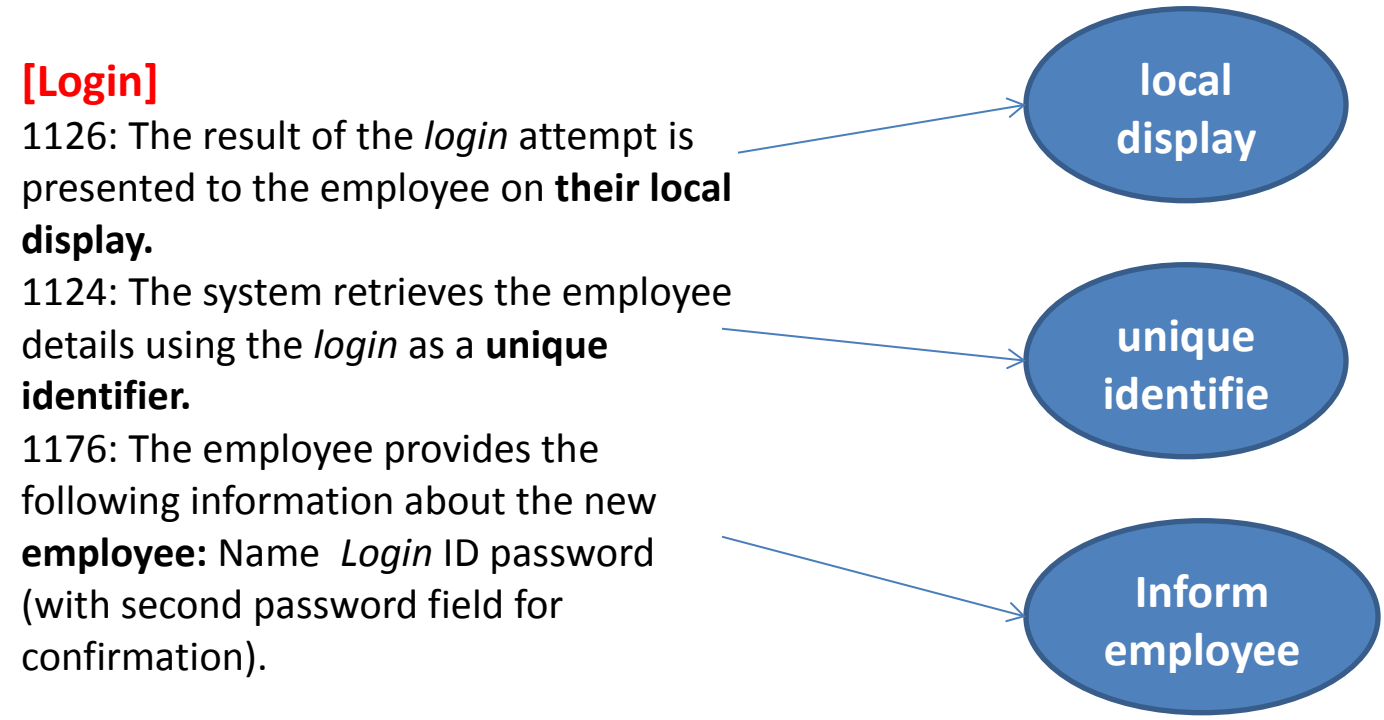


Figure 1.1 Trivial topics distort more meaningful clustering.

Third, many clustering applications in RE must support incremental clustering because requirements arrive from stakeholders in increments. For example, in a requirements forum

where stakeholders contribute, communicate, and modify requirements, the set of requirements to be clustered grows over time. In such a forum, one particularly important consideration in incremental requirement clustering is that dynamically generated forums should not exhibit substantial fluctuation between incremental re-clusterings; in other words, the structure of forums is preferred to be stable. Meanwhile, the quality of the clusters should not degrade too much. If a stakeholder is either frequently re-assigned to a different forum by the clustering system, or if the strong emphasis on stability leads to deterioration of cohesiveness within a forum, then the stakeholder might be reluctant to use the system. This need for stable incremental clustering without sacrificing performance makes requirement clustering challenging compared with ordinary document 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. Unfortunately, requirements clustering has no such answer set to serve the purpose of evaluation, and this is a significantly non-trivial problem. 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.

Investigating these four problems and improving the clustering of requirements is the objective of research in this dissertation. The primary contributions of the work include (1) the extensive investigation, experimentation, comparison, and analyses of existing document clustering methods applied to the requirements domain, (2) construction and evaluation of various hybrid models that utilize consensus clustering or pipelining techniques to improve clustering results, (3) incorporation of user feedback to improve cluster quality. As just mentioned, the nature of requirements' text poses big challenges to the clustering algorithm and experimental results show that a fully automated approach cannot produce clusters good enough for practical use. In some RE activities such as requirement elicitation, however, a great number of user interactions are expected. These interactions can be analyzed and transformed into certain forms of feedback that may be incorporated into the clustering system to generate better clusters; and finally (4) proposal and validation of a novel incremental clustering algorithm that's able to generate stable and high quality clusters.

The research also addresses the application of clustering to support three requirement engineering activities: early aspect identification, automated requirement tracing, and requirement forum management. Aspects are referred to as the cross-cutting concerns distributed at different locations of the system specification and they have a crucial impact on software architectural decisions. Clustering methods can be used to aggregate similar concerns in the system specification and then automatically inspect the dispersion of these concerns in order to identify candidate aspects [Duan07a]. Requirements tracing tracks a requirement and its relationships to other software engineering artifacts across various directions of the development

cycle. One approach to traceability adopts information retrieval methods to generate candidate traces [Duan07b]. Clustering was used in the research to automatically structure the traces and to give each trace a context, thereby enhancing the comprehension and validation of the traces. Requirement forums, which were briefly described earlier, can also benefit from clustering support. In a recent study [Cleland-Huang08] we found a high degree of redundancy existing in the forums when discussion threads were created and managed entirely by humans. When clustering techniques were used to automatically organize the feature requests the forums were found to contain significantly more cohesive and less redundant discussion threads. These three examples demonstrated the usefulness of clustering requirements. This dissertation work was designed to improve clustering results so as to provide higher quality support for applications that rely on requirements clustering.

The remainder of the dissertation is organized as follows. Section B includes chapters 2, 3, 4, the majority of which survey existing clustering techniques. Chapter 2 provides a background of document clustering techniques, with the focus on existing term-based crisp document clustering algorithms. It then presents the author's early work on using these algorithms to cluster three small scale requirement data sets. Chapter 3 introduces probabilistic topic modeling of documents and requirements. Chapter 4 reviews existing consensus clustering techniques, which utilize a combination of multiple clustering techniques to improve the clustering results. The main body of research is found in Section C, including Chapters 5, 6, 7, 8, 9, and 10. Chapter 5 gives an overview of the research. Chapter 6 discusses the selection of cluster validation metrics. Chapter 7 describes a thorough study of clustering requirements using existing well-known clustering algorithms and various enhancement techniques. It reports the results of evaluating standalone crisp clustering algorithms, matrix decomposition based term weighting, SOM sampling seeded SPK, pipeline hybrid clustering algorithms, and finally consensus clustering. Chapter 8 proposes a consensus-based constrained clustering framework for incorporating user feedback. This approach is empirically shown in our experiments to be effective to enhance the quality of requirement clustering. Chapter 9 discusses a new incremental clustering algorithm that focuses on maintaining clustering stability without seriously sacrificing cluster quality. Chapter 10 describes how clustering is utilized to support the two requirement engineering activities. Chapter 11 concludes the whole dissertation and outlines the direction for future work.

Because the background survey in Section B includes a number of illustrations of clustering algorithms and clustering results, the data sets used throughout this dissertation need to be introduced here.

Five requirement data sets are used in experiments: IBS, EBT, PHW, STUDENT, and SUGAR. 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.

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.

STUDENT is a requirements collection gathered for a student web-portal system, similar in nature to Amazon.com. Thirty six Master level students, enrolled in two different advanced Software Engineering classes at DePaul University, were instructed to select one or more of the following roles: typical student, text book reseller, textbook author, textbook publisher, instructor, portal administrator, architect, project manager, and developer. Each participant was then asked to contribute approximately ten statements of need for the system, resulting in a total of 366 statements. A candidate answer clustering for STUDENT comprised of 29 clusters was generated through manual evaluation of requirements by two researchers at the DePaul Systems and Requirements Engineering Center (SREC). This answer clustering was perceived by both researchers to be significantly better than the clusters previously generated by several automated algorithms.

SUGAR data set is comprised of 1000 feature requests mined from SugarCRM, a large open source project. SugarCRM supports campaign management, email marketing, lead management, marketing analysis, forecasting, quote management, case management and many other features. The tool has been in the open source community since 2004. To enter a request in the Sugar CRM forum, a user browses through a list of existing threads and determines whether to submit to an existing thread or to create a new one. As such, the threads can be seen as a user-defined clustering of the feature requests. 1000 requests were contributed by 523 different stakeholders over a two year period, and distributed across 293 threads. The answer clustering consists of 40 clusters and received thorough inspection of two analysts at SREC.

In addition to these data sets, six standard data sets from TREC are included and their descriptions can be found in two papers of Zhao [Zhao01, Zhao02]. Compared with other TREC data sets, these six have similar properties to requirements data sets, and they have been widely tested and used in experiments of clustering research. The characteristic of all datasets are showed in Table 1.1.

Although the chosen TREC data sets are more similar to requirements data sets than most other TREC data sets, two major differences between them and requirements data sets can be observed in Table 1.1. First, the total number of terms and the average number of distinct terms of requirements are far less than those of TREC documents; in other words, the texts of requirements are highly terse. Second, because the requirement clusters useful to RE domain tasks should typically be fine-grained, the number of clusters or threads of requirements is usually smaller than that of TREC documents.

Table 1.1 Summary of requirement data sets and some TREC data sets.

(n_d is the total number of requirements, n_w the total number of terms, |d| the average number of distinct terms each requirement contains, K the number of clusters or threads)

	Data	n_d	n_w	|d|	K
Requirement data sets	IBS	171	122	5	/
	EBT	41	116	7	/
	PHW	244	134	5	/
	STUDENT	366	908	8	29
	SUGAR	1000	3463	27	40
TREC data sets	tr11	414	6429	281	9
	tr12	313	5804	273	8
	tr23	204	5832	385	6
	tr31	927	10128	268	7
	tr41	878	7454	195	10
	tr45	690	8261	280	10

SECTION B BACKGROUND SURVEY

CHAPTER 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, fuzzy clustering algorithms 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 dissertation 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 dissertation, 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

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. The following sections in this chapter discuss these components in details.

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 words. 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, a thesaurus or explicitly constructed matching word list can be used to 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 [Deer90, Dumais93, Dumais95, Berry05]. The basic derivation of LSI is as follows. Let **X** be the term by document matrix

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

$\boldsymbol{t}_i = [x_{i,1}, \ldots, x_{i,m}]$ is the occurrence vector of term i, and $\boldsymbol{d}_j^T = [x_{1,j}, \ldots, x_{m,j}]$ is the vector of document j. The dot-product $\boldsymbol{t_i t_p^T}$ then gives the correlation between terms, and matrix $\boldsymbol{XX}^T$ 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:

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

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

$$\boldsymbol{XX}^T = (\boldsymbol{U\Sigma V}^T)(\boldsymbol{U\Sigma V}^T)^T = (\boldsymbol{U\Sigma V}^T)(\boldsymbol{V\Sigma U}^T) = \boldsymbol{U\Sigma}^2\boldsymbol{U}^T$$
$$\boldsymbol{X}^T\boldsymbol{X} = (\boldsymbol{U\Sigma V}^T)^T(\boldsymbol{U\Sigma V}^T) = (\boldsymbol{V\Sigma U}^T)(\boldsymbol{U\Sigma V}^T) = \boldsymbol{V\Sigma}^2\boldsymbol{V}^T$$

In other words, the columns of $\boldsymbol{U}$ are the eigenvectors of matrix $\boldsymbol{XX}^T$, 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 $\boldsymbol{XX}^T$ or $\boldsymbol{X}^T\boldsymbol{X}$. This can also be denoted as follows:

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

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, LSI has three serious problems. First, the purely matrix factorization derivation of LSI makes the resulting dimensions difficult to interpret. Second, and more important, LSI assumes that words and documents form a joint Gaussian model, which 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 LSI. 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, $\boldsymbol{X} = \boldsymbol{UV^T}$, $u_{ij} \geq 0$ and $v_{ij} \geq 0$. NMF for a matrix is not exclusive and the choice of factorization depends on the divergence of resulting factorization $\boldsymbol{UV^T}$ 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 $\|\boldsymbol{X} - \boldsymbol{UV}^T\|_F$ or equivalently $\frac{1}{2}\|\boldsymbol{X} - \boldsymbol{UV}^T\|_F^2$:

$$J = \frac{1}{2}\|\boldsymbol{X} - \boldsymbol{UV}^T\|_F^2 = \frac{1}{2}tr\big((\boldsymbol{X} - \boldsymbol{UV}^T)(\boldsymbol{X} - \boldsymbol{UV}^T)^T\big)$$
$$= [tr(\boldsymbol{XX}^T) - 2tr(\boldsymbol{XVU}^T) + tr(\boldsymbol{UV}^T\boldsymbol{VU}^T)]$$

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:

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

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

Similarly to LSI, 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

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

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 2.1. 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 $\|\boldsymbol{x} - \boldsymbol{y}\|^2 = (\boldsymbol{x} - \boldsymbol{y})^T (\boldsymbol{x} - \boldsymbol{y}) = \|\boldsymbol{x}\|^2 + \|\boldsymbol{y}\|^2 - 2\boldsymbol{x}^T \boldsymbol{y} = -2cos < \boldsymbol{x}, \boldsymbol{y} >$.

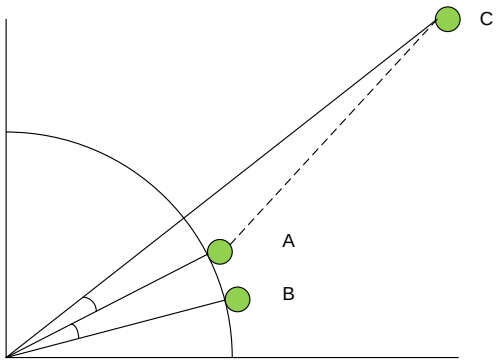


Figure 2.1 Normally Euclidean distance and cosine give unrelated scores for vectors. It is demonstrated by the fact that $\angle AC$ is incompatible to the measure of $\overline{AC}$; on the other hand, for normalized vectors, the two metrics function equally.

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 [Duda01]. 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 2.2.

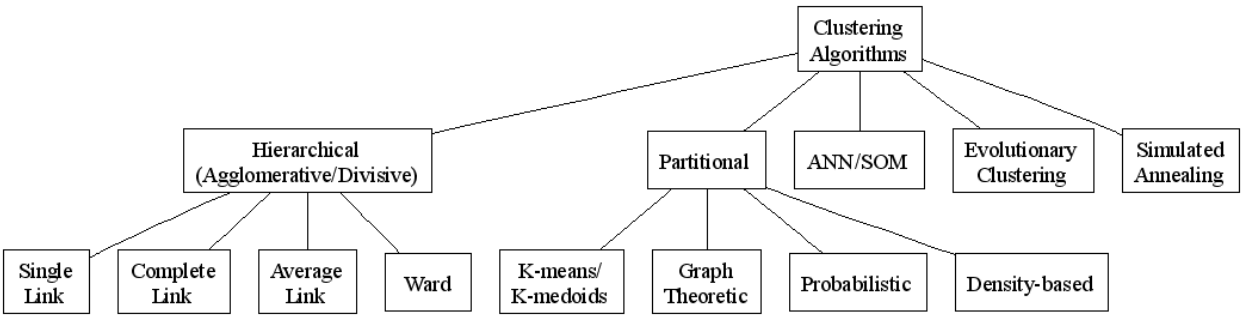


Figure 2.2 Classification of clustering algorithms.

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 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 2.3, 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.

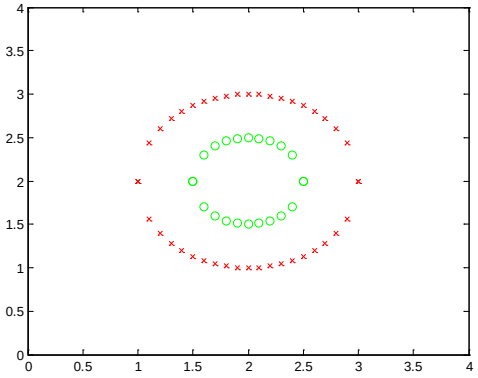


Figure 2.3 The versatility of single-link clustering.

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 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 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 dissertation 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)

2.8.1 Introduction

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- 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, 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 2.4 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.

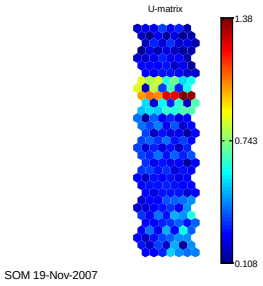


Figure 2.4 The U-matrix of Iris data.

2.8.2 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 the 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,

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

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.

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).

2.8.3 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) $\Phi(k) = \sum_j \left|\left\{i \middle| \|m_i - m_j\| > k; C_{ij} = 1\right\}\right|$, 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 the granularity of a set of clusters than it does to generate actual clusters.

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 $c_1, c_2, \ldots, c_K$, 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 within a cluster. 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 2.5 at successive values of K.

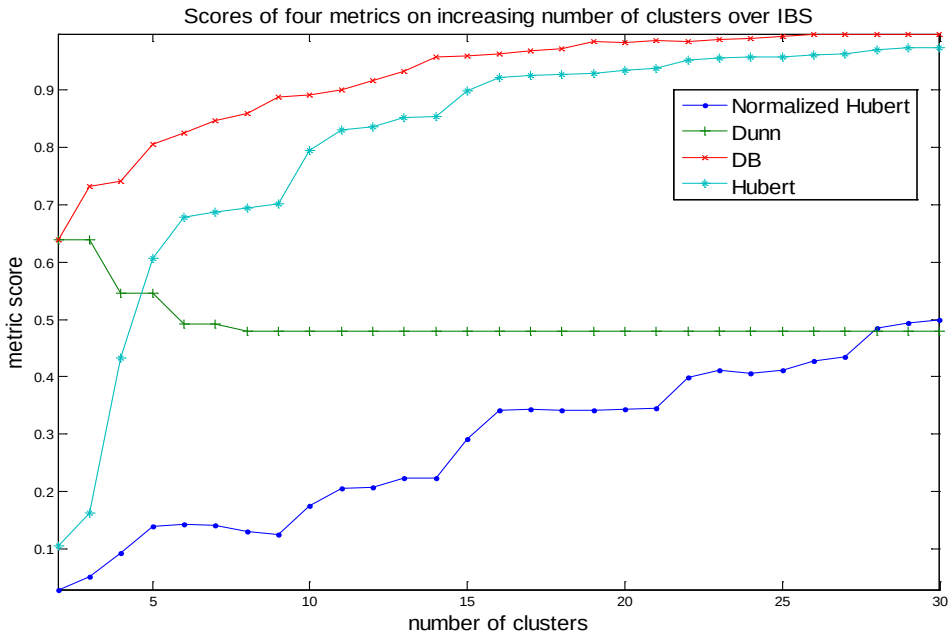


Figure 2.5 Score curves of DB, Dunn, Hubert, and normalized Hubert for IBS.

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 2.5, 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 such that Rand index is too high, and Jaccard index is too low, and unfortunately no great guidelines exist for index choice.

One problem with the 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, selected 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, a preliminary study was conducted to investigate the use of traditional term-based clustering algorithms within the requirements domain.

CHAPTER 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 $P(w_i|z), i = 1,2,\dots,T, and\ \sum_i w_i = 1$. 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.

3.1.1 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 $X = \{\boldsymbol{x_1}, \boldsymbol{x_2}, \dots, \boldsymbol{x_n}\}$, 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 $\Sigma = E[(\boldsymbol{x}-\boldsymbol{\mu})(\boldsymbol{x}-\boldsymbol{\mu})^T] = \int(\boldsymbol{x}-\boldsymbol{\mu})(\boldsymbol{x}-\boldsymbol{\mu})^T p(\boldsymbol{x})d\boldsymbol{x}$, the MLE of $\boldsymbol{\theta}$ calculated by the vanishing gradient of L are:

$$\widehat{\boldsymbol{\mu}} = \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{x_i}$$

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

3.1.2 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 $x_1, x_2, \ldots, x_n \in I \; and \; \lambda_1, \lambda_2, \ldots, \lambda_n \geq 0 \; with \; \sum_i \lambda_i = 1$, 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:

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

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 $\ln P(\boldsymbol{X}|\boldsymbol{\theta_n}) = \ln \sum_z \big(P(z|\boldsymbol{X},\boldsymbol{\theta_n}) P(\boldsymbol{X}|\boldsymbol{\theta_n}) \big)$

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

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 $\boldsymbol{\theta}$, :

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

The intuitive way to understand this is: now that the values of **X** and $\boldsymbol{\theta_n}$ are known and $\boldsymbol{\theta}$ is the parameter to be adjusted, then distribution of **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. **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 **Z**.

3.1.3 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 [McLach00]. 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.

3.2 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 **θ**, 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.

3.3 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.

3.3.1 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.

Aspect Model

The model used by PLSA to model documents is called the aspect model, a latent variable model which associates an un-observed topic $z \in Z = \{z_1, \ldots, z_k\}$ 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.

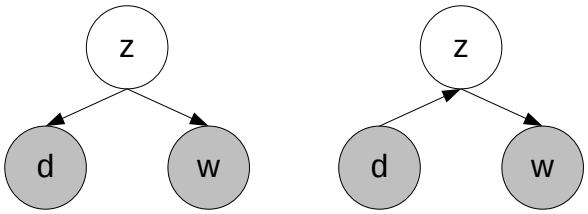


Figure 3.1 Aspect model in the symmetric and asymmetric parameterization.

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:

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

3.3.2 Latent Dirichlet Allocation (LDA)

Modeling

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:

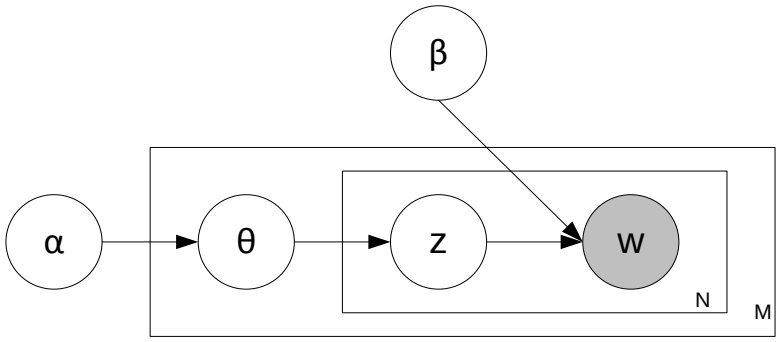


Figure 3.2 Graphical model representation of LDA.

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.

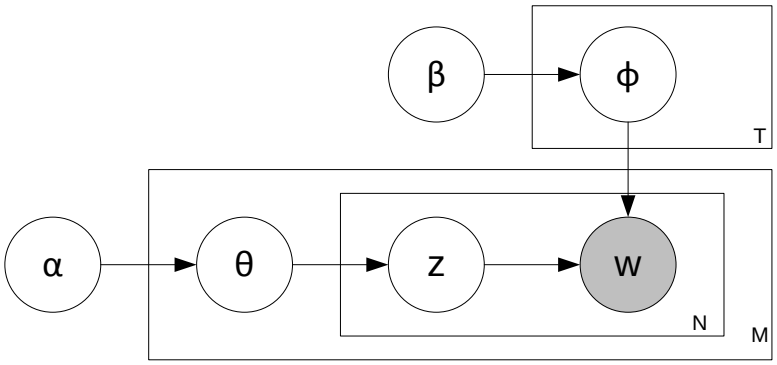


Figure 3.3 Graphical model representation of LDA with Dirichlet Prior on term distributions

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:

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

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

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.

CHAPTER 4. CONSENSUS CLUSTERING

In most cases, a single clustering algorithm inevitably makes sub-optimal decisions at some step in the process. This chapter will discuss the techniques for combining multiple partitions generated from diverse clustering algorithms in order to produce higher quality clusters.

4.1 Introduction

The combination of multiple clusterings, possibly containing different numbers of clusters, in order to achieve better groupings, is inspired by the success of various voting and combination methods in supervised learning. Several weak classifiers can be synthesized into a good performance high end classifier, as shown in boosting algorithms and other combined classifiers [Schapire03]. Similarly, an ensemble of diverse partitions can be combined to arrive at consensus clustering that delivers better performance and robustness. In addition, the usefulness of consensus clustering has been inspired from a number of different domains. For example, in distributed data mining, large quantities of different results are collected in dispersed locations and these results must be collated in order to provide useful knowledge.

4.2 Formulation of the problem

Let $\chi = \{x_1, x_2, \dots, x_n\}$ be the objects to be clustered, P $= \{P^1, P^2, \dots, P^R\}$ be R clusterings, soft or hard, of the data, called a clustering ensemble, where $P^i = \{C_1^i, C_2^i, , , , C_{k_i}^i, \}$ represents the individual clusters, the problem of consensus clustering is to find an optimal partition P^* based on the existing clustering ensemble. The process of consensus clustering can be illustrated in Figure 4.1.

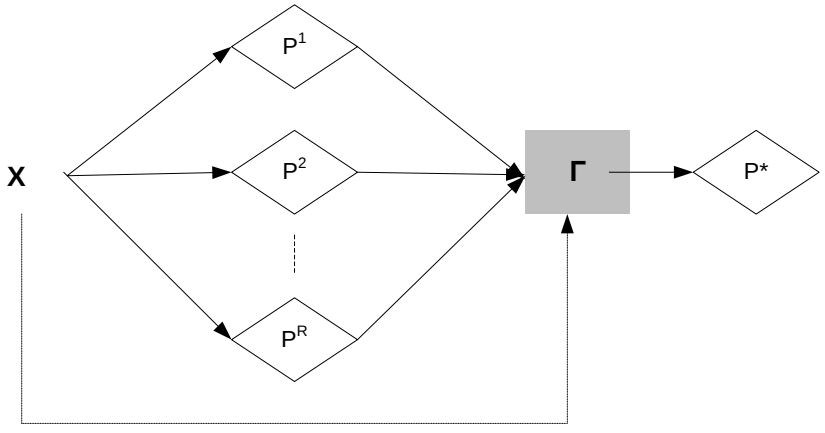


Figure 4.1 A consensus function Γ combines an ensemble to a consistent and robust clustering P*.

A more detailed formulation of the problem, given in [Li04], describes consensus clustering as a categorical clustering. Let $p = \sum_{i=1}^{R} k_i$, then each data x_i can be represented as a p-dimensional binary vector:

$$x_i = (x_{i11}, \dots, x_{i1k_1}, \dots, x_{iN1}, \dots, x_{iRk_R},)$$

$$x_{ijl} = \begin{cases} 1, & x_i \in C_l^j \\ 0, & otherwise \end{cases}$$

A good consensus clustering should at least satisfy the following three properties [Fred05]:

(1) Consistent with the clustering ensemble.
(2) Robust to insignificant variations in P.
(3) P^* fits well with the true allocation of the data for known reference sets.

4.3 Generation of the ensemble

The diversity of the ensemble, as well as the quality of the ensemble, significantly impacts the final consensus. The clustering ensemble can be generated in a number of ways including the following:

(1) Choosing either a subset or a different representation of the features. The former is adopted in one of the experiments of [Strehl02], and the latter is used in [Fern04] in which random projections were used and clustering was performed in the transformed lower dimension.
(2) Adopting different clustering algorithms. This technique is used in [Strehl02], where 10 diverse clustering algorithms were implemented.
(3) For a certain algorithm, varying the parameters, such as the number of clusters and initial configuration. For example, in [Fred04], K-means was run many times where each time a different number of clusters and initial set of centroids were selected. A note to the variation of cluster number, the objective function $NMI_a(P^x, P) = \frac{1}{R}\sum_{i=1}^{R} NMI(P^x, P^i)$ (introduced next) tends to favor the P^x whose cluster number approximates the average number of clusters in ensemble P. Thus, variation of the number of clusters should be used with caution.
(4) Clustering over a sub-sampling of data. The work in [Fern04], for instance, randomly sub-sampled the original data set with a 70% sampling in each round.

4.4 Objective function for consensus optimization

It may appear at first that consensus can be achieved by simple voting to determine cluster membership according to dominant opinion. In reality, however, the decision made by voting violates the property of consistency, as can be illustrated in this simple example. Suppose several hard partitions on three artifacts a, b, and c have be obtained, and consensus clustering is used to vote if any artifact pair belong together by checking their distributions in all the partitions. Then the following voting results for artifact pairs are possible (1/0: together/separate):

	a	b	c
a			
b	1		
c	0	1	

Apparently the consensus through voting is inconsistent since the first two items, ab=1 and ac=0, imply b and c should be separated, which contradicts the third voting result.

Therefore, the precise meaning of consistency is that consensus clustering should share as much information as possible with the clustering ensemble. An objective function serves to measure the fitting between the ensemble and consensus. Many choices exist to measure the pair-wise agreement between two partitions P^a and P^b, the most widely used being normalized mutual information which has been introduced in Chapter 2:

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

Using all the pair-wise NMIs, the average normalized mutual information between an arbitrary clustering P^x and a clustering ensemble P is:

$$NMI_a(P^x, P) = \frac{1}{R}\sum_{i=1}^{R} NMI\left(P^x, P^i\right)$$

and accordingly, the optimal consensus partition P* is defined as:

$$P^* = argmax_x\{NMI_a(P^x, P)\}$$

So the consensus clustering is an optimization problem regarding the above objective function. Many optimization techniques, such as simulated annealing and genetic algorithms, can be used to search the clustering space for a good consensus, but they are computationally expensive and thus unattractive for clustering large data sets. The greedy search [Strehl03, Zhao01] is also a choice, but it has a strong dependency on the initialization, usually resulting in poor local optima. A better approach involves clustering over the space determined simultaneously by the data and the clustering ensemble. This is discussed in the following section.

4.5 Existing methods for ensemble integration

Existing methods of ensemble integration can be generally categorized into instance-clustering and meta-clustering.

Instance-clustering

The instance-clustering still partitions the artifacts but with a different set of proximities rather than the proximities defined on the original features. The core of most consensus instance-

clustering methods is the co-association matrix, a proximity matrix that records the artifact similarity in the ensemble space. The underlying assumption of using a co-association matrix is that artifacts belonging to a "real" cluster are very likely to appear in the same cluster in different partitions. It is an N x N matrix, with each element

$$C(i,j) = \frac{n_{ij}}{R}$$

where n_{ij} is the number of times the artifact pair x_i, x_j is assigned to the same cluster over the ensemble. Notice that since only the proximities between pairs are available, for further clustering, feature-based clustering algorithms are excluded, and usually two types of clustering algorithms are used: hierarchical clustering and graph partitioning. The former is adopted in [Fern03, Fred05, Topchy03], which used single-link or average-link agglomerative hierarchical clustering algorithms [Jain88] to cluster over a co-association matrix. The latter is discussed in [Strehl02, Fern04], which transforms the co-association matrix into a weighted graph represented by G = (V, W), where vertex set V is the set of artifacts and edge weight set $W = \{w_{ij} | w_{ij} = C(i,j)$. Then to partition a graph into K parts is to find K disjoint clusters of vertices $P = \{P_1, P_2, \dots, P_K\}$ where $\cup_k P_k = V$, and to minimize the cut by partition P: $Cut(P,W) = \sum w_{ij}$ where vertex i and j don't belong to the same cluster.

A more generalized co-association matrix is derived by noticing that the clustering can be soft. Let Y^a be a $N \times K$ membership matrix corresponding to partition P^a, where the entry y_{ij} quantifies the degree of membership of artifact i to the cluster j, with constraints $y_{ij} \in [0,1], \sum_{j=1}^{K} y_{ij} = 1$, then individual co-association matrix for Y is:

$$C^a = Y^a (Y^a)^T$$

and the co-association for the clustering ensemble P is:

$$C = \frac{1}{R}\sum_{i=1}^{R} C^i = \frac{1}{R}\sum_{i=1}^{R} Y^i (Y^i)^T$$

The method Probabilistic Label Aggregation (PLA), proposed in [Lange05], applied Probabilistic Latent Semantic Analysis (introduced in Chapter 4) to seek partitioning on the un-normalized generalized co-association matrix $C = \sum_{i=1}^{R} C^i$. As in PLSA, PLSA assumes independency between artifacts conditioned on the cluster variable *v*, namely,

$$p(j|i) = \sum_v p(j,v|i) = \sum_v p(j|v)p(v|i) \text{ and } p(j,i) = \sum_v p(i|v)p(j|v)p(v)$$

The log likelihood of the observations, combined with the membership information in the co-association, can be written as:

$$L = \log p(C|\mathbf{\Phi}) = \sum_{ij} C(i,j)\left(\sum_{v} p(i|v)p(j|v)p(v)\right) = \sum_{ij} C(i,j)\left(p(i)\sum_{v} p(j|v)p(v|i)\right)$$

The parameters **Φ**, including $p(j|v), p(v|i), and\ p(i)$, are estimated by applying EM. Finally, as mentioned in the Chapter 4, the consensus clustering can be derived from these probabilities.

Meta-clustering

The meta-clustering involves clustering of the clusters. The underlying assumption of meta-clustering is that there exists a correspondence structure among different clusters in the ensemble. Still within the graph partitioning framework just introduced, let $V = \{C_i^j | j = 1,2,\dots,N, i = 1,2,\dots,k_j\}$, $W = \{w_{ij} | w_{ij} = s_c(C_x, C_y), C_x, C_y \in V\}$, the graph G = (V,W) can be partitioned into meta-clusters. $s_c(C_x, C_y)$, the similarity between clusters, is normally set to the Jaccard coefficient $s_c(C_x, C_y) = \frac{|C_x \cap C_y|}{|C_x \cup C_y|}$ [Fern04, Strehl03]. The consensus clustering P* can be generated by associating each artifact d to the meta-cluster in which d appears most frequently.

In another meta-clustering method, proposed in [Siersdorfer04], instead of looking for the meta-clusters, an attempt is made to extract a meta-mapping of cluster labeling from the clustering ensemble. The meta-mapping is a bijective function map vector. For every clustering P^i, $map_i: \{1,\dots,K\} \to \{1,\dots,K\}$ assigns each label l to a meta-label $map_i(l)$. Three optimization methods, correlation-based approach, purity-based approach, and association-rules-based approach, are applied to generate the meta-mapping. The final labeling (clustering) of an artifact d is defined as:

$$L(d) = map_i(P^i(d))$$

where $P^i(d)$ is the original label of d in clustering P^i. One limitation of this method that it can only work on ensembles comprised clusterings which have identical numbers of clusters.

Hybrid partitioning

Fern at el. presented a bipartite graph partitioning method to cluster the clusters and artifacts simultaneously, which produces the hybrid meta-clusters [Fern04]. The vertices of the graph consist of both the artifacts and clusters in the ensemble. No edges exist between artifacts or between clusters and an edge weighted 1 connects a cluster C_i^j and an artifact d if d belongs to C_i^j (an idea similar to the formulation in [Li04]), as shown in the Figure 4.2.

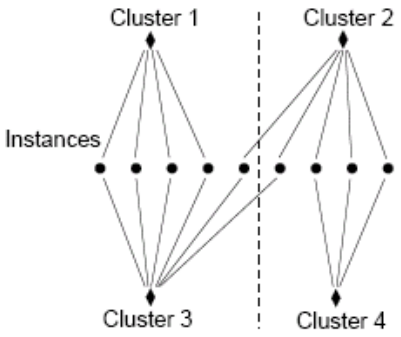


Figure 4.2 A hybrid bipartite graph representation of the clustering ensemble.

The partitioning of a bipartite graph is a well-studied area, for example, as described in [Dhillon01c]. A notable conceptual strength of this bipartite modeling is that it preserves all the information in the ensemble; in other words, the original ensemble can be recovered from this bipartite graph.

Comparison

Instance-clustering over clustering ensemble has a high time complexity $O(N^2KR)$, the product of squared data size, number of clusters, and the size of the ensemble, while most meta-clustering methods exhibit the time complexity raging from $O(NKR)$ to $O(NK^2R^2)$, making the latter more computationally efficient. With regard to quality, the experiments in [Fred04] show the agglomerative hierarchical clustering of ensemble performs better than instance-clustering graph partitioning and meta-clustering graph partitioning. It is also demonstrated empirically in [Fern04] that hybrid bipartite graph partitioning outperforms not only the base clustering methods, but also the instance-clustering graph partitioning and meta-clustering graph partitioning. And the experiments described in [Lange05] show PLA performs competitively with all graph partitioning consensus methods. No other definite comparison work appears in the literature.

SECTION C REQUIREMENTS CLUSTERING RESEARCH

CHAPTER 5. OVERVIEW OF RESEARCH

The research in this dissertation includes not only a series of experiments that are designed to compare the effectiveness of existing clustering algorithms applied to the task of clustering requirements, but more importantly, it also proposes several new approaches to cope with some distinct challenges in the domain of requirements engineering, such as the need to incorporate feedback and to maintain a relatively stable clustering in the incremental setting. Four chapters in this section report on this research, each of which addresses a unique requirements clustering problem.

Chapter 6 discusses the selection of an informative cluster validation metric. Based on the observation of a series of Monte-Carlo simulations, normalized mutual information (NMI) was found to be the most suitable validation metric as it is able to associate highly discriminate scores with clusterings of various qualities.

Chapter 7 addresses the question of "which is the best fully unsupervised clustering algorithm to cluster a set of requirements?" The answer to this question is laid out in four sections. Section 7.1 compares the performance of several popular clustering algorithms when applied individually. Section 7.2 studies clustering based on two matrix decomposition-based term weighting methods, PCA and SVD. Section 7.3 describes two methods for enhancing spherical K-means (SPK). In particular, section 7.3.1 examines the idea of initializing SPK by sampling from Self-organizing maps, and section 7.3.2 introduces a pipeline hybrid clustering model where hierarchical clustering algorithms are first applied, followed by SPK. Section 7.4 reports the experimental results of using consensus clustering. Conceptually, section 7.1 and 7.2 consider each clustering algorithm individually, whereas section 7.3 and 7.4 study the combined use of different clustering algorithms.

Chapter 8 addresses this question: "If user feedback is captured in a special training session then how should the session be structured, and specifically what information should be elicited, in order to optimize the value of the users' input?" The particular type of feedback considered in the research is pair-wise constraint, a type of specification that indicates whether a pair of requirements should be bound to each other or not. A new two-phase framework is proposed to utilize pair-wise constraints for better clustering. The first phase of the framework identifies more informative constraints and the second phase effectively produces constraint-enhanced clustering. Characteristically, this framework resorts to consensus clustering in both phases, and is therefore named consensus-based constrained clustering framework.

Chapter 9 focuses on incremental requirements clustering. It studies how to construct high quality and stable clusters in a dynamic requirements elicitation context. The proposed method

achieves these goals by building a seed preservation strategy into traditional spherical K-means. The experimental results demonstrate the effectiveness of this method.

CHAPTER 6. SELECTING CLUSTER VALIDATION METRIC

The cluster quality metrics introduced in Chapter 2 differ in their abilities in discriminate between high and low quality clusters. To be discriminate, a metric can assign a wide range of scores to indicate various qualities of clusterings, which is important because too narrow a score range makes it hard to tell good clusterings from bad ones. The first set of experiments explored the score ranges of several well-known clustering comparison metrics when applied to measure the agreement between two random clusterings. The second set of experiments was conducted to investigate whether certain quality metrics, in situations where an answer clustering is not available, can indicate the true quality of requirements clusters in a consistent manner.

6.1 Baseline distributions of clustering comparison metrics

In clustering research, the quality of a generated clustering is often measured by clustering comparison metrics (CCM), which compute certain statistics to capture the agreement between the generated clustering and a predefined answer clustering. Two usability issues are associated with clustering comparison metrics in practice. The first is that scores for good clusterings and bad clusterings should be sufficiently separated and the other is that users should be informed about the significance of scores. For an example of the latter issue, if one knew that the worst clustering had a score of 0.9, then 0.95 should not be conceived as an indicator of excellent clustering despite its proximity to the maximum score of 1.0.

The experiments in this section are designed to address the above two issues by looking at the baseline distribution of various metrics i.e. the population comprised of scores calculated between two random clusterings.

Experiment 6.A Distribution of CCMs for random clusterings

Objective

To investigate the shapes of distributions of various CCMs when used to measure the agreement of pairs of random clusterings

Experimental Method

1. A large number P of pairs of random clusterings were generated. Values of P tested in experiments ranged from 500 to 5000 however as they did not result in significant differences only the results with P set to 1000 are reported. Other variables included N, the number of instances, and K, the number of clusters. N was set equal to 1000 and K was set equal to 5 and 50 respectively.

2. Clusterings with empty clusters were abandoned.

3. Metric scores were computed for the five metrics of: Rand, Jaccard, Dice, FM, and NMI.

Results

The two histograms in Figure 6.1 show the distributions of five metrics with N equal to 1000 and K equal to 5 and 20 respectively.

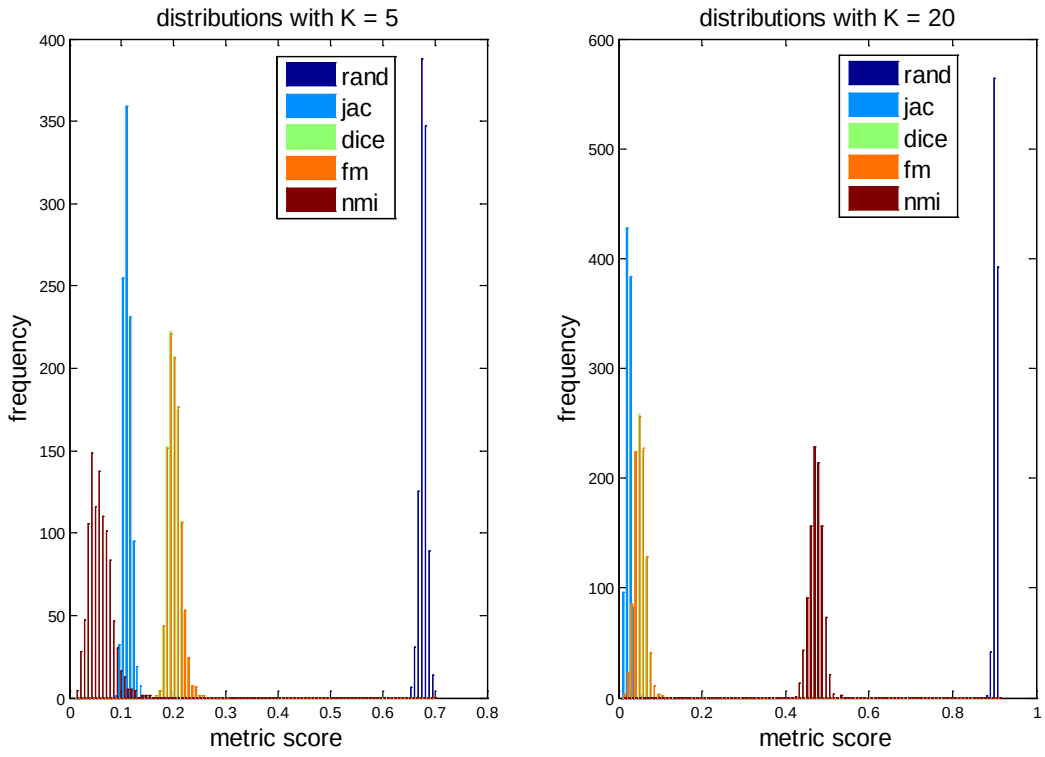


Figure 6.1 Baseline distributions of 5 clustering comparison metrics.

These two plots suggest the following observations for all five metrics:

(1) The scores of each metric are distributed approximately normally.

(2) Variances of distributions are rather small. Distributions at larger K values peak so sharply that variances are ignorable.

(3) Except those of Dice and FM, all distributions are well-separated but the ordering of their means on the x-axis is not fixed.

Conclusions

The five CCM metrics have symmetric distributions in measuring agreement between random clusterings, therefore the sample mean can be fairly used to represent a set of scores with certain N and K.

Experiment 6.B Expected values of metric distributions with different N and K

Objective

To study the trend of metric scores as N and K increase.

Experimental Method

1. The previous experiment was repeated 1000 times for each pairwise combination of N set among the range [200, 1000] at increments of 200, and K from [5, 50] at increments of 5.

2. The average score was computed for each configuration of N and K.

Results

Results are depicted in Figure 6.2. Each subplot corresponds to a metric, and contains several curves each of which represents metric scores with fixed N but increasing K. A close observation reveals two distinctions between NMI and pair counting metrics (PCM), including RAND, JACCARD, DICE, and FM:

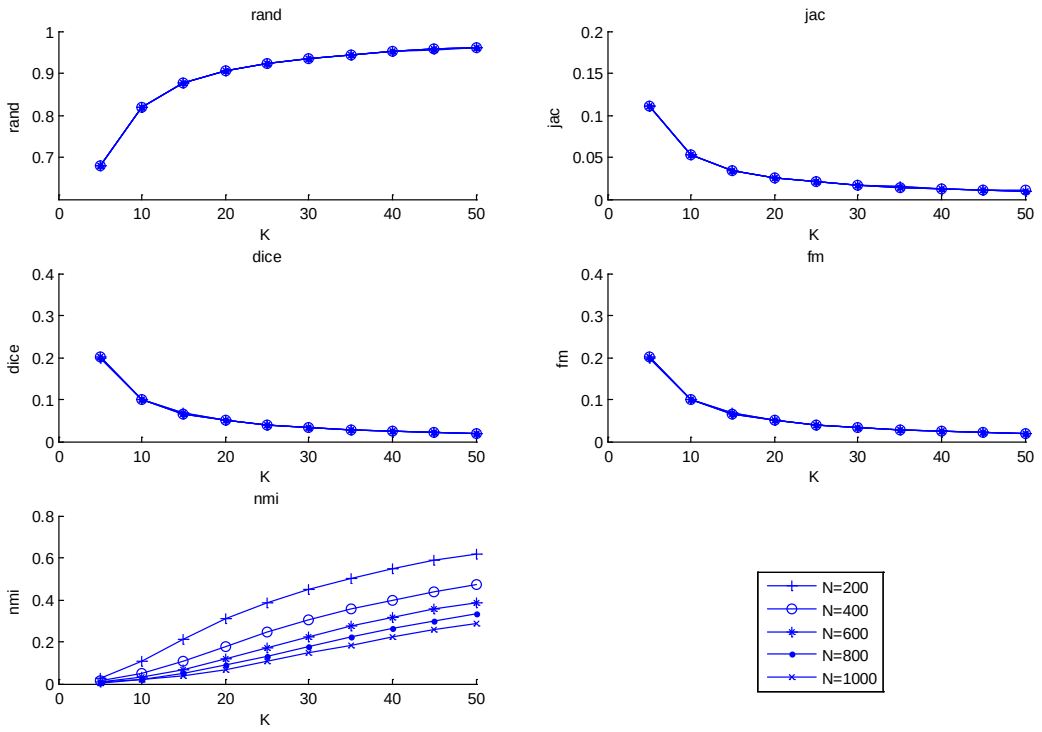


Figure 6.2 Curves of baseline distribution averages with different scores of K and N.

(1) PCM curves consistently go up or down but are independent of N. Their curves for different N values have a high level of overlap to the extent that it's hard to discern them. In contrast, there's clear separation between NMI curves with different values of N.

(2) PCM curves quickly enter into a very narrow range. For example, with only 15 clusters, the scores of RAND have jumped to 0.9 and after then approach 1.0 slowly. On the other hand, NMI scores range broadly between 0.0 and 1.0.

Conclusions

NMI shows a much better discriminate ability than other CCMs. Due to the difficulty of generating a large population of clusterings whose scores of qualities are correctly labeled, a complete measurement of various metrics' ability of discriminate is hard. However, the results and analyses about baseline distributions can at least provide guidelines on how to appreciate the significance of a score calculated by a certain metric.

6.2 Correlation between CCM and non-CCM

Evaluating a clustering without an answer clustering is inevitable in a practical clustering task. This section studies the feasibility of using non-clustering comparison metrics (non-CCMs) other than CCMs to accurately evaluate a clustering. The basic idea is to examine the correlation between CCM and a few non-CCM metrics and to choose the non-CCM quality metrics that exhibit the high correlations with CCM. Two experiments are presented, the first studying the distributions of metrics on a collection of clusterings generated by a two-stage K-means algorithm, and the second studying the correlation between CCMs and non-CCMs.

Algorithm: Two-stage spherical K-means clustering
Input: unlabelled instances $\chi = \{x_i\}_{i=1}^{N}$, number of clusters *K*, initial centroids *I*, convergence condition
Output: crisp *K*-partition $P = \{C_i\}_{i=1}^{K}$.
Steps:
1. Initialization: initialize centroids using *I*: $\{\mu_i^0\}_{i=1}^{K} = I; t = 0$
2. Batch instance assignment and centroid update until convergence
   2a. assign each instance x to nearest cluster i with largest $x^T \mu_i^t$
   2b. update each centroid: $\mu_i^{t+1} = \frac{\sum_{x \in C_i^{t+1}} x}{|C_i^{t+1}|}$
   2c. $t = t + 1$
3. Incremental optimization of objective function until convergence:
   3a. randomly select an instance $x \in C_h^t$, move it to cluster C_v^t that maximizes the gain of objective function $J_{sp} = \sum_{i=1}^{K} \sum_{x \in C_i^t} x^T \mu_i^t$ caused by the moving of x, therefore after the moving $C_h^{t+1} = C_h^t \setminus \{x\}$, $C_v^{t+1} = C_v^t \cup x$
   3b. update each centroid: $\mu_i^{t+1} = \frac{\sum_{x \in C_i^{t+1}} x}{|C_i^{t+1}|}$
   3c. $t = t + 1$

Figure 6.3 Two-stage spherical K-means clustering.

Experiment 6.C Distributions of CCMs and non-CCMs on non-random clusterings

Objective

To study the distributions of clustering comparison metrics and non clustering comparison metrics, when both classes are applied on the clusterings generated from a specific clustering algorithm.

Experimental Method

(1) The following metrics were selected: NMI, RAND, and JACCARD; non-CCMs included two cohesion metrics, I_1 and I_2, a coupling metric, E_1, and two hybrid metrics, Hubert Γ index and C-Hubert. Most metrics have been introduced in Chapter 3 but C-HUBERT needs explanation. Unlike the original Hubert index, which calculates the correlation between a clustering and original inter-instance proximity matrix, C- Hubert computes the correlation between a clustering and co-association matrix, another proximity matrix that is generated from a set of clusterings, and is potentially more accurate in representing true similarity. The details of the co-association matrix will be discussed in Chapter 9.
(2) A test clustering set with size 1000 was generated by a two-stage spherical K-Means clustering algorithm (described in Figure 6.3) whose initial centroids were selected by sampling instance vectors.

Results

Figure 6.1 shows the distributions of all metrics calculated in eight data sets as histograms. In the results for STUDENT, despite the randomness introduced by the centroid selection, it appears that all distributions are approximately normal, with ignorable variances, which allows one to use average of a set of metric scores as a representative value. However, closer inspection of the other seven data sets gives rise to different suggestions. It is observed that the distributions of NMI, RAND, and JAC remain similar through all data sets but not all non-CCMs have similar distributions. A distributional statistic, skewness is introduced here to quantify findings from Figure 6.5. Skewness is defined as the third moment from the mean of a set of samples, normalized by the cube of the standard deviation, namely, $skewness = E[(x-\mu)^3]/\sigma^3$, where μ represents the sample mean and σ the sample standard variance. Intuitively, skewness impacts the eligibility of representing a set of metric scores by its mean. The skewness scores of 6 metrics are listed in Table 6.1.

Generally, compared with NMI, non-CCMs are seriously skewed. From the perspective of clustering, however, the high level of skewness is desired – it implies most clusterings try to generate optimal clusterings, therefore pushing cohesion (I_1 and I_2) and similarity-partition-correlation (HUBERT and C-Hubert) right skewed, and coupling (E_1) left skewed. It is also worth noting that, in contrast to most TREC data sets, STUDENT and SUGAR don't have high

skewness scores, a potential indicator that these two requirement data sets are difficult to cluster using standard spherical K-means algorithm.

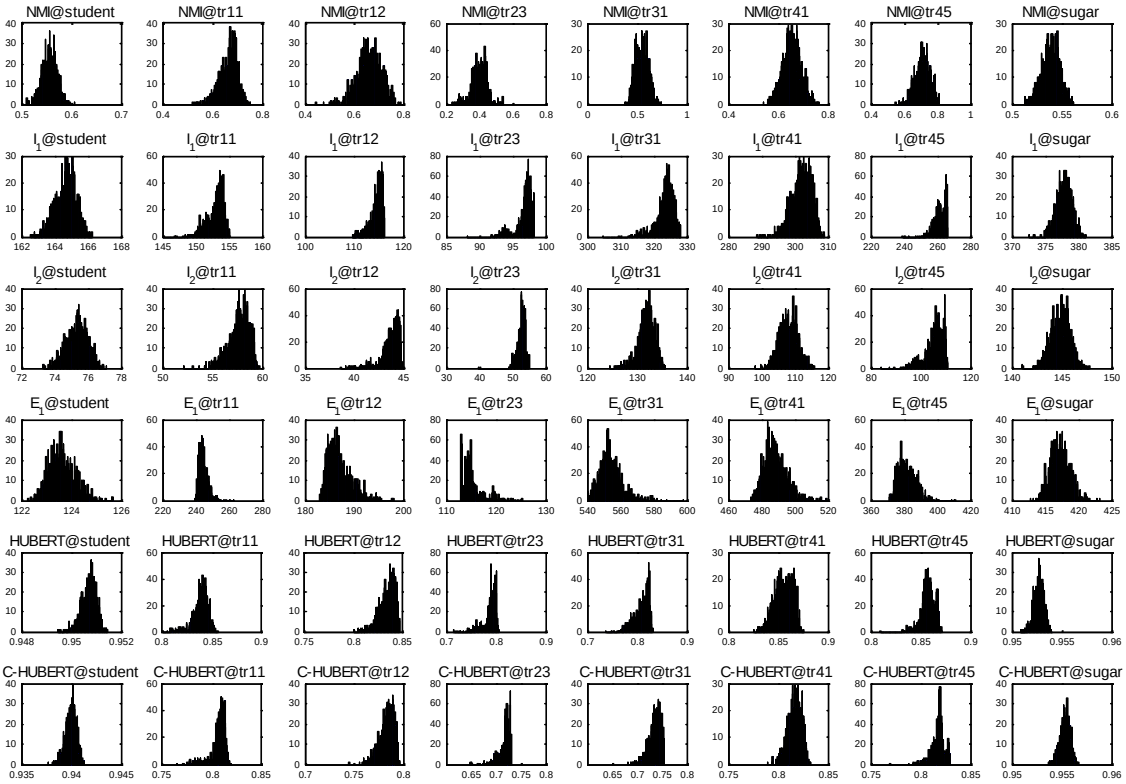


Figure 6.4 Distributions of 6 metrics on 8 data sets

Table 6.1 Skewness scores of 6 metrics on 8 data sets

Data	***NMI***	***I1***	***I2***	***E1***	***HUBERT***	***C-HUBERT***
STUDENT	-0.1358	-0.3577	-0.2689	0.4476	-0.5665	-0.4801
TR11	-0.7268	-1.2295	-0.8984	1.3995	-1.3580	-1.8945
TR12	-0.4456	-1.3910	-1.7770	1.0030	-0.9807	-0.8647
TR23	-0.3361	-1.8204	-1.8483	1.7647	-1.9760	-2.1318
TR31	0.1542	-1.5317	-0.7403	1.1427	-1.0789	-0.8617
TR41	0.0498	-0.7320	-0.0562	0.7784	-0.3151	-0.4710
TR45	-0.2309	-1.1155	-1.3272	0.8639	-0.9698	-1.2558
SUGAR	-0.1300	-0.3013	-0.2232	0.2345	-0.4019	-0.3192
mean	-0.2251	-1.0598	-0.8924	0.95428	-0.9558	-1.0348

Conclusions

On the clusterings produced by an effectively clustering algorithm, CCMs are usually distributed normally whereas non-CCMs are skewed. The degree of skewness often correlates with the ease of the clustering.

Experiment 6.D Correlation between CCMs and non-CCMs

Objective

To identify, if any of the clustering comparison metrics has high correlation with clustering comparison metrics.

Experimental Method

Following the two steps in Experiment 6.C, the CCMs are calculated between each generated clustering and answer clustering, whereas the non-CCMs are only computed on generated clusterings. The standard Pearson correlation is then applied between each comparison metric and non-comparison metric.

Results

Although the complete correlation results are not shown here, a very strong positive correlation was found between members inside three groups: the one including NMI, RAND, and JAC, the one including I_1 and I_2, and the one including HUBERT and C-HUBERT. Therefore for clarity of presentation, the following discussion focuses on three correlations: NMI and I1, NMI and E1, and NMI and HUBERT. Their scores are listed in Table 6.2.

Table 6.2 Correlation between metrics

Data	*NMI with I1*	*NMI with E1*	*NMI with HUBERT*
STUDENT	0.4761	-0.4100	0.1813
TR11	0.4108	-0.1883	-0.2240
TR12	0.3980	-0.2324	-0.2532
TR23	0.3414	-0.4264	0.3052
TR31	0.0143	-0.0849	-0.2892
TR41	0.3299	-0.0915	-0.2321
TR45	0.5752	-0.4988	-0.2202
SUGAR	0.4002	-0.4035	0.1457

There are only moderate positive or negative correlations between CCMs and non-CCMS, meaning that without an answer clustering, measuring clustering quality by non-comparison metrics is unfortunately not accurate. In fact, if such a metric does exist, then as classical K-means uses I_1 as objective function, a K-means using the ideal CCM will give us a close to perfect clustering.

Conclusions

This problem of finding an effective CCM, or broadly put, the evaluation of a clustering without an answer clustering, is very challenging, which in part explains why in industry the manual evaluation of automatically generated clustering is still an indispensible part of the cluster review process.

CHAPTER 7. UN-SUPERVISED REQUIREMENTS CLUSTERING

This chapter reports experimental results from applying a wide variety of standard and enhanced clustering techniques to the requirements problem. Section 7.1 concentrates on a number of single clustering algorithms applied to raw data. Section 7.2 investigates clustering on the data transformed by various term weighting schemes. Section 7.3 discusses two enhancement methods to SPK: SOM sampling based SPK and pipeline clustering; in the latter, output clustering from one clustering algorithm becomes the input to another clustering algorithm, which produces the final clustering that can overcome certain weakness found in single clustering algorithms. Section 7.4 studies another combination of multiple clusterings, consensus clustering, where a voting among a set of clusterings leads to enhanced clustering quality.

7.1 Requirement clustering using basic hierarchical and partition-based algorithms

This section discusses the application of a number of popular basic clustering algorithms to requirements. The best result will be established as a baseline against which various enhancement methods can be compared. Two categories of clustering algorithms, hierarchical algorithms and spherical K-means, are included. Each category and associated experiment will be described and the results will be analyzed.

Experiment 7.A Clustering using hierarchical clustering algorithms

Objective

To study the effectiveness of basic hierarchical clustering algorithms

Experimental Method

Four agglomerative hierarchical clustering (AHC) algorithms and a divisive hierarchical clustering, bisecting hierarchical algorithm were experimentally evaluated. The general framework of AHC algorithms and a detailed description of the bisecting algorithm were presented earlier in section 2.12.2. The four variations of AHC were single-link, complete-link, average-link, and centroid-link. The only difference between them was the similarity measurement between a pair of clusters. The similarity for each hierarchical algorithm is defined as follows:

$$S_{single}(C_i, C_j) = \max_{x \in C_i, y \in C_j} s(x, y)$$

$$S_{complete}(C_i, C_j) = \min_{x \in C_i, y \in C_j} s(x, y)$$

$$S_{average}(C_i, C_j) = \frac{1}{n_i n_j} \sum_{x \in C_i, y \in C_j} s(x, y)$$

$$S_{centroid}(C_i, C_j) = s(\mu_i, \mu_j)$$

where $s(x, y)$ is the similarity function between two instance vectors, n_i is the number of instances, and μ_i represents the centroid of cluster C_i.

Results

The results are shown in the table 7.1.

Table 7.1 NMI scores of clustering of 8 data sets using 5 hierarchical algorithms.

Data	*Random*	*Single*	*Complete*	***Average***	*Centroid*	***Bisecting***
STUDENT	0.3034	0.1543	0.3859	0.4915	0.1460	0.5076
TR11	0.0460	0.0414	0.5837	0.6736	0.0522	0.6777
TR12	0.0406	0.0782	0.4517	0.4534	0.0560	0.7197
TR23	0.0354	0.1995	0.3917	0.4333	0.1180	0.4087
TR31	0.0137	0.0193	0.5655	0.5882	0.0231	0.6736
TR41	0.0213	0.0262	0.5645	0.6033	0.0419	0.6674
TR45	0.0409	0.0350	0.5851	0.5379	0.0318	0.6158
SUGAR	0.2003	0.0767	0.2884	0.4759	0.0763	0.4832

Of the four AHC algorithms, average-link generally outperformed the others, but was inferior to the bisecting algorithm in most data sets.

Conclusions

As the bisecting algorithm actually uses K-means (K equals to 2) during each division, these experimental results demonstrate that proximity-based hierarchical clustering algorithms are not effective in directly clustering high-dimensional and noisy documents such as requirements. This conclusion is consistent with the results reported in a number of document clustering research papers [Zhao01].

Experiment 7.B Clustering using spherical K-means (SPK)

Objective

To study the effectiveness of SPK in clustering requirements

Experimental Method

A version of SPK has been described in section 6.1.2, where it was used to generate a population of clusterings. It is prefixed by “two-stage” because unlike the traditional K-means that only completes batch assignment of instances, it includes an additional incremental optimization phase during which instances are randomly moved between clusters in order to improve the objective function. In this experiment a more thorough study of SPK was performed, involving four variations of SPK. Each variation responded to a combination of two variables: the presence or absence of the second optimization stage and the method of seeding – initial centroids were either generated as random vectors or taken as samples from data vectors. Since results produced

by SPK were non-deterministic, each variation was run 200 times and the minimum, mean, and maximum of their NMI scores were reported in table 7.2.

Table 7.2 NMI scores of four variations of SPK clustering.
SPK-RI (random seeding plus incremental optimization), SPK-R (random seeding), SPK-SI (sampling seeding plus incremental optimization), SPK-S (sampling seeding)

Data	SPK-RI			SPK-R			SPK-SI			SPK-S		
	min	*mean*	*max*	*Min*	*mean*	*Max*	*min*	*mean*	*max*	*min*	*mean*	*max*
STUDENT	0.52	0.56	0.59	0.34	0.38	0.43	0.51	0.56	0.59	0.40	0.45	0.51
TR11	0.49	0.65	0.72	0.33	0.51	0.66	0.57	0.66	0.74	0.40	0.58	0.73
TR12	0.51	0.64	0.76	0.26	0.45	0.63	0.54	0.67	0.77	0.32	0.51	0.68
TR23	0.24	0.39	0.48	0.17	0.30	0.44	0.25	0.40	0.54	0.18	0.34	0.48
TR31	0.40	0.54	0.69	0.39	0.51	0.68	0.39	0.55	0.72	0.38	0.53	0.70
TR41	0.56	0.64	0.74	0.46	0.59	0.70	0.55	0.66	0.76	0.49	0.62	0.72
TR45	0.60	0.71	0.80	0.48	0.63	0.77	0.57	0.71	0.81	0.48	0.65	0.77
SUGAR	0.52	0.54	0.56	0.35	0.40	0.44	0.51	0.54	0.56	0.38	0.42	0.47

Results

As can be observed from table 7.2, the differences between random seeding and sampling seeding are not significant; whereas incremental optimization greatly impacts the quality of clustering. Although all shown here, the runtime performances of all the four variations were comparable to each other.

Conclusions

Based on the results just described, two-stage SPK was shown to be generally better than any of the other algorithms, and therefore in the latter chapters it will serve as a baseline algorithm.

Further discussion

A key question worth discussing is when incremental optimization of second stage SPK can significantly improve clustering. Table 7.3 juxtaposes the main properties of the studied data sets and the NMI scores of clustering for them in both SPK-S and SPK-SI. Some interesting correlations can be observed by ordering the rows in the table by **Δ**, the mean NMI difference between the SPK and SPK-I scores.

Strong negative correlations were found between Δ and both |d| and n_w. In other words, the smaller the average size of documents, or less rich the vocabulary of a data set, the more beneficial the incremental optimization becomes when using spherical K-means. In fact as previously stated, this terseness is a characteristic of most requirements data sets. Therefore it is almost mandatory to include incremental optimization in K-means clustering of software requirements.

Table 7.3 Performance of SPK-S and SPK-SI.
N is the number of documents, K is the number of clusters, |d| is the average number of distinct terms of documents, and Δ is the difference of NMI score between SPK and SPK-I.

Data set	N	K	|d|	n_w	SPI	SPK-I	Δ
Tr31	927	7	132	10128	0.53	0.55	0.02
Tr41	878	10	87	7454	0.62	0.66	0.04
Tr45	690	10	69	8261	0.65	0.71	0.06
Tr23	204	6	34	5832	0.34	0.40	0.06
Tr11	414	9	46	6429	0.58	0.66	0.08
STUDENT	366	29	8	908	0.45	0.56	0.11
SUGAR	1000	40	27	3463	0.42	0.54	0.12
Tr12	313	8	39	5804	0.51	0.67	0.16

The correlations between N, K, |d|, n_w, and **Δ** are shown in table 7.4.

Table 7.4 Correlations between data properties and NMI difference of SPI-S and SPK-SI.

	N	K	|d|	n_w	Δ
N	1	0.3271	0.5766	0.3499	-0.3833
K		1	-0.5332	-0.7432	0.4499
|d|			1	0.8999	-0.7508
n_w				1	-0.6716
Δ					1

7.2 Clustering after indexing through PCA and SVD

As previously mentioned, requirements data sets are high-dimensional and sparse, containing high portions of noisy information that make clustering especially difficult. These problems are not new in traditional multivariate analysis and data mining areas. One of the effective approaches to mitigate the problem is to apply linear dimension transformations, such as PCA or SVD, to the original data so that each data instance is represented in a different space, often with a fewer number of dimensions. This ameliorates the "curse of high dimensionality" [Duda01], filters out noisy information, or discovers hidden semantic relationships. This section discusses the application of PCA and LSI as a means of providing better filtering of noisy information in order to produce better clustering. The basic formula of PCA is first reviewed (the formula for LSI was previously introduced in chapter 2), followed by the description of the experiment design and results.

Basic formulation of PCA

PCA retains only the dimensions that can preserve most of the variances in the data. Fewer dimensions directly attack the "curse of dimensionality" because the analyses in a much lower-dimensional space are more efficient and accurate. In PCA, the covariance matrix of the mean-centered original data matrix is decomposed into the product of its eigenvectors and eigenvalues:

$$\boldsymbol{C} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{U}^T = [\boldsymbol{u}_1, \boldsymbol{u}_2, \dots, \boldsymbol{u}_l] \begin{bmatrix} \lambda_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_l \end{bmatrix} \begin{bmatrix} \boldsymbol{u}_1^T \\ \boldsymbol{u}_2^T \\ \dots \\ \boldsymbol{u}_l^T \end{bmatrix}$$

Since a covariance matrix is always symmetric, the eigenvalues are all positive. By keeping only K largest eigenvalues and multiplying the approximated covariance matrix and mean-centered original data matrix, the representation of data in the space of subset eigenvectors of covariance matrix can be obtained:

$$\boldsymbol{X}_K = \boldsymbol{X}\boldsymbol{C}_K$$

In the application of PCA, one especially important question is how to choose the number of eigenvalues to retain. The metric of loading, defined as $\sum_{i=1}^{K} \lambda_i / \sum_{i=1}^{l} \lambda_i$, measures the extent of preservation. Smaller loading means a higher "compression" rate and lower level of preservation of variance, and vice versa.

Two experiments were designed. The first studied the concentration of loading in various decompositions, and the second investigated the effectiveness of clustering after indexing through these decompositions.

Experiment 7.C Concentration of loading through PCA/LSI

Objective

To study loading with an increasing number of preserved ranks in different decompositions

Experimental Method

Four decompositions were conducted. The first was N-SVD, a SVD on the data that have been weighted by tf-idf and normalized. The second was O-SVD; it was a SVD applied on the original data (without tf-idf and normalization). The third N-PCA and fourth O-PCA, both using PCA instead of SVD, were differentiated in the same way as N-SVD and O-SVD. Loading trends with an increasing number of ranks were recorded for each decomposition.

Results

The results are depicted in Figure 7.6. Four observations can be made

(1) For all data sets, the concentration of loading in the first few largest eigenvalues decreases in the order: O-PCA, O-SVD, N-PCA, and N-SVD.

(2) Loading of SVD increases linearly with increasing ranks for all data sets.
(3) The differences between O-SVD and N-PCA are very small in STUDENT, SUGAR, and TR41, but significant for the other 6 data sets.
(4) High concentration of loading of PCA and SVD is much less prominent on STUDENT, SUGAR, and TR41 than on other six data sets, which have such high concentrations on first a few eigenvalues that the curves appear flat.

Conclusions

The variances of requirements data sets are difficult to represent using only a few dimensions.

Experiment 7.D Clustering of requirements that indexed through PCA/LSI

Objective

This experiment was not designed to select appropriate loading. Instead, the general purpose was to study if indexing by SVD or PCA at some loading could improve clustering.

Experimental Method

In each decomposition, the total number of ranks denoted as K, 20 intervals were chosen from K/2 to K. For example, in SVD of SUGAR data, the original dimensionality was 1000, so the length of each rank interval would be 500/20=25. Then starting at half the total ranks, each iteration added 25 more ranks to form a new matrix X_k, an approximation of the matrix without decomposition, followed by 50 clusterings on X_k using SPK, and the resultant NMI scores were averaged.

Results

The results of clustering on decompositions with different setup methods and ranks are shown in the Figure 7.7. The X-axis represents the loadings and the Y-axis represents the average of NMI scores of 50 clusterings for certain loading. Because after decomposition the highly sparse data matrix becomes highly dense (almost 100%), the time cost of clustering skyrockets, only the results for N-SVD and N-PCA are reported.

Some interesting findings emerge from inspecting Figure 7.7 and comparing it to Figure 7.6:

(1) In general SVD outperforms PCA, but the difference varies on different data sets. In particular on two requirement data sets STUDENT and SUGAR, the differences are trivial.
(2) Unlike the curves of loading in Figure 7.6, the qualities of clustering on both SVD and PCA fluctuate when loading increases. However, after closer inspection, it can be found most of the time the fluctuations of SVD and PCA are synchronized.

Consider now the improvement of best decomposition N-SVD over baseline SPK. TR11 stands out as its N-SVD results are far better than baseline. TR23, TR31, and TR45 have better results of N-SVD most of time, whereas STUDENT, TR12, TR41, and SUGAR have worse results for N-SVD most of time.

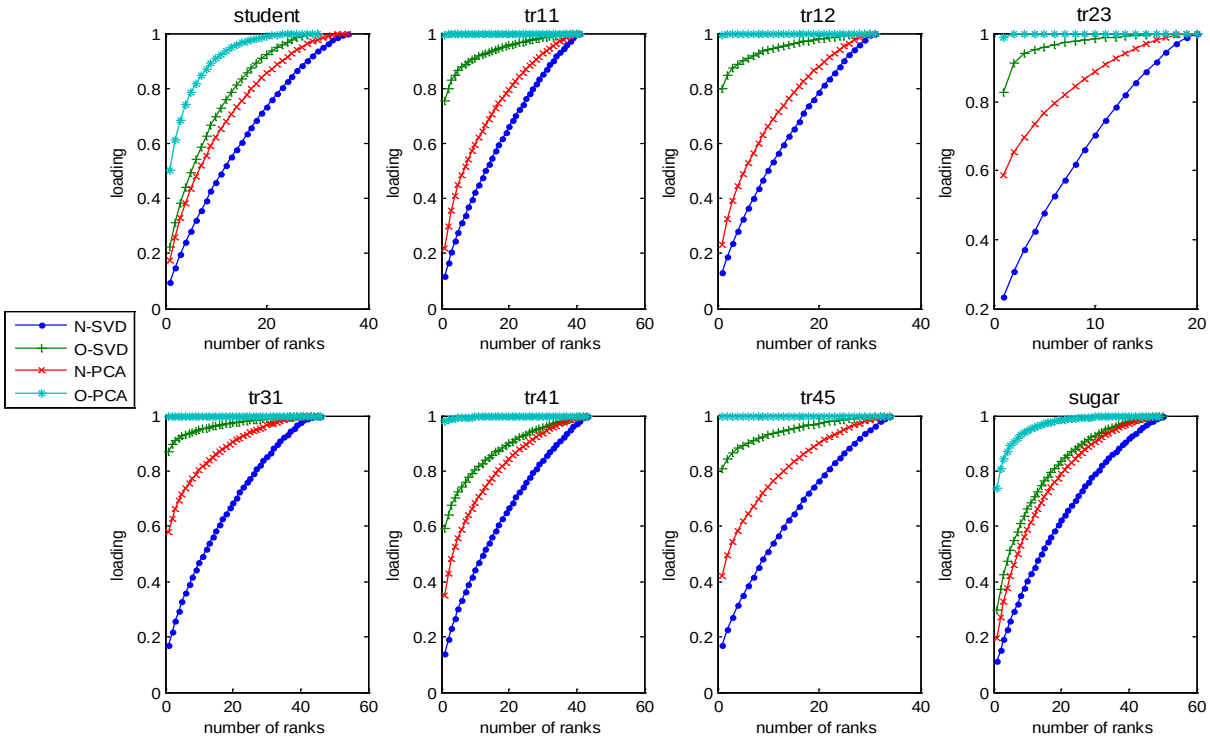


Figure 7.1 Loading curves for four decompositions.

Conclusions

The effects of clustering over indexing by SVD/LSI vary across different data sets.

Further analysis

Given the mixed experimental results on clustering over decomposition, it is worth asking when to cluster over decomposition and when not to. As displayed in Figure 7.1, in data STUDENT, TR41, and SUGAR, the concentration of loading is less outstanding, which implies the variances of the data as a whole are relatively spread in many directions, making it difficult to find a good approximation to summarize the variance. Figure 7.2 then shows that the decompositions of these three data sets give the worst clustering quality. This correlation suggests that data with initial low level concentrations of loading are unlikely to benefit from weighting by decomposition.

However, considering the huge increase of computational cost of using PCA and SVD both before and during clustering, the approach of clustering after matrix decomposition is not recommended when clustering software requirements.

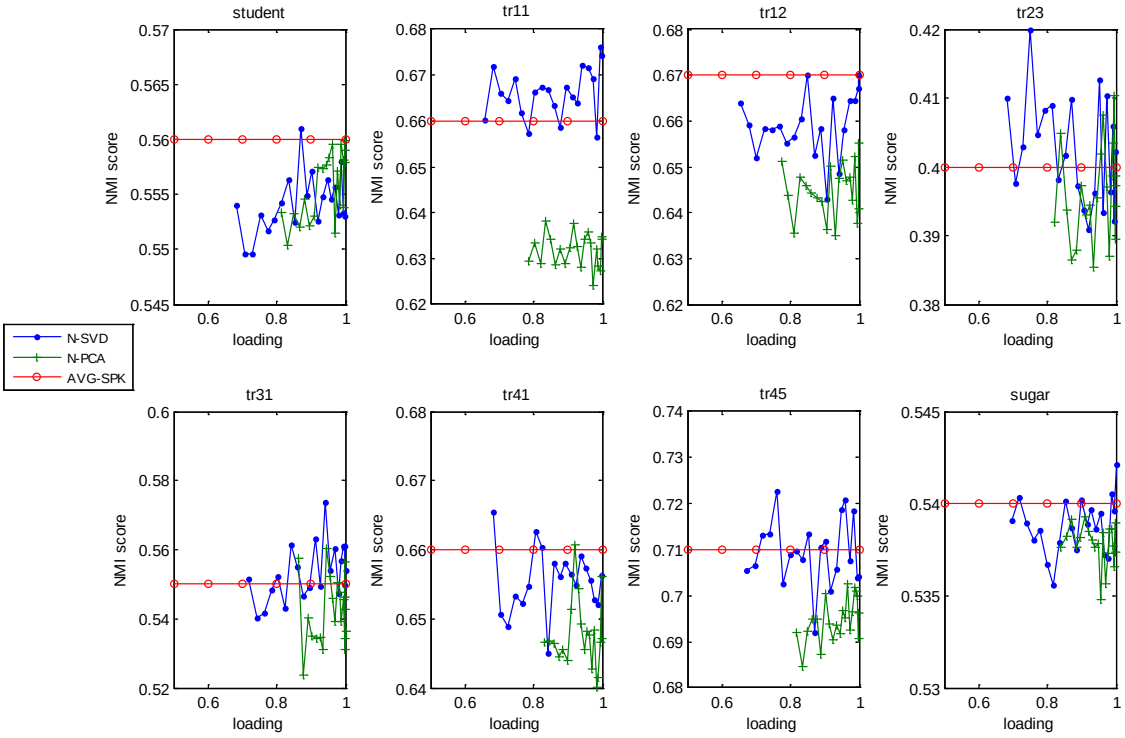


Figure 7.2 The NMI scores obtained by clustering over decompositions of various loadings.

7.3 Enhancing spherical K-means

7.3.1 Introduction

The previous discussion suggests SPK performs better than other crisp clustering algorithms on average. It is well known that SPK's performance strongly depends on the choice of initial centroids – the seeding problem; bad choices of seeding will lead to convergence to poor local minima. The question remains: is there a better initialization approach that can enhance SPK?

Several strategies and approaches have been proposed. They include random selection [Pena99], a genetic algorithm [Pena99], multiple sub-sampling [Bradley98], pre-clustering by hierarchical methods [Meila98, Zhang], and pre-processing with a self-organizing map (SOM) [Bacao05]. Most of these enhancements serve as a pre-processing step that attempts to constrain the space of initial centroids so as to avoid poor local optima. Unfortunately, most of these methods have never been applied to high-dimensional data such as software requirements.

The work presented in this section proposes two new methods for enhancing SPK in high-dimensional data. Section 7.4.2 discusses the use of SOM sampling for SPK initialization and section 7.4.3 discuses a pipeline model in which SPK clustering is conducted over an intermediate clustering from hierarchical algorithms.

7.3.2 Sampling SOM array for SPK seeding

Several researchers have investigated techniques for directly clustering the trained SOM [Ambroise00, Ciampi00, Flexer01, Laerhoven01, Vesanto00], but their purpose is not to improve clustering quality, but rather to prove that, without significantly decreasing clustering quality, clustering of SOM units has smaller time complexity than clustering of the original data. It should be noted that these methods are clustering SOM units, instead of the original data. In contrast, the method proposed in [Bacao05] uses an entire SOM training array as initial centroids for K-means where each SOM unit actually becomes a centroid. This method is questionable because, as well known, the quality of SOM itself degrades when the size of the array is not large enough.

The proposed approach described here is a compound use of SOM and SPK to overcome the problem of local minima. Instead of training a small-scale SOM with all units being centroids, this approach trains a SOM with sufficient scale and then samples part of the SOM units as centroids. The theoretical explanation of this idea is associated with SOM's underlying objective function [Kaski97]:

$$J_{SOM} = \sum_{k}\sum_{i} h_{ci}\|x_k - m_i\|^2 = \sum_{k}\|x_k - n_c\|^2 + \sum_{i}\sum_{j} h_{ij}N_i\|n_i - m_j\|^2$$

The far right-hand side of this formula shows the tradeoff SOM makes: the first term is equal to the objective function of K-means, while the second term represents the sum of distances between nearby centroids. This balance can also define what optimal centroids are: they should not be too distant from each other, otherwise some may appear as outliers. The centroids should not be too close either, because that would cause a redundant decomposition of the instance space.

In practice, the direct reuse of the entire SOM array is problematic because the number of units needed to train an accurate SOM is much larger than the number of clusters [Vesanto00]. A sampling method is proposed here to exploit the trained units from SOM without degrading the quality of the SOM. It runs as follows in 2 steps to cluster requirements into K clusters:

(1) Train the SOM using Euclidean distance. Since all matrices have been normalized to an Euclidean norm of 1, training a Euclidean distance based SOM is only slightly different from training a SOM based on Cosine direction.
(2) Sample K units from the SOM array, each of which is randomly selected from K areas partitioned from a SOM. Execute SPK by setting these samples as seeds.

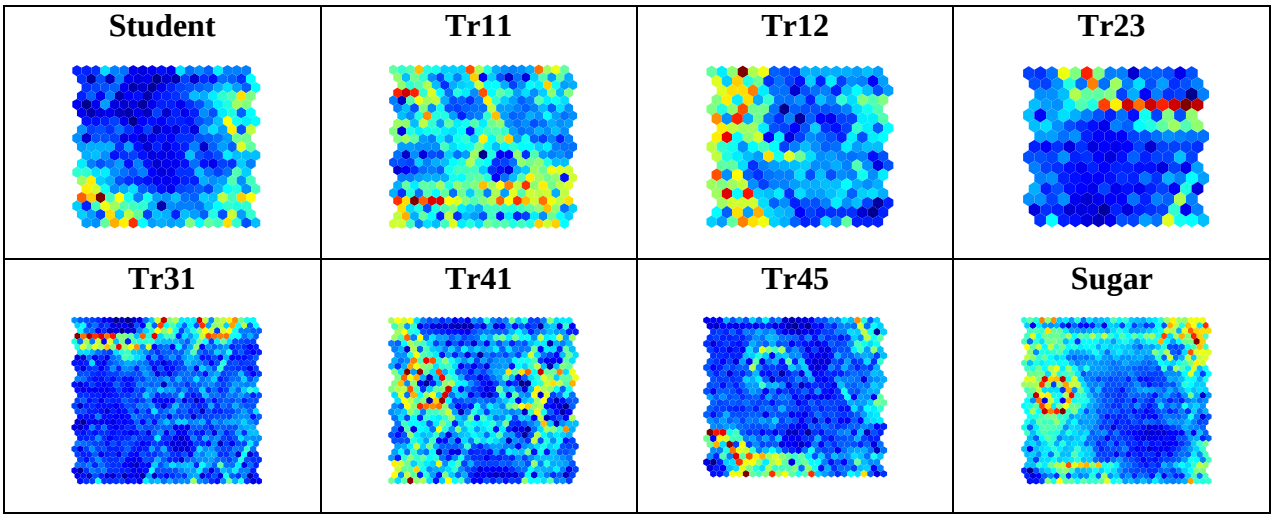


Figure 7.3 U-matrix visualizations of B-SOM.

An experiment was run to validate the proposed method.

Experiment 7.E Comparison between SOM-seeded SPK and basic SPK

Objective

To study the effect of SOM-seeded SPK

Experimental Method

Two sets of SOM were trained and used in clustering: one with a larger unit array (B-SOM) and another with a smaller unit array (S-SOM). The procedure described earlier was carried out to initialize the centroids of SPK and then a SPK clustering follows. The average of 200 runs of clustering results was reported.

Results

Figure 7.3 depicts the U-matrix visualizations of B-SOM. Table 7.5 shows average NMI scores of B-SOM and S-SOM, as well as their SOM related statistics, including SOM map size, quantization error (QE), and topographic error (TE).

Compared with U-matrices of SOM trained on low-dimensional data (such as the one described in Chapter 2 for the three flowers species), U-matrices of TREC data (ordinary documents) and requirements, particularly the latter, do not have a very clear boundary between clusters. In terms of clustering quality, SOM-seeded SPK delivers only a small improvement for data sets Tr23 and Tr31, but does not outperform basic SPK for the other six data sets.

Conclusions

The effects of SOM-seeded SPK vary for different data sets; however even in the cases that showed improvement, this improvement was minor.

Table 7.5 SOM-seeded SPK compared with randomly initialized SPK.

Data	SPK	B-SOM				S-SOM			
	NMI	*Map Size*	*NMI*	*QE*	*TE*	*Map Size*	*NMI*	*QE*	*TE*
STUDENT	0.56	10×10	0.55	0.92	0.01	8×8	0.56	0.95	0.01
TR11	0.66	11×11	0.66	0.88	0.00	5×5	0.66	0.96	0.00
TR12	0.67	9×9	0.66	0.89	0.00	5×5	0.64	0.96	0.00
TR23	**0.40**	**8×8**	**0.41**	**0.79**	**0.01**	**5×5**	**0.40**	**0.91**	**0.00**
TR31	**0.55**	**16×16**	**0.59**	**0.83**	**0.02**	**5×5**	**0.58**	**0.96**	**0.00**
TR41	0.66	15×15	0.66	0.86	0.02	6×6	0.65	0.96	0.01
TR45	0.71	14×14	0.70	0.85	0.00	6×6	0.68	0.95	0.00
SUGAR	0.54	16×16	0.53	0.93	0.02	9×9	0.54	0.96	0.01

Further analysis

What factors differentiate SOM-seeded SPK on Tr23 and Tr31 from its application in other data sets? A review of the main properties of the data sets listed in Chapter 7 highlights one potential factor, K, the number of clusters. Tr23 and Tr31 are two data sets with the smallest numbers of clusters, 6 and 7 clusters respectively. On the other hand, U-matrix plots shown in Figure 7.3 also show that Tr23 and Tr31 are characterized by more clearly depicted boundaries than the other data sets. Experiments on more data sets are needed to accurately determine which factors are positively correlated with improvement brought by SOM sampling.

Two other aspects of SOM-seeded SPK, time complexity and support for data visualization, should also be discussed. It is true that the training of SOM increases time consumption of a single clustering, but it has also been found in experiments that the number of iterations required for convergence of SPK greatly declined. Therefore, in a clustering task where multiple rounds of clustering are to be conducted the SOM-seeded SPK may eventually shorten the overall consumed time. Moreover, rather than receiving a simple list of generated clusters, analysts in certain requirement related clustering tasks may expect to navigate the clustering structure in a more visual way when using SOM [Kohonen01].

7.3.3 Pipeline hybrid model

As is widely accepted, every clustering algorithm is biased toward certain types or characteristics of data. For example, single-link merges optimistically, often resulting in poor clustering of high dimensional data, while complete-link merges conservatively and is based on the assumption of a spherical structure of the data. In contrast to these similarity-based merging methods, K-means, or SPK, iteratively moves the instances back and forth in order to optimize an objective function; but its objective functions typically assume the data are distributed regularly, for example spherically. In essence, partition-based methods such as K-means make

decisions that trade local accuracies for global optimization. In particular, when compared with other algorithms, SPK sometimes misses some obviously correct grouping decisions. To validate this hypothesis, the following experiment was designed and operated on each of the eight data sets.

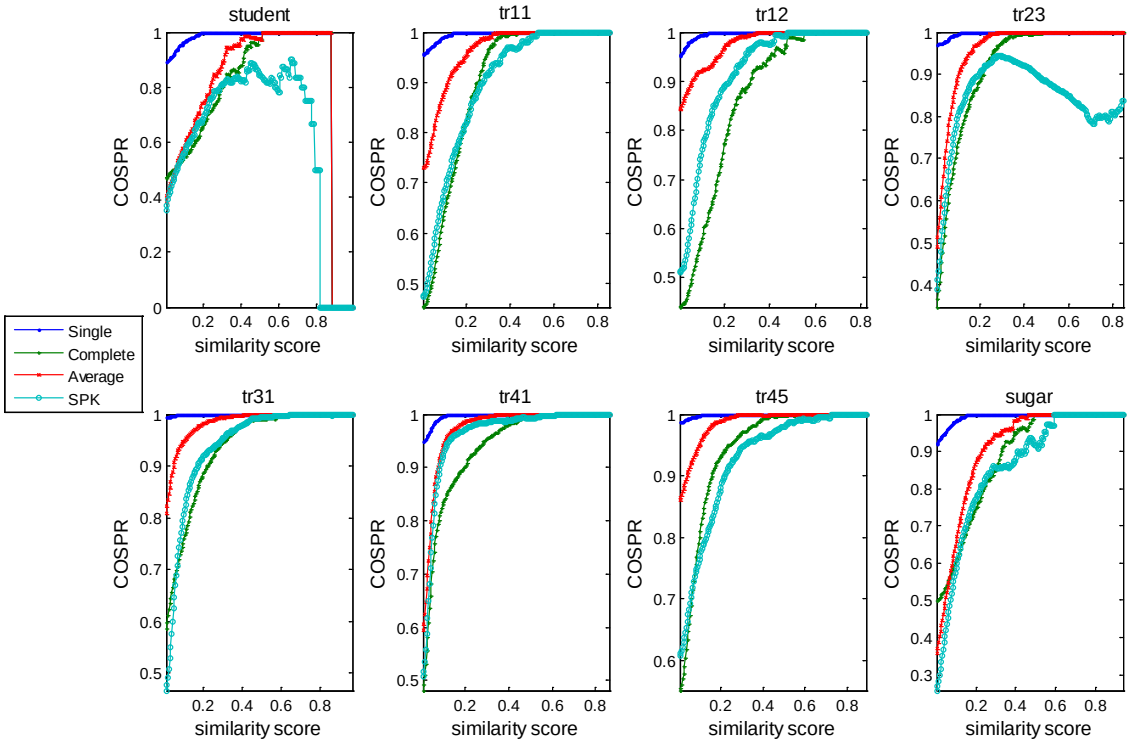


Figure 7.4 Colocation of similar pair ratios for four algorithms.

Experiment 7.F Assignments of similar artifacts in AHC algorithms and SPK

Objective

To compare the ability of different clustering algorithms to correctly assign similar artifacts into the same group

Experimental Method

(1) Four clusterings were chosen from results generated by single-link, complete-link, and average-link, and SPK
(2) Minimum and maximum scores in the similarity matrix were calculated and 50 points were evenly selected from the range.
(3) For each point with score *s*, the instance pairs with similarities equal or larger than *s* were submitted for a togetherness analysis, which first calculated the number of pairs whose

instances were together in the answer clustering and then computed the numbers of pairs with instances together not only in the answer clustering but also in the four selected clusterings, respectively. The ratios between these two sets of numbers (each corresponding to a clustering) are called colocation of similar pairs ratio (COSPR) .

(4) Curves of COSPR are depicted in Figure 7.4, where the X-axis represents the similarity point and the Y-axis represents the COSPR score.

Results

An examination of Figure 7.4 produces an interesting finding. As expected, most clusterings tend to have large scores of COSPR in pairs that have larger similarities. Among them, however, SPK exhibits the worst COSPR scores in two ways. First, it misses much more together pairs than other clusterings, especially when similarity is less than 0.5. Second, it is not only insensitive to the increase in similarity, but also progresses negatively after an early rise in smaller similarity scores on STUDENT and TR23.

The finding of experiment 7.F gives rise to the idea that, if certain clustering algorithms that are better at grouping similar instances preludes the running of SPK, the succeeding SPK may be able to produce an improved clustering since it does not need to make decisions that it does not excel in. This clustering structure, as depicted in Figure 7.5, is named the "pipeline" hybrid clustering. The rest of this section describes the experiments to validate this hybrid clustering.

Experiment 7.G Comparison between pipeline hybrid clustering with basic SPK

Objective

To study the effect of pipeline hybrid clustering

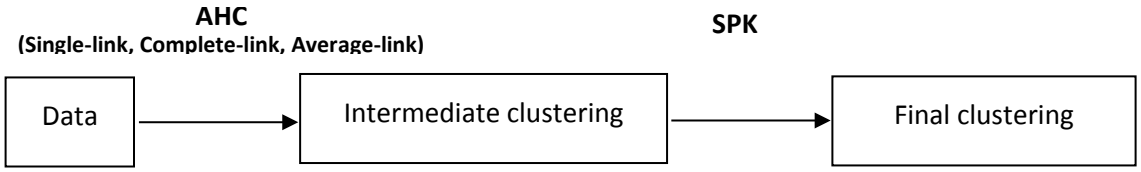


Figure 7.5 The structure of pipeline hybrid clustering.

Experimental Method

For each data set, the following steps were applied:

(1) Ten cluster numbers of intermediate clusterings were evenly chosen from the range starting with the final cluster number multiplied by eight and total number instances Steps 2, 3, and 4 are executed for each testing point pK_i.

(2) Three AHC algorithms were used to generate intermediate clusterings.

(3) The instance vectors within each cluster of the intermediate clusterings were added and resultant vectors were normalized to norm 1.
(4) SPK was run on the new instance sets (with size of pK_i) 200 times and the NMI scores from the final clusterings were averaged.

Results

The fluctuations of NMI scores with an increase in cluster number in the intermediate clusterings are shown in Figure 7.6.

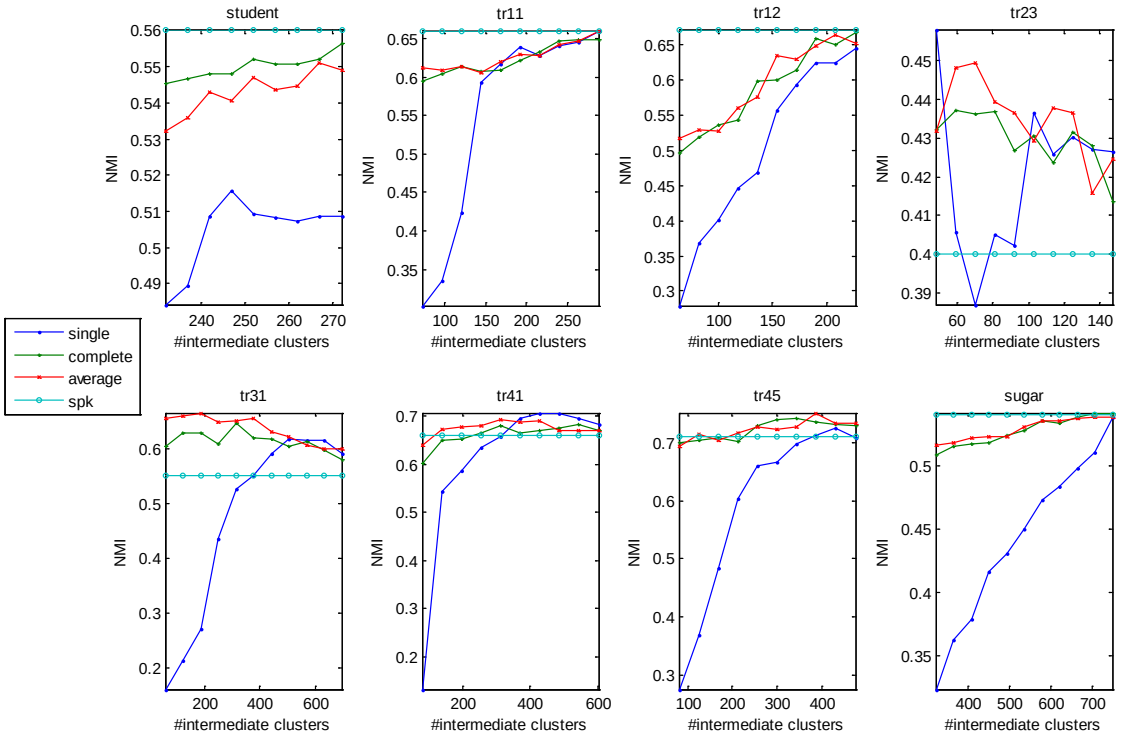


Figure 7.6 Averaged NMI scores of clusterings generated by pipeline model.

The eight data sets fell into three categories with regard to the extent of improvement over baseline SPK results:

a) Significantly and consistently improved. This case included Tr23 and Tr31, both producing better clustering almost throughout the entire testing range.
b) Improved slightly but consistently. Tr41 and Tr45 are included in this category. They outperformed the baseline result over most of the testing range but did not enhance clustering substantially.

c) Comparable to or worse than the baseline. The other four data sets, STUDENT, Tr11, Tr12, and SUGAR did not achieve the same performance as that of baseline SPK, and produced better results only toward the end of the testing range.

Despite the differences in degree of improvement, one general behavior is shared by all three cases: clustering with the pipeline model showed initial improvement, arrived at a maximum point, and then continuously declined afterward.

Another point worth noting is that the performances of the three AHC algorithms are different. Complete-link and average-link algorithms performed much better than single-link in the early phases of the testing range and they progressed more stably. On the contrary, single-link gave a very bad result at an early stage, but caught up quickly and often outperformed the other two in the final phases of the testing range.

Conclusions

The effects of pipeline hybrid clustering vary on different data sets, but majority of data sets exhibit positive improvement

Further analysis

The experimental results have shown that the pipeline model improved clustering consistently on half of the data sets. However, its effects on the two requirement data sets were not pronounced. Reviewing the COSPR results discussed in the previous section provides some insight into why the pipeline model works on some data but not on others.

First it should be noted that good clustering not only puts similar instances together, but also keeps dissimilar instances apart. Most AHC algorithms, even though achieving high COSPR, do not generate good clustering because they have a poor score for the colocation of dissimilar pairs ratio (CODPR) – the number of correctly separated pairs with regard to the total number of separate pairs. The CODPR scores shown in Figure 7.7 are generated through the same procedure that is used to generate COSPR scores, except that instead of considering assignment of similar pairs, the procedure considers assignment of dissimilar pairs.

Then juxtaposing Figure 7.4 and Figure 7.7, depicting COSPR and CODPR respectively, and examining their relationships helps explain several observations from the previous experiment:

(1) Most CODPR scores of SPK, as expected, are better than those of AHC algorithms, but not in the case of Tr23. This fact, combined with SPK's big decline of COSPR scores on Tr23, justifies why pipeline clustering obtains such large improvements for Tr23. Using the same reasoning, pipeline clustering of Tr31 gives consistent positive improvement because CODPR scores of SPK are no better than those of AHCs (note that CODPR curve of SPK actually almost fully overlaps with curve of complete-link). In contrast, CODPR scores of SPK are consistently better than CODPR scores of AHC algorithms on other data sets; this gap offsets the advantages AHC algorithms have on COSPR.

(2) CODPR scores of single-link are far below those of complete-link and average-link, which explains why pipeline clustering using single-link is inferior to the other two methods most of the time.

But COSPR and CODPR cannot entirely explain the quality difference between SPK and pipeline clustering. For example, if single-link has poor CODPR all the time, then why does it catch up with complete-link and average-link? It is believed by the author that the interaction between COSPR and CODPR is the primary factor in explaining how pipeline models can improve over SPK, but this interaction cannot be represented by a simple arithmetic combination of the two statistics.

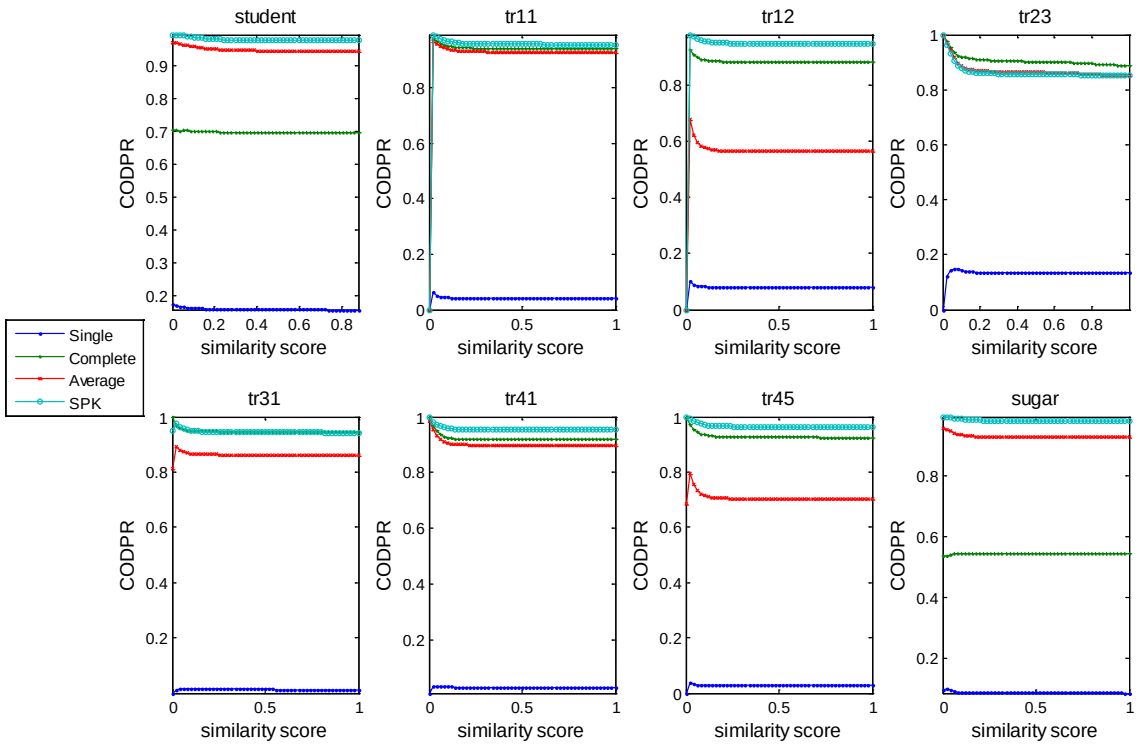


Figure 7.7 Colocation of dissimilar pairs ratios of four algorithms.

Another important question is, without answer clustering and derived COSPR and CODPR, how can one decide whether pipeline clustering should be applied and how many intermediate clusters are optimal? Table 7.6 potentially answers the first part of the question. It lists the ordered average similarity of each data set. Although a strict correlation analysis would be inaccurate due to the limited number of cases, a mere observation can lead to the hypothesis that, the more similar instances are to each other, the more substantial and more consistent improvement can be expected using pipeline clustering model. In fact this hypothesis matches

the results in Figure 7.7 well. For instance, Tr23, the data set with the greatest improvement, has far larger scores of average similarity than others. Conversely, SUGAR and STUDENT, two data sets with the least improvement, exhibit very low average similarities. Intuitively, a larger average similarity indicates stronger separation between pairs that should be together and pairs that should be apart, thereby increasing the precision of merging committed by AHC algorithms, and then boosting the overall clustering.

Table 7.6 Average similarities of 8 data sets.

Data	*Average similarity*
tr23	0.1290
tr11	0.0878
tr12	0.0850
tr45	0.0810
tr31	0.0763
tr41	0.0680
SUGAR	0.0468
STUDENT	0.0411

The answer to the latter part of the question posed in the beginning of the last paragraph is rather difficult. Basically two factors impact the choice: data characteristics and algorithm choice. For a data set that is known to respond well to the pipeline clustering model, such as Tr23 and Tr31, the cluster number of the intermediate clustering should be set smaller when using average-link and complete-link; but when using single-link, the cluster number should not be too big nor too small (this optimal cluster number for single-link is approximately half of the total instance number for Tr23 and Tr31). On the other hand, for a data set for which one is not sure about its response to pipeline model, setting larger intermediate cluster numbers is suggested.

As a final comment to pipeline clustering, since there are actually two clustering phases, and AHC usually is more expensive than SPK, the overall time complexity would be the same as AHC, which is $O(N^2)$.

7.4 Consensus clustering

The topic of this section is consensus clustering, another form of hybrid clustering model. Typically hybrid clustering models can have two modes: pipeline and consensus. In the pipeline hybrid clustering, such as the ones that have been discussed in section 7.3.3, the output of one clustering method is passed as input to another clustering method, attempting to constrain the search space for the latter and thereby avoiding certain bad choices. In the consensus hybrid clustering, on the other hand, a carefully selected set of clusterings, called a clustering ensemble, is generated using diverse clustering methods and integrated to deliver an improved partitioning.

Thus, in terms of structure, pipeline hybrid clustering is serial, whereas consensus clustering is parallel.

While pipeline hybrid clustering aims explicitly at avoiding the limitations of certain algorithms, consensus clustering attempts to achieve consistency within the ensemble. In other words, it produces the solution by voting among the members of the ensemble. Based on the assumption that the average decisions made in the ensemble are usually accurate, it filters out bad partitions and is potentially able to construct higher quality clusterings.

This section is organized as follows: section 7.4.1 describes the methods of ensemble generation and integration adopted in the research, section 7.4.2 discusses the problem of choosing good ensemble, and section 7.4.3 compares the performance of consensus clustering with SPK, and analyzes the results.

7.4.1 Ensemble generation and integration

The research uses sub-sampling and average-link AHC as methods for ensemble generation and integration respectively. Sub-sampling of a clustering ensemble of size R is through subset clustering and classifying. In each run of sub-sampling, a proportion α of the whole dataset is randomly chosen and then partitioned into K clusters using SPK, followed by the classification of each remaining need into its most closely related cluster.

The association-matrix M is generated as described in section 5.5 and then average link is used to produce the final clustering. The following two experiments are designed to study two questions: what values of α and R are optimal in producing better clustering? And what is the optimal integration method? The following experiment is setup to empirically study the effect of different scores of α and R on clustering quality.

Experiment 7.H Study of effects of using different values of α and R

Objective

To determine the optimal setup of sampling ratio and ensemble size

Experimental Method

(1) α was set to 0.5, 0.6, 0.7, 0.8, and 0.9; considering the sizes of data sets (less than or equal to 1000), setting a proportion below 0.5 was shown through a series of initial experiments to generate a distorted subset clustering.
(2) R was set over the range between 100 to 1000 with increment of 100.
(3) For a certain combination of α and R, single-link, complete link, average-link, and centroid-link were used and NMI scores were calculated.

The results are shown in Figure 7.8 and table 7.7.

Results

Figure 7.8 only contains the NMI scores for average-link. It can be observed that:

(1) Generally very low scores of α, such as 0.5 or 0.6, produced worse results than higher scores of α; however, after 0.6, the differences between different α scores were not substantial, which is especially obvious in STUDENT, Tr23, and SUGAR. An associated observation is that the proportion of 0.9 outperforms other proportions most of the time.

(2) Clustering results are less sensitive to the changes of size of ensemble R than to α. For example in STUDENT and Tr23, NMI scores virtually remained flat with R scores between 100 and 1000.

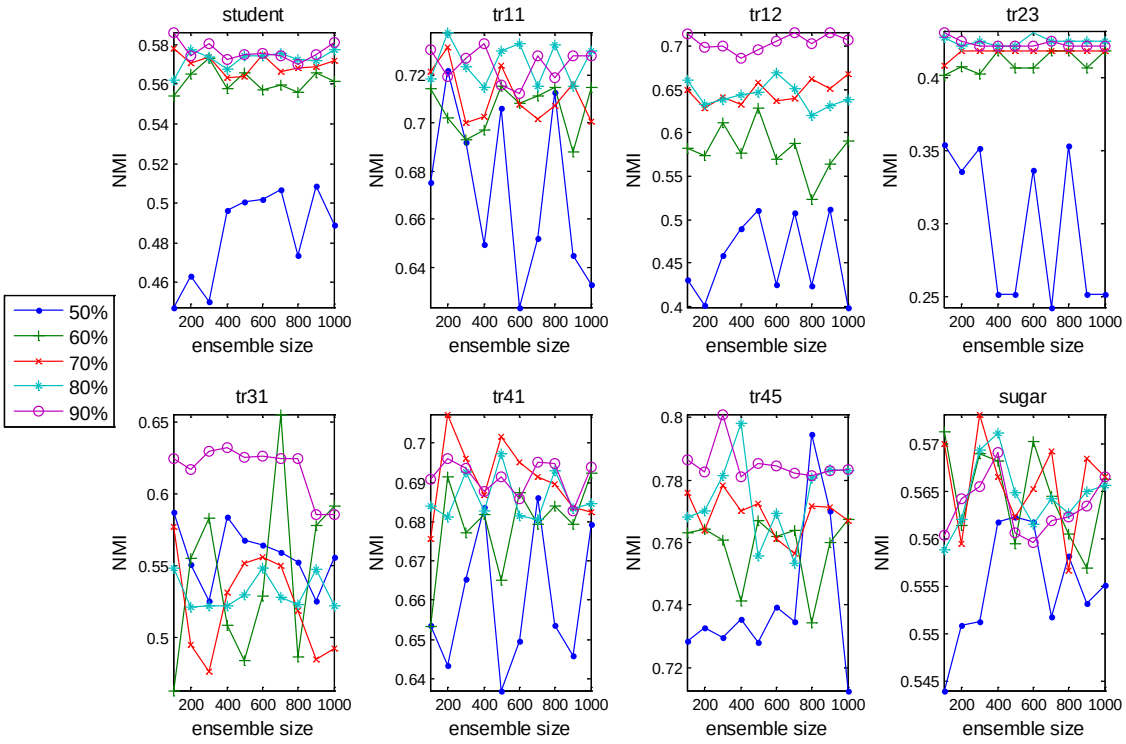


Figure 7.8 NMI scores of consensus clustering with different scores of sampling proportion and ensemble size.

The remainder of this section adopts α of 0.9 and R of 500 in other experiments.

Also, the NMI scores listed in Table 7.7 justify the use of average-link to integrate clustering ensemble s.

Conclusions

Empirically, for data sets of sizes less than 1000, setting the sampling ratio to 0.9 and ensemble size to a number between 200 and 500 can give satisfying results.

An experiment was conducted to evaluate the improvements obtainable by using consensus clustering on the six TREC datasets and two requirement data sets STUDENT and SUGAR.

Table 7.7 Performance of clustering using four AHC algorithms.

Data	*Single*	*complete*	*average*	*centroid*
STUDENT	0.4071	0.5588	**0.5675**	0.2529
tr11	0.7516	0.7068	**0.7301**	0.7234
tr12	0.3568	0.6594	**0.6658**	0.6265
tr23	0.5009	0.4190	**0.4221**	0.4221
tr31	0.1163	0.5298	**0.5246**	0.4678
tr41	0.5608	0.7071	**0.6802**	0.6581
tr45	0.5694	0.7442	**0.7663**	0.7449
SUGAR	0.2703	0.5633	**0.5671**	0.0752

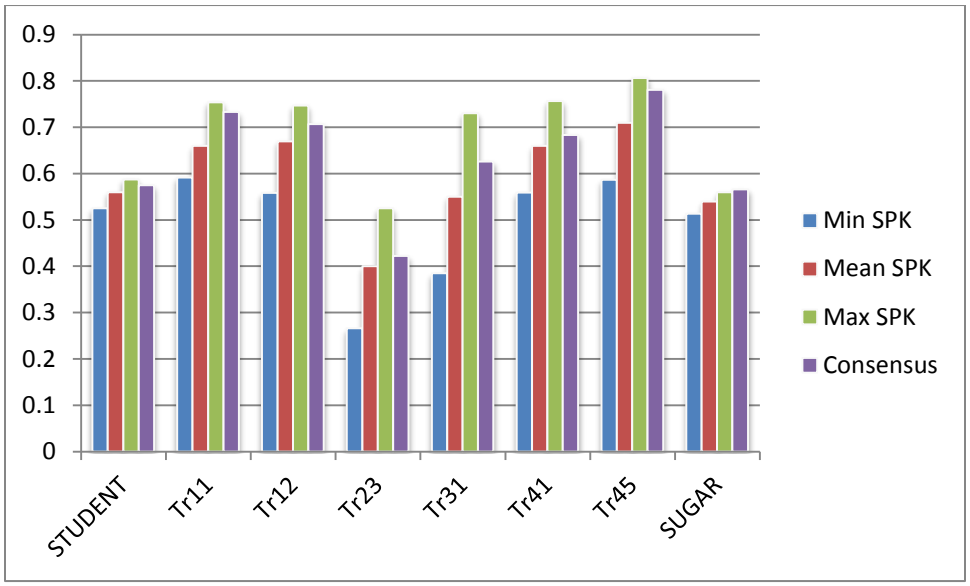


Figure 7.9 Results of consensus clustering versus spherical K-means.

Experiment 7.I Comparison between consensus clustering with basic SPK

Objective

To study the effect of consensus clustering

Experimental Method

The experiment compared the NMI scores of each of these datasets when clustered using basic 2-stage SPK versus the consensus algorithm just described, which used average linkage hierarchical clustering to partition the co-association matrix. Because SPK generates different results depending upon initial seedings, the algorithm was run 200 times for each dataset, and the minimum, maximum, and mean scores are reported. These results are depicted in Figure 7.9.

Results

Figure 7.9 shows that consensus clustering results were always above the mean obtained using SPK, but usually just below the maximum. An interesting exception to this is that for the SUGAR data, representing a very realistic medium sized project, the consensus clustering scores were higher than the maximum obtained using SPK. In general, these results are highly significant because they demonstrate that consensus clustering is more consistent and robust than basic SPK.

A t-test was also conducted to measure the significance of consensus clustering's improvement over basic SPK results. For every data set, 200 ensembles, each containing 200 clusterings generated through a 0.9 ratio sub-sampling, produced 200 clusterings using the average-link algorithm. A one-tailed t-test was then applied to evaluate if these clusterings were statistically better than the 200 clusterings generated using the basic SPK. The p-scores for eight data sets, by the order listed in Table 7.7, are 3.9718E-039, 7.6654E-064, 0, 2.1869E-006, 0, 0, 2.3601E-030, and 7.9120E-056. Apparently, under usual significance levels of 95% and 99%, those p-scores support the alternative hypothesis that consensus results are better than SPK.

Further analysis

This small section analyzes why consensus clustering improves the clustering significantly and consistently from the perspective of similarities. These similarity scores in the co-association matrix M define relationships between instances more precisely than the similarity scores in the original matrix O. First a few words about notations. For the remainder of this discussion, pair-wise values in M are referred to as voting scores (VS), while the values in O are referred to as similarity scores (SS) represented by the cosine direction values between pairs of instance vectors. Also, if a pair of instances is assigned together in the answer clustering, it is called a ML, otherwise it is called a CL.

The distributions of voting scores and similarity scores for all eight data sets were analyzed in respect to knowledge of MLs and CLs from the published answer clusterings. In Figure 7.10, each of the data sets is represented by a single column. The graphs in the first row show the voting score distributions of ML in M, while the second row shows the voting score of CL in M. The third and fourth rows show similar values for the cosine values in O. NMI scores, depicting the similarity between the consensus clustering and the published reference clustering, are shown on the top row in parenthesis next to each dataset name.

These results clearly indicate that all of the datasets had similar distributions of ML and CLs for the original proximity matrices, with very high concentrations over the range near 0. In contrast the scores in the co-association matrix provided a clearer differentiation between MLs and CLs by boosting ML scores. This implies that the pair-wise voting scores compiled from the consensus algorithm tend to approach their "true" similarity scores. It also explains why the proximity-based algorithms such as agglomerative hierarchical clustering yield poor results on the similarities directly from O but perform much better on M.

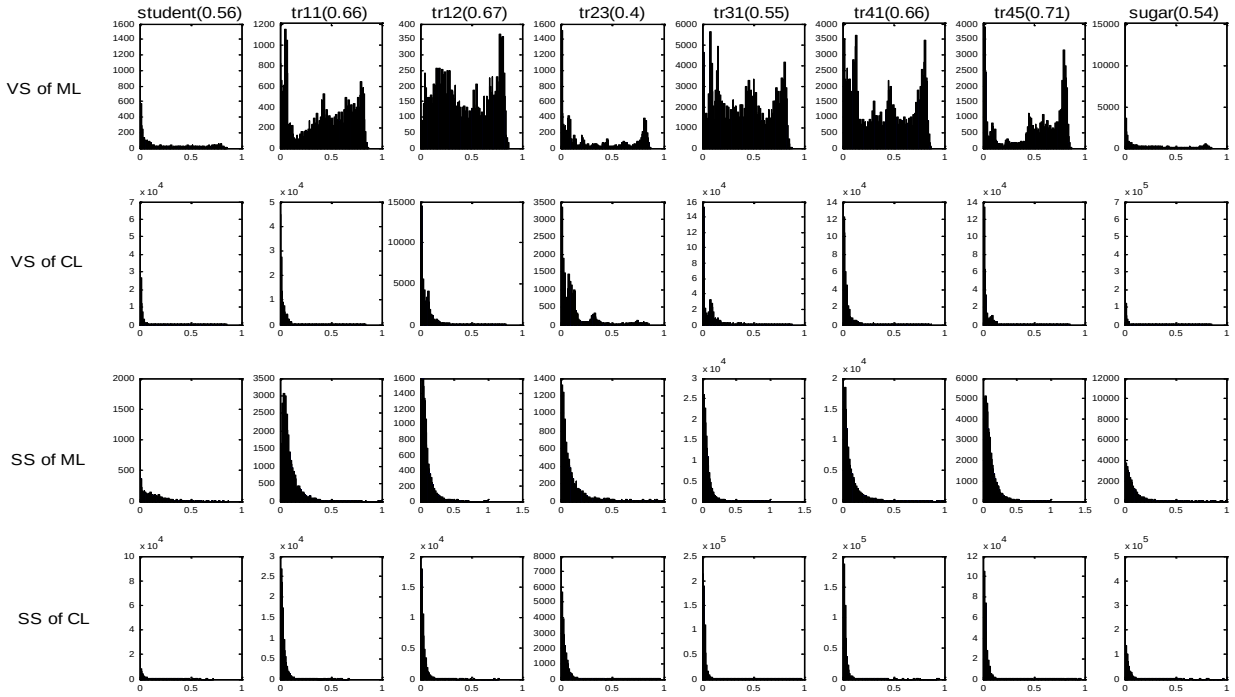


Figure 7.10 Score differentiations between ML and CL instances in the co-association versus original matrices.

It can be observed that among TREC data sets Tr11 and Tr45 scored the highest NMI values while Tr23 scored lowest, and it is interesting to note that these differences correspond to the different shapes of ML/CL distribution shown in Figure 7.10, where the easy-to-cluster Tr11 and Tr45 have high concentrations approaching one (i.e. towards the right of the graph) for ML and more concentrations towards 0 for CL, while the harder-to-cluster Tr23 did not exhibit such strong differentiation, meaning that even in the co-association matrix there was still significant disagreement about the MLs.

7.5 Summarization and more analysis

This chapter described a considerable number of methods, experiments, results (tables and figures), analyses, and hypotheses. The main findings included:

(1) There is no silver bullet of cluster validation metric that does not involve comparison with an answer clustering.
(2) Two-stage SPK is far better than batch-mode SPK in clustering high-dimensional documents and software requirements. Most of the time it also outperforms other algorithms, such as the agglomerative hierarchical and bisecting algorithms.
(3) Various approaches improve over SPK to different extents on different data sets. The studied methods included:
    a. Matrix decomposition via PCA and SVD.
    b. SOM-seeded SPK.
    c. Pipeline hybrid clustering.
    d. Consensus clustering.

The last finding requires further analysis. Given the results from these approaches, when and how can an approach be used for a certain data set? Similar questions have been discussed in various places within the previous discussion, but an attempt must be made to uncover some higher level insight with better generality.

Two main issues are involved in determining when a particular approach should be used. The first is effectiveness – measured by the quality of the generated clustering. This is usually the primary concern of most clustering tasks. The second is efficiency – the time cost of the approach, an issue that is more relevant to the clustering of large data sets.

Effectiveness

Effectiveness will be discussed first. Table 7.8 displays again the main properties of eight data sets but also lists: i) the average similarity between instance pairs calculated using Cosine direction (denoted as Avg_S), and ii) the responses to various enhancement approaches (denoted as SVD, SOM, Pipeline, and Consensus respectively), where Δ represents comparatively minor improvement and ✓ indicates substantial improvement.

By observing the columns, one can find consensus clustering obtained improvements on all data sets and three of the improvements are significant; both pipeline model and SVD showed improvements in four datasets, but while the pipeline delivered two significant improvements; SOM brought the least improvements. These observations show that from a practical perspective, consensus clustering offers an effective and robust way to improve cluster quality.

By examining the rows, one finds that Tr23 and Tr31 stand out by responding positively to each enhancement approach, especially to pipeline and consensus. Other data sets, according to their responses, can be ranked as: Tr45, Tr11, Tr41, Tr12, SUGAR, and STUDENT. To determine if there is any property of the data that determines how effectively certain enhancement approaches work, a correlation analysis was conducted between data properties and their ranks of improvement. These results are shown in Table 7.9.

Although ranking here is a very rough quantification of enhancement, this impreciseness does not void the comparative meaning of the observation. Enhancement rank has a rather strong positive or negative correlation with the average length of the document |d|, the number of clusters K, or the average similarity Avg_S. Also note that since there's a very strong positive correlation between |d| and Avg_S, the finding here is consistent with the discussion in the section of pipeline model where it was hypothesized that larger average similarity leads to more significant improvement. Another noticeable correlation is between K and |d|, which is less relevant as K is usually subjectively determined and therefore does not belong to basic characteristics of the data.

Table 7.8 Main properties and responses to various enhancement approaches of data sets.

Data	n_d	|d|	K	Avg_S	SVD	SOM	Pipeline	Consensus
STUDENT	366	8	29	0.0411				Δ
tr11	414	281	9	0.0878	Δ			✓
tr12	313	273	8	0.0850				Δ
tr23	204	385	6	0.1290	Δ	Δ	✓	Δ
tr31	927	268	7	0.0763	Δ	Δ	✓	✓
tr41	878	195	10	0.0680			Δ	Δ
tr45	690	280	10	0.0810	Δ		Δ	✓
SUGAR	1000	27	40	0.0468				Δ

Table 7.9 Correlations between data properties and their responses to enhancement.

	n_d	|d|	K	Avg_S	Rank
n_d	1	-0.3653	0.3271	-0.5210	-0.0574
|d|		1	-0.9120	0.9347	**-0.8202**
K			1	-0.7602	**0.74180**
Avg_S				1	**-0.7129**
Rank					1

All previous discussions lead to a very plausible conclusion that: (1) Generally, the evaluated enhancement approaches tend to work better on datasets with longer documents; and (2) Consensus clustering can be effectively applied to a broad spectrum of data sets. Unfortunately, software requirement data sets, such as STUDENT and SUGAR, are highly terse and in the conducted experiments were demonstrated to be insensitive to enhancement approaches. This suggests that un-supervised clustering may be limited in their effectiveness for requirements clusterings. The next chapter therefore investigates the user of supervised clustering techniques that incorporate prior knowledge in order to improve clustering results.

Efficiency

Besides clustering quality, a few others factors deserve consideration in certain clustering tasks, including time efficiency, space efficiency, and usability.

SVD/PCA and SOM require a lot of computation in producing the matrix decomposition or in training the SOM array. After that the decomposition and SOM can be used again and again, but the clustering will be much slower, as the originally sparse matrix has become highly dense. Pipeline models primarily require calculating a proximity matrix, which is of $O(N^2)$ time complexity. Consensus clustering has R clustering in ensemble, so its time complexity is proportional to the size of the ensemble.

SVD/PCA and SOM also have high space complexity because their main structures, such as matrix factors and SOM arrays, are not sparse. Pipeline and consensus clustering add little extra space overhead.

Using consensus clustering is relatively simple: there are only two main parameters to adjust and both have rather stable suggested range. The other three methods require finer tuning and therefore more heuristics must be provided to achieve satisfactory results.

These factors support our decision to utilize consensus clustering.

Hypotheses on the impact of data sets' characteristics to clustering

As suggested by the proceeding discussions, the clustering of requirements is more difficult than the clustering of ordinary documents. A further investigation was conducted to investigate whether several characteristics of the data sets could contribute to the difficulty of clustering. A number of hypotheses were then proposed based on the findings.

Distributions of document lengths and word occurrences

The plots with terms, including distributions of term occurrences (Figure 7.11) and plots showing Zipf's law (Figure 7.12), which states that the frequency of a term is inversely proportional to its rank in the frequency [Zipf49], did not differentiate the requirements data sets from the six TREC data sets. On the other hand, distributions of document length, calculated as the summed numbers of distinct terms within a document, and distributions of term occurrence, computed as the distinct times a term appears in a corpus, draw a clear line between requirements data sets and TREC data sets. As can be easily observed from Figure 7.13, documents in the six TREC data sets are much longer than requirements in STUDENT and SUGAR, and contain more repetitious terms. From these results it can be hypothesized that **data sets with larger document sizes are relatively easy to cluster.** The following sections attempt to address the question of why this is the case.

Distributions of term repetition inside documents

A high level of repetition of non-trivial terms inside a document potentially clarifies and strengthens the central theme of the document, thus making clustering easier. The plots in Figure 7.14 and 7.15 display the distributions of term repetitions represented in two ways: the number of terms occurring more than once per document and the average occurrence of terms per document. In both types of distributions, the levels of term repetition in STUDENT and SUGAR are significantly less than those in the TREC sets. This observation gives rise to a hypothesis that **the documents exhibiting more term repetition tend to be easier to cluster**.

Coupling between cluster topics

The difficulty of partitioning a set of documents can also be measured by examining the uniqueness of cluster topics. If each cluster represents a distinct topic, its coupling with other clusters should be relatively low, and a clustering algorithm can be expected to produce relatively accurate results. On the contrary, if the majority of clusters have a high proportion of overlapping topics, then clustering algorithms will be more likely to make incorrect clustering decisions, leading to poor results. To measure the coupling between cluster topics, the topic of each cluster is identified as the set of T most frequently occurring terms within the cluster. The set of overlapping terms is then identified between each pair of clusters, and then the total number of overlapping terms is computed for each cluster, normalized by K-1, to produce a coupling score. For the answer clusterings, with T set to 10, the average topic coupling scores of six TREC datasets are 0.1975, 0.1250, 0.8889, 0.2449, 0.1400, and 0.1800, while the average topic coupling scores of STUDENT and SUGAR are 0.4828 and 0.2737 respectively. The average coupling scores for different values of T are shown in Figure 7.16 and give consistent rankings for various T values across the eight data sets. The generally higher coupling scores of STUDENT and SUGAR suggest that the topics within a requirements data set are less distinct than those of the TREC datasets. Furthermore, the high topic coupling of requirements data can be observed not only within the human created answer clusterings, but also in dynamically generated clusterings. This implies that, high coupling serves as an indicator that a data set might be difficult to cluster. The measure of topic coupling, however, is strongly related to the determination of the number of clusters, and how this dependency impacts the judgments made based on topic coupling requires further investigation.

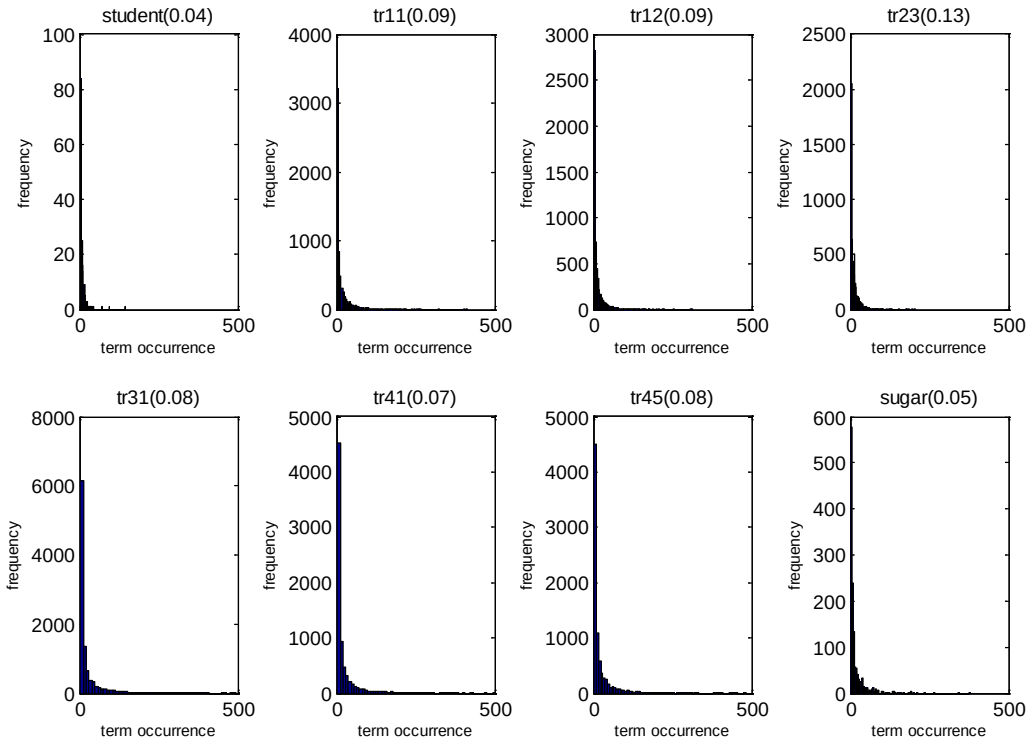


Figure 7.11 Distributions of term occurrences

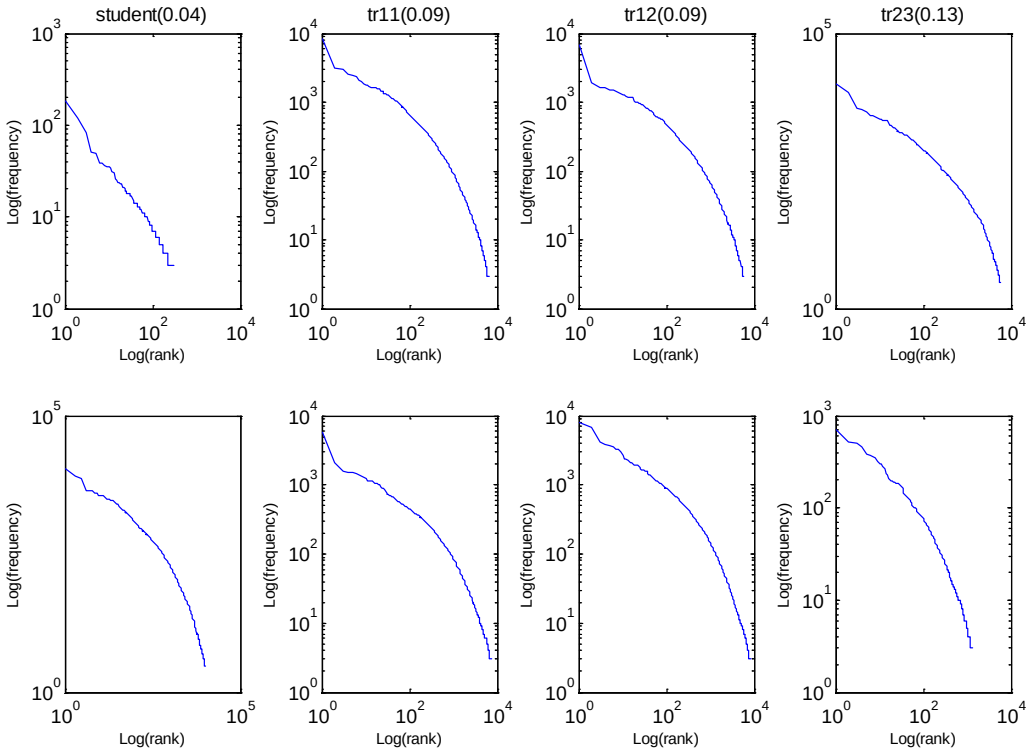


Figure 7.12 Log of Terms' frequencies versus Log of terms' ranks (Zipf's law)

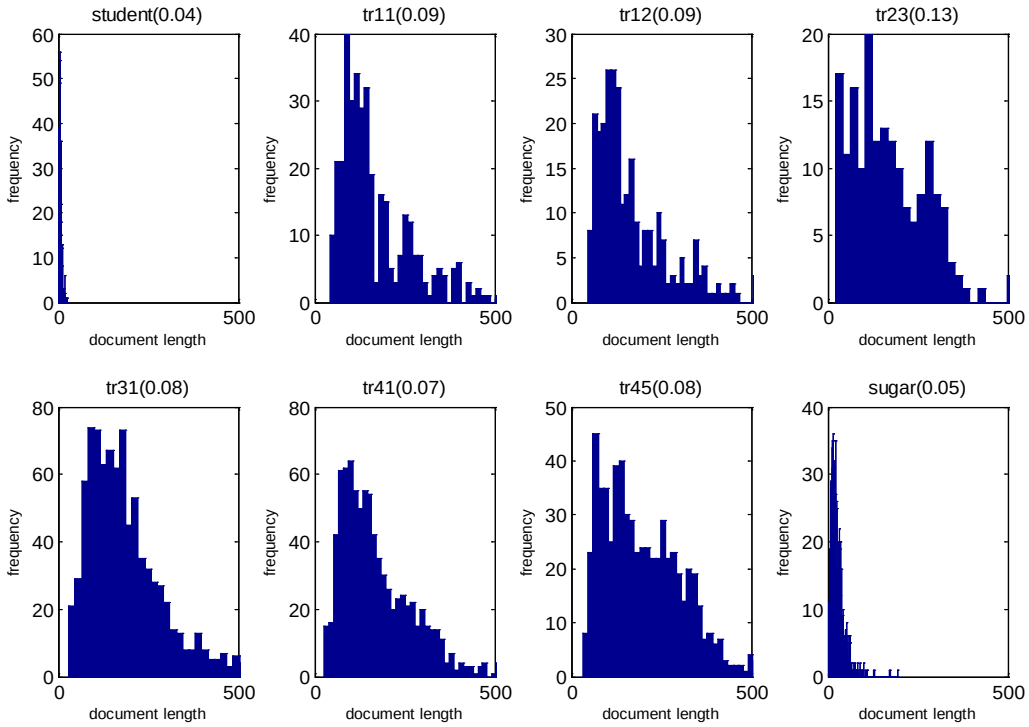


Figure 7.13 Distributions of document lengths

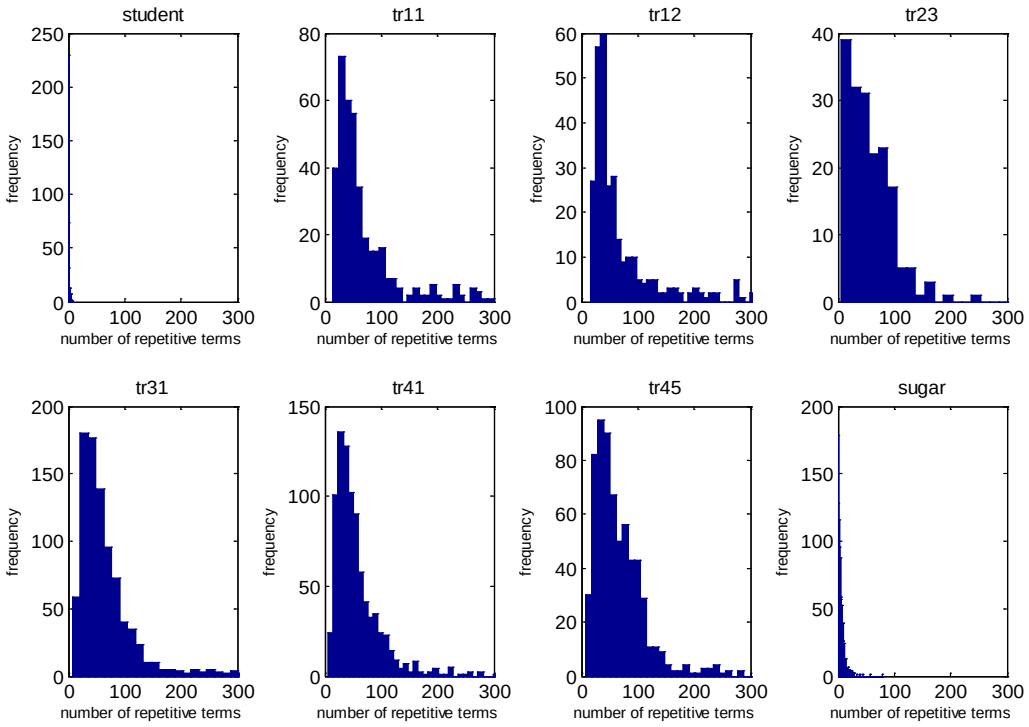


Figure 7.14 Distributions of number of terms with more than one occurrence

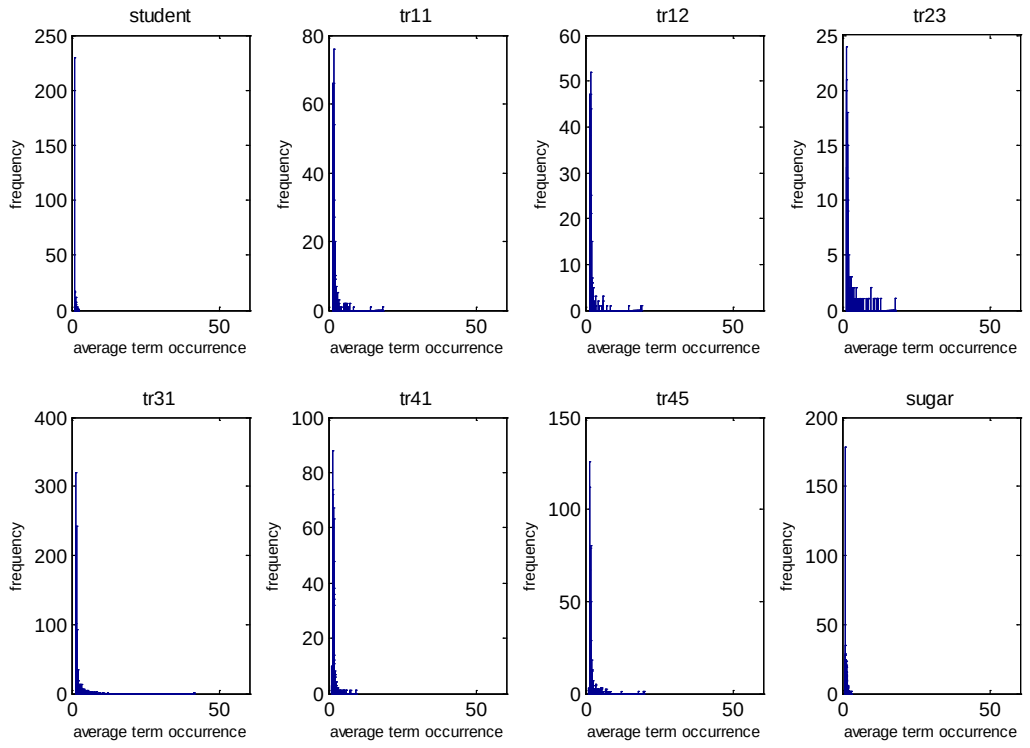


Figure 7.15 Distributions of average occurrence of terms

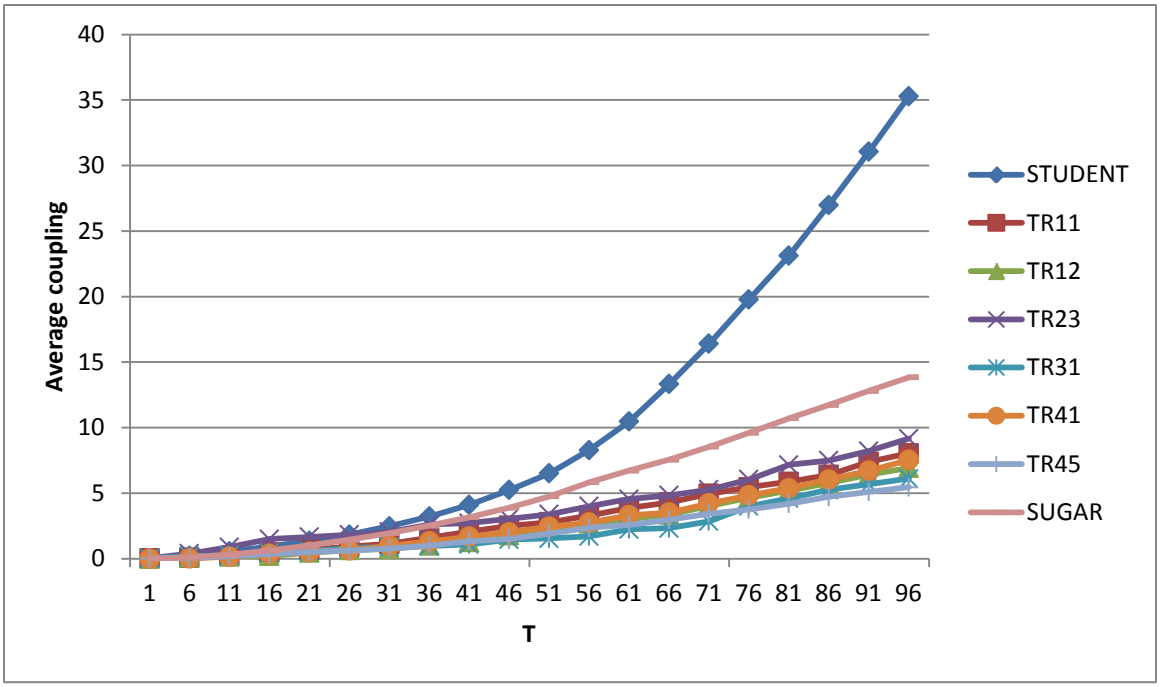


Figure 7.16 Average topic coupling scores against increasing number of topic terms T

CHAPTER 8. CONSTRAINED REQUIREMENTS CLUSTERING

8.1 Introduction

As the previous chapters have suggested, the clustering of software requirements is highly difficult. AHC, SPK, and various enhancement approaches, including consensus clustering, did not significantly improve results due to the terseness and background noise underlying high-dimensional requirements. In fact a number of researchers have investigated the use of semi-supervised clustering, also named constrained clustering, to improve clustering quality. In constrained clustering, the clustering process is guided by prior knowledge or constraints collected through user feedback, and prior research results have demonstrated the effectiveness of constrained clustering applied across many types of data sets.

This chapter studies constrained clustering in the domain of requirements engineering. It not only validates existing constrained algorithms for clustering requirements, but also proposes a new framework that is experimentally demonstrated to be able to generate more informative constraints and produce higher quality requirements clusters. The remainder of this chapter is laid out as follows: section 8.2 describes the forms of constraints to be studied, and then briefly surveyed the research in constrained clustering. Section 8.3 outlines the new constrained requirement clustering framework and section 8.4 and 8.5 discuss two main phases of the framework in detail. Section 8.6 describes the setup and results of experiments designed to evaluate this framework on eight data sets, and finally section 8.7 concludes the whole chapter.

8.2 Background of constrained clustering

Clustering constraints can be generally classified as cluster level or instance level, where cluster level constraints dictate global rules such as prohibiting empty clusters, and instance level constraints specify something about the relationships between pairs of elements. The most commonly used instance level constraints are pair-wise Must-Link (ML) and Cannot-Link (CL) constraints, indicating respectively whether a pair of instances must be placed in the same or in separate clusters. Due to inconsistencies when constraints are gathered from real users, e.g., $(x_i, x_j) \in \mathrm{ML}$ and $(x_i, x_k) \in \mathrm{ML}$, but can't-link $(x_j, x_k) \in \mathrm{CL}$, the constraints cannot be treated as hard and fast rules. The use of constrained clustering is particularly pertinent in the requirements domain because the high level of interaction with stakeholders makes feedback gathering quite attractive. This chapter focuses on ML/CL constraints as opposed to other types of instance constraints primarily because this simple form of constraint is the easiest one to design and collect from users.

Constrained clustering has been investigated across a wide variety of algorithms, including hierarchical clustering [Davidson05], non-negative matrix factorization [Li07], and partitioned clustering, especially the variants based on K-means which has been shown to be the very efficient. Constrained K-means variant algorithms can be further categorized as constraint

enforcement, learning distance metric, seeding, violation penalty, and hybrid approaches. One representative technique for the constraint enforcement algorithm is COPK-means [Wagstaff01], which strictly enforces both ML and CL constraints during the cluster assignment stage. The algorithm proposed by Xing [Xing03] tries to learn a diagonal or full covariance matrix from the constraints, and then calculate the Mahalabios distance for points to reflect the impact from the constraints. Seeded-KMeans proposed in [Basu02] utilizes labeling information that is more specific than standard ML/CL, of partial points to initialize the centroids and constrain the cluster assignment. PCK-means, which is a violation penalty algorithm [Basu04a], modifies the objective function in K-means by adding a penalty for constraint violations in the form of a weighted number of violations. MPCK-means [Bilenko06] is a metric learning-enhanced violation penalty algorithm, which combines the ideas from [Basu04b] and [Xing03]. Additionally, model-based and probabilistic partitioning algorithms incorporating pair-wise constraints have been studied in extensively [Basu04b, Cohn03]. Tang et al. [Tang07] proposed a hybrid method named SCREEN which, like our framework, was designed specifically for constrained document clustering. SCREEN projects the instance vectors using an orthonormal matrix derived from constraints so that in the new feature space the similarity between instances involved in a ML constraint are maximized and the similarity between instances in a CL constraint are minimized. However, our analysis of this approach indicates that the experiments Tang et al. reported against six TREC datasets used a far from optimal baseline against which to compare their results. This oversight meant that much of the gain in quality achieved through their method could have been more quickly and efficiently achieved through a simple optimization of the underlying SPK algorithm.

8.3 A new consensus-based constrained clustering framework

As large scale constraint collection is human intensive and therefore extremely expensive in practice, it is important to maximize the potential usefulness of the constraint set, namely, the degree to which the set of constraints can improve the clustering quality. One basic criteria is the feasibility of finding a partitioning that can satisfy all the ML and CL constraints, as this a crucial problem. Davidson et al. [Davidson06a, Davidson07] proved that the general feasibility determination is NP-complete but by relaxing feasibility as a q-inductive graph coloring problem, constrained clustering can be improved by generating a constraint set that can be satisfied by a permutation of the instance order. In a related paper [Davidson06b], they proposed the two utility metrics of informativeness, measuring the amount of information in a constraint set that the clustering algorithm could not have determined on its own, and coherence, representing the amount of agreement within the constraints themselves, with respect to a given distance metric.

One of the weaknesses of most well-known approaches for constrained clustering is that the pairs of instances for which ML and CL constraints will be generated tend to be randomly selected from an answer set. Typically, for experimental purposes, a pair of documents is randomly selected and a ML or CL constraint is generated based on the true assignment of the

documents within the answer set. However, as our results will show, the random approach does not perform well on requirements datasets which are typically characterized by large numbers of finely grained clusters, and composed of documents that tend to be very short and ambiguous.

A few researchers have also investigated active learning techniques for constraint generation [Basu04a, Greene07], as this approach can improve informativeness of future constraints based on feedback gathered incrementally from the user. In actual practice, constraint generation techniques based on active learning assume the existence of an oracle that can respond to an incremental query, and this may not always be feasible in practice.

To address the problem of generating more informative constraints and improving the quality of constrained clustering especially for software requirements, we proposed a unified consensus based constrained clustering framework. This framework, which is depicted in Figure 8.1, is comprised of the two main phases of constraint generation and constrained clustering. In phase 1, an initial clustering ensemble and a related co-association matrix M are generated, and used to identify a set of constraints. In phase 2, these constraints are used to generate a second improved clustering ensemble against which a third clustering is performed to create the final output P*. In related work, Yan et al. [Yan06] used a consensus-based approach to cluster genes. They divided the feature space into random subsets and then applied a distance-metric learning-based constrained clustering to each feature subset, producing multiple subspace clusterings. Our approach adopts some of the same techniques as Yan et al, but is customized to our domain through the selection of algorithms for building the ensemble, selecting constraints, and generating the final partitioning. These phases are described in more detail below.

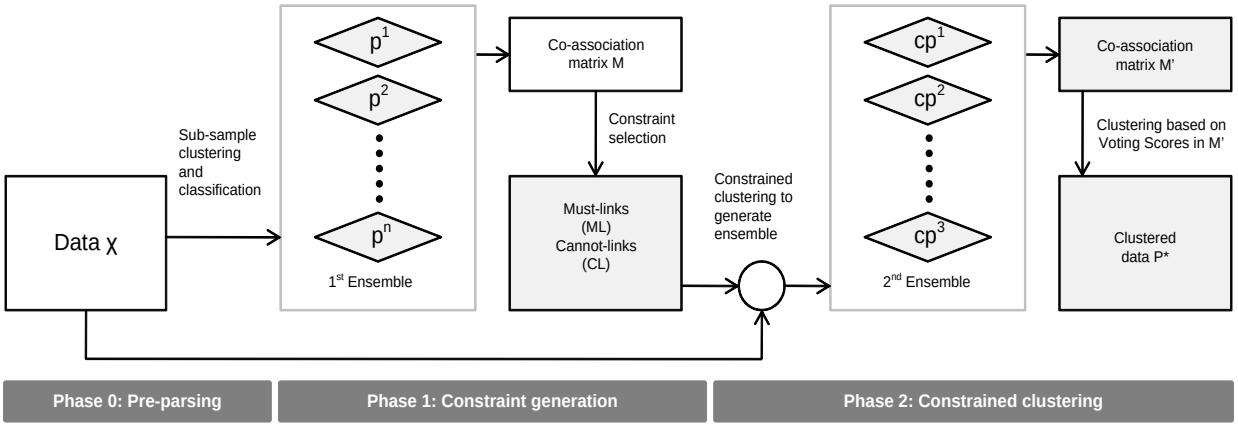


Figure 8.1 The framework of consensus based semi-supervised clustering.

8.4 Bounded constraint Selection

The 1st ensemble in our framework is created through randomly selecting and clustering subsets of the documents from the full feature space, and then generating complete partitions through classification. A co-association matrix M is then produced by summing the frequency with which each pair of instances occurs together across all of the clusterings in the ensemble.

To generate the 2nd ensemble, our framework does not choose constraints randomly, but pair-wise constraints ML/CL are selected only from the subset of instance pairs in M that exhibit voting scores within a given interval window [a, b]. The illustration of this bounded constraint selection scheme consists of two parts, one intuitive and another formal.

An intuitive explanation of the underlying idea is provided first. As previously stated, it is important to select constraints that maximize the benefits of constrained clustering. Intuitively speaking, it is likely that the greatest benefits will be obtained by identifying borderline relationships. For example, pairs of instances exhibiting either very high or very low proximity scores already tend to get correctly placed together or separately by the unsupervised clustering algorithm, and so asking a user to categorize them as ML or CL provides less useful information than asking for feedback on pairs of instances with more intermediate proximity values. For a graphical illustration, Figure 8.2 shows three clusters with 4 constraints: A and B are CL and C and D are ML. Apparently choosing A over B is more advantageous as relationship B can be easily identified by any un-supervised clustering algorithm while A is much harder to find, therefore selecting A as a CL can delimitate the boundary between clusters much better. For the same reason, D is more informative than C.

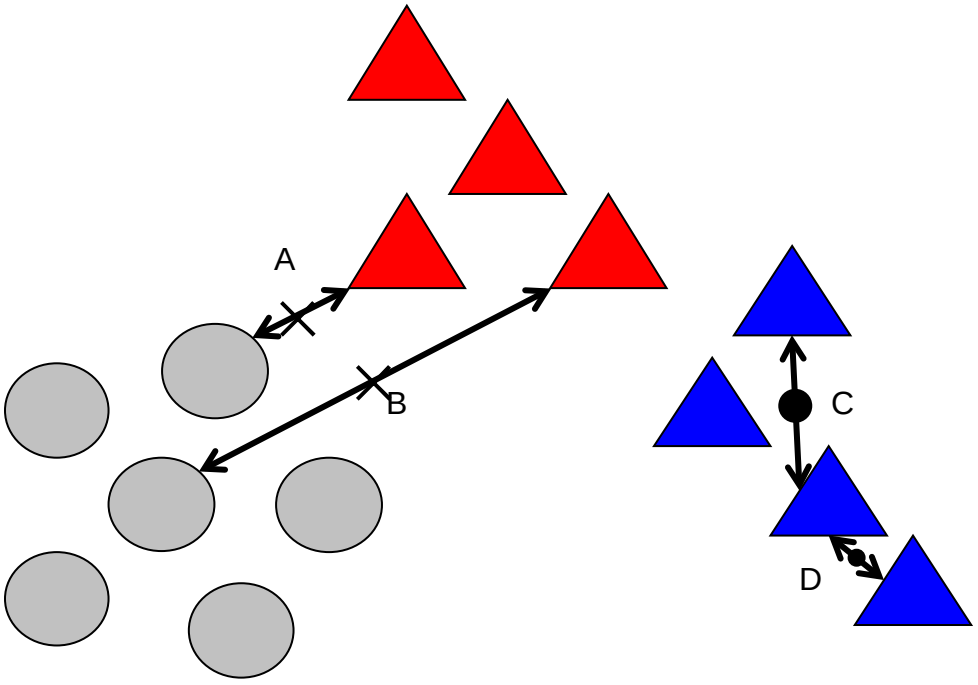


Figure 8.2 The framework of consensus based semi-supervised clustering.

Next presented is a probabilistic analysis for bounding the voting score of target constraints in order to obtain more informative constraints. In the general case, a random pair of instances are selected from the entire voting score range of [0,1]. In the restricted case, the lower limit is increased so as to eliminate as many obvious CLs as possible, and the upper range is decreased in order to eliminate as many obvious MLs as possible. The remaining interval is expected to contain a large number of border constraints. Formally, we define a constraint l with a voting score s_l in the co-association matrix, as a border constraint if the probability that l belongs to ML is close to the probability that l belongs to CL, namely $P(l \in ML|s_l) \approx P(l \in CL|s_l) \approx 0.5$ given that $P(l \in ML|s_l) + P(l \in CL|s_l) = 1$. Even though the exact forms of these two probabilities are hard to attain, assuming there exists a range where most border constraints occur, we can reason about an important property of such a range. Let $w = [a, b], 0 \leq a, b \leq 1$ be a window over which those border constraints will be drawn, then if an occurrence amounts to drawing a constraint from w, as each drawing has an approximately equal probability of being ML or CL, the distribution of drawing ML n_{ML} times out of total n_t draws is apparently $P(n_{ML}) = \frac{n_t!}{n_{ML}!(n_t - n_{ML})!}(0.5)^{n_t}$ which has the mode $n_{ML} = n_{CL} = n_t/2$. Therefore the problem of identifying border constraints can be relaxed to finding a window $w = [a, b]$ over which there exists an approximate equal probability of drawing either an ML or a CL, namely, $P_{a,b}(l \in ML) \approx P_{a,b}(l \in CL)$, or equivalently, $\Delta_{a,b} = \left|P_{a,b}(l \in ML) - P_{a,b}(l \in CL)\right| \approx 0$.

For a specific data set, we can calculate $P_{a,b}(l \in ML)$ and $P_{a,b}(l \in CL)$ within window $[a, b]$ by counting the numbers of MLs and CLs whose voting scores are between a and b, and then dividing them by the total number of possible constraints. In a purely random generation of constraints, where the window w is set to [0,1], the scores of $\Delta a,b$ are listed in the first column of Table 8.1. For all of them, especially for the SUGAR and STUDENT, the difference between $P_{0,1}(l \in ML)$ and $P_{0,1}(l \in CL)$ is significant.

In contrast, by narrowing the window w and moving it away from 0, $\Delta_{a,b}$ will be usually lower than random. As can be seen in the second data column of Table 8.1, when we applied a narrower bounded window $w = [0.1,0.5]$, $P_{0.1,0.5}(l \in ML)$ and $P_{0.1,0.5}(l \in CL)$ are closer in most data sets. Therefore it can be hypothesized that the bounded constraint selection will generate more informative constraints in most cases. Finally it should be noted that this bounded strategy based on using a co-association matrix for constraint generation is generic and should therefore be applicable across different data types and proximity metrics.

Table 8.1 The probability difference between drawing ML and CL within two windows, [0,1] and [0.1,0.5].

	$\Delta_{0,1}$	$\Delta_{0.1,0.5}$	$\Delta_{0,1} > \Delta_{0.1,0.5}$
Tr11	0.6281	0.2450	✓

Tr12	0.6663	0.4317	✓
Tr23	0.4311	0.5206	×
Tr31	0.7218	0.4699	✓
Tr41	0.652	0.1338	✓
Tr45	0.5013	0.2867	✓
STUDENT	**0.9160**	**0.5599**	✓
SUGAR	**0.9169**	**0.4890**	✓

8.5 Consensus combination of constrained partitions

Once constraints are selected and defined as either ML or CL, another set of partitions $\{CP^i\}_{i=1}^G$ are generated using a collection of constrained clustering algorithms based on the constraints generated from M.

Wagstaff [Wagstaff01] and Davidson et al. [Davidson06a], demonstrated that most constrained clustering algorithms are sensitive to the order in which the constraints are applied and in which instances are assigned to clusters. In fact some orderings may lead to clusterings that are worse than those produced by a fully unsupervised algorithm. By using consensus integration of multiple constrained partitions, incorrect placements in one clustering can be outvoted by correct placements in others. To enhance the diversity of the clustering ensemble, it is helpful to permute both the order of instances and constraints before applying a constrained algorithm. In our framework, a new co-association matrix M' is derived from the constrained ensemble $\{CP^i\}_{i=1}^G$, and then average-link agglomerative hierarchical clustering [Jain88] is applied on the co-association matrix to obtain the final clustering P*. Although several choices exist for clustering the co-association matrix $M'M$, such as variants of hierarchical agglomerative clustering, spectral clustering, and graph partitioning algorithms, the average-link algorithm was chosen as it was demonstrated in our experiments to be very stable across ensembles of various sizes and characteristics.

8.6 Experiments

8.6.1 Experiment setup

To increase the variability of the constrained partitioning used in phase two of our framework, two consensus-based constrained partitioned clustering algorithms were selected: COPK-means [Wagstaff01] and PCK-means [Basu04a]. The original COPK-means and PCK-means are based on the K-means algorithm and differ only in their methods for constrained instance assignment, which corresponds to step 2a in the algorithm that has been described in Figure 6.3. COPK-means enforces a hard constrained instance assignment in which an instance x from cluster C_i is assigned to the nearest cluster C_j if this assignment results in no conflict, namely, for each $(x, y) \in ML$, $y \in C_j$, and also for each $(x, y) \in CL$, $y \notin C_j$. PCK-means, on the other hand, applies a soft constrained assignment for which instance x is assigned to whichever cluster C_j maximizes

$$x^T\mu_j - \sum_{(x,x_i)\in ML} w^*\delta(l_i \neq j) - \sum_{(x,x_i)\in CL} w^*\delta(l_i = j)$$

where l_i is the cluster label for constraint-involved instance x_i, δ is a binary function that returns 0 if boolean variable is false and 1 otherwise, and w* is a penalty parameter, which is used to adjust the "hardness" of the constrained assignment and set to 0.001 in our experiments [Basu04a].

Unlike many previous experiments [Basu04a, Tang07], our implementation of COPK-means and PCK-means not only applies constrained instance assignment to the usual batch assignment stage (step 2a in Figure 6.3), but also to the incremental optimization stage (step3a in Figure 6.3). Other clustering techniques such as metric learning enhanced constrained clustering, including HMRFK-means [Basu04b] or MPCK-mean [Bilenko06], were not used to help build the ensemble, as they are computationally expensive and exhibit coarse approximations for clustering high-dimensional data in which using a covariance matrix to calculate Mahalanobis distance is not precise. Furthermore, as the experiments in [Tang07] have empirically shown, when applied to document clustering, these approaches did not introduce significant quality improvement.

Two experiments were conducted to validate the effectiveness of our framework. The first focused on comparing the framework's performance with a state-of-art method SCREEN, and the second investigated the utility of bounded constraint generation.

Experiment 8.A The proposed framework's performance with small number of constraints

Objective

To validate the effectiveness of the proposed framework, particularly when compared with that of SCREEN [ref]

Experimental Method

Bounded constraints were generated from 10 to 100 with an interval of 10 and applied to three approaches of COPK-means, PCK-means, and consensus clustering of constrained ensemble, denoted as B-COPK, B-PCK, and B-Cons respectively in Figure 8.3.

Results

The performance was then compared with the SCREEN approach reported by Tang et al [Tang07]. As there were insufficient details in the paper to reproduce the SCREEN experiments against additional datasets, the comparison was only made against the six TREC data sets for which results were extrapolated from the published graphs [Tang07]. The results are shown in Figure 8.3. Consensus integration over a constrained partition ensemble (B-Cons) significantly outperformed base algorithms COPK-means and PCK-means, and was generally comparable to SCREEN. Furthermore, excluding STUDENT and SUGAR for which SCREEN results were not

available, the data sets over which consensus constrained clustering outperforms SCREEN are Tr12, Tr23, and Tr45, which are in fact the three data sets which contain the shortest documents among all six sets.

Conclusions

The results suggest that our proposed approach might be highly applicable for clustering requirements documents, which tend to be relatively short.

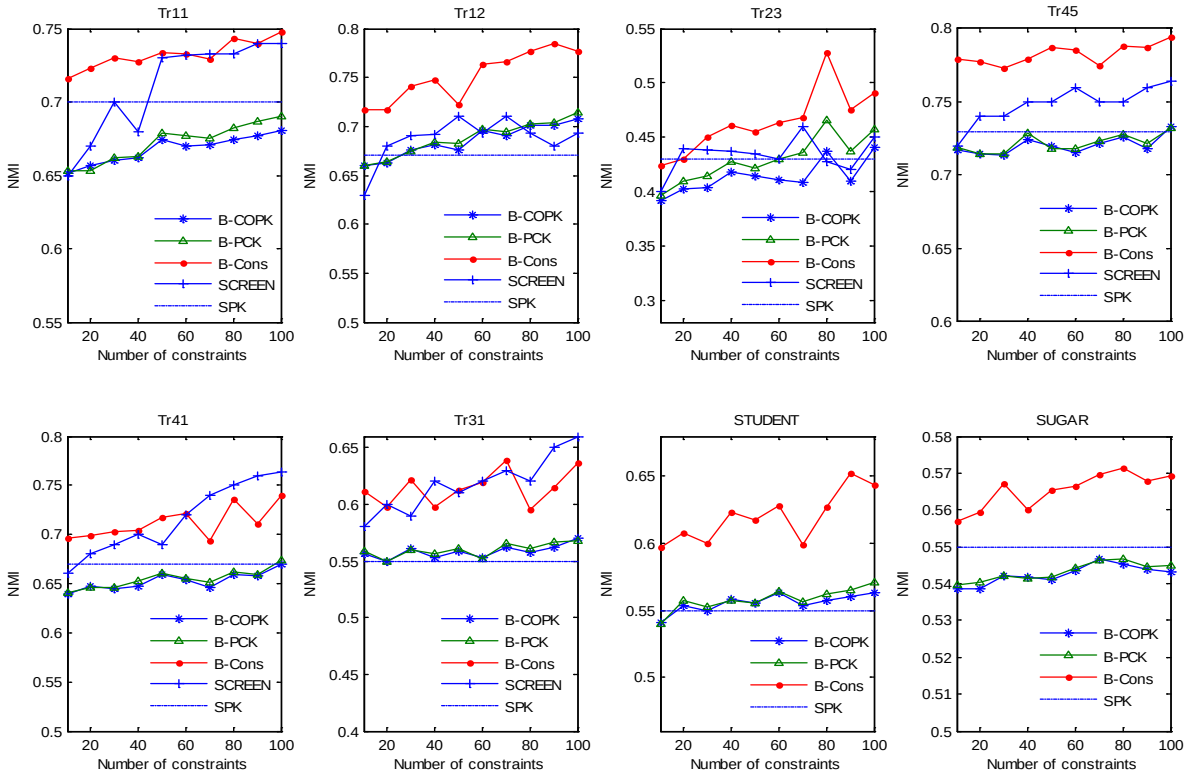


Figure 8.3 Comparison of various constrained clustering algorithms with 10 to 100 constraints.

Experiment 8.B The utility of bounded constraint generation

Objective

To measure the improvement of using bounded consensus based constraint generation over random generation

Experimental Method

Both bounded and random constraints were generated from 50 to 1000 at intervals of 50. As COPK-means tends to suffer from a feasibility problem when presented with a large number of CLs, and because it has been shown to have comparable performance to PCKmeans, it was not used in any of the additional experiments.

Results

The results of this experiment are shown in Figure 8.4. The results of PCKmeans, consensus clustering using bounded and random constraint generation as well as baseline spherical K-means are denoted as B-PCK, B-Cons, R-PCK, R-Cons, and SPK respectively in the plots.

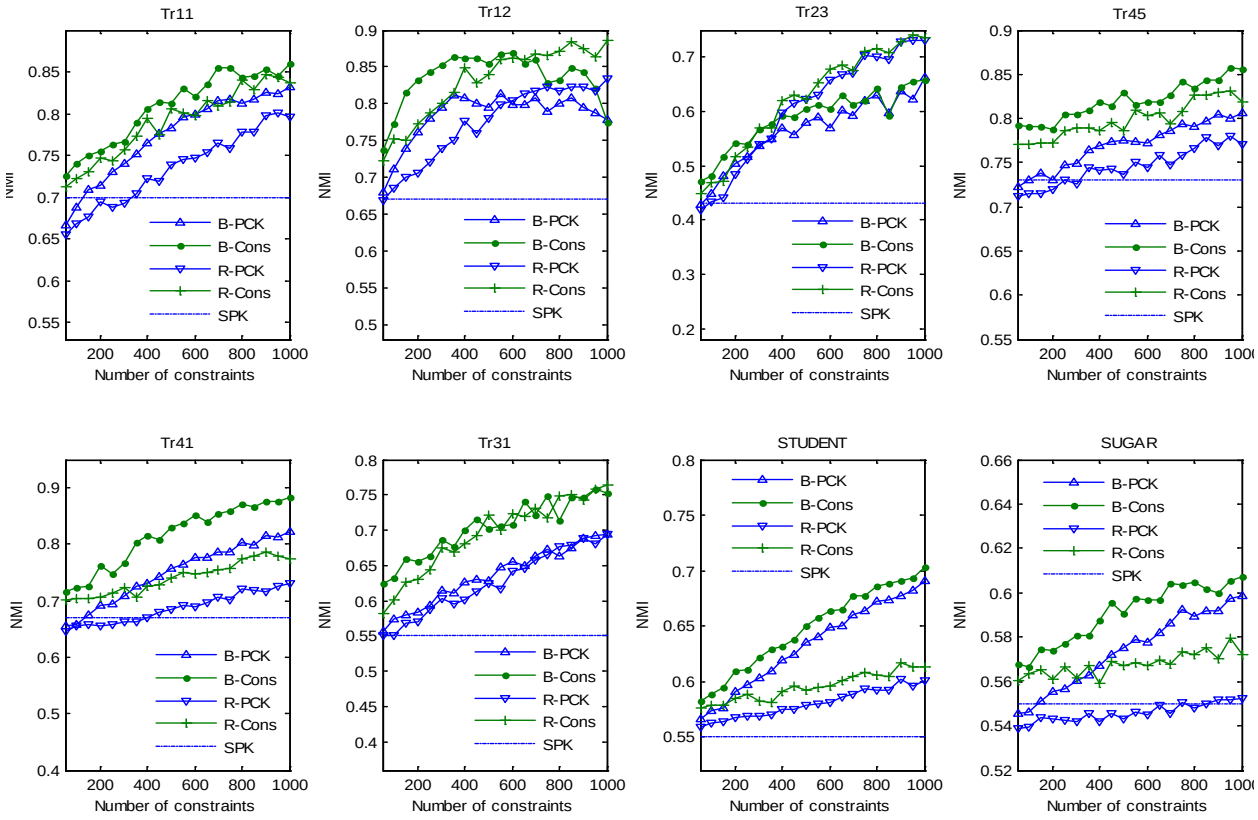


Figure 8.4 Comparison of bounded and random constraint generation with 50 to 1000 constraints.

The TREC datasets returned rather mixed results. For example, the bounded approach appeared to do better for datasets Tr11, Tr41, and Tr45, but did not do well in datasets Tr23 and Tr31. Furthermore, despite a good start in Tr12, there was a significant drop off after about 600 constraints. An initial observation of these results suggested that the bounded approach did best when applied to datasets with a larger number of small clusters, meaning that there could be more borderline cases. For example, datasets Tr11, Tr41, and Tr45 which performed well had 9 or 10 clusters each, while Tr23 and TR31 had only 6 clusters each. TR12, which also started off well and then dropped off had 8 clusters. The results for the two requirements datasets, which are also depicted in Figure 8.4, show a marked improvement obtained through using bounded constraint generation. It should be noted however, that datasets can be clustered at any level of granularity, but that an ideal granularity exists for each dataset in respect to a given task. Our observations are therefore based on fixed levels of granularity.

Conclusions

In the software engineering domain, tasks such as feature extraction and requirements management typically require fine levels of granularity, in order to create manageable clusters which can be used by stakeholders to perform their tasks [Castro08, Cleland-Huang08, Duan07]. As reported in our prior work, SUGAR is clustered at a granularity of 40 clusters resulting in average sized clusters of 25, while Student has 29 clusters containing an average of 13 feature requests. These fine grained granularities suggest that requirements documents are potentially highly suited to the bounded constraint generation.

8.7 Conclusions and future work

This chapter has described a new framework for clustering high dimensional datasets such as requirements documents. The framework adopts a hybrid model which combines both consensus and constrained clustering techniques, and in which constraints are selected that are expected to maximize supervisory potential for improving cluster quality. The reported experimental results demonstrated the effectiveness of this approach especially for clustering short documents into finely grained partitions. These characteristics closely match those of the targeted requirements domain, and in fact the clustering results were especially promising for the SUGAR and STUDENT datasets. Future work will include building a far more extensive set of requirements related datasets and corresponding answer sets, so that we can further assess and fine-tune the usefulness of our framework.

The work in this chapter was primarily motivated by observations described in the last chapter that rudimentary clustering techniques did not produce sufficiently cohesive clusters to support our intended tasks. The clustering improvements obtained through use of this framework have significantly mitigated this problem, to the extent that they are anticipated to support future research and tool development that will enable a move towards higher levels of automation in the requirements engineering domain.

CHAPTER 9. INCREMENTAL STABLE REQUIREMENTS CLUSTERING

9.1 Introduction

One very important phase in software development projects is requirements gathering where business analysts work with project stakeholders to elicit the functional and non-functional requirements of the system. This phase is very critical for every software project, and numerous case studies and surveys have shown that incomplete or incorrect requirements are one of the primary causes of project failure, and lead to millions of dollars in lost revenue every year. Traditionally the process of requirements elicitation is performed using face-to-face techniques such as interviews, brainstorming sessions, and other interactive workshop activities. However, recent advancements in technology have led to a growing trend towards using online forums and wikis to facilitate the requirements gathering process [Decker07]. In these forums and wikis, project stakeholders gather in a virtual meeting place, often asynchronously, to explore and specify the project requirements. Generally, topic-based discussions take place within discussion threads, which are created on demand by the project stakeholders [Cleland-Huang09].

For this forum process to effectively support the requirements process, the discussion threads need to work in much the same was as in-person meetings by drawing together stakeholders with similar interests so that they can explore and articulate their needs for the product under development. In practice, this does not happen very effectively. A survey of feature request forums for seven open source projects, demonstrated that human users do not do a very good job in creating cohesive, and distinct user defined threads.

The survey included a customer relationship management tool named SugarCRM, a UML modeling tool named Poseidon, an Enterprise Resource Planning tool named Open Bravo, a groupware product named ZIMBRA, a web tool for administrating MySQL server named PHPMyAdmin, an open .NET architecture for Linux named Mono, and finally a large web-based immersive game world named SecondLife. All of these forums exhibited a significantly high percentage of discussion threads consisting of only one or two feature requests. For example, as depicted in Figure 9.1, 59% of Poseidon threads, 70% of SugarCRM threads, 48% of Open Bravo, and 42% of Zimbra threads contained only one or two requests. The presence of so many small threads suggests that there were either a significant number of distinct discussion topics, or that users created unnecessary new threads. An initial analysis indicated that the second case was true. For example in the SugarCRM forum, stakeholders' requests related to email lists were found across 20 different clusters, 13 of which contained only one or two comments.

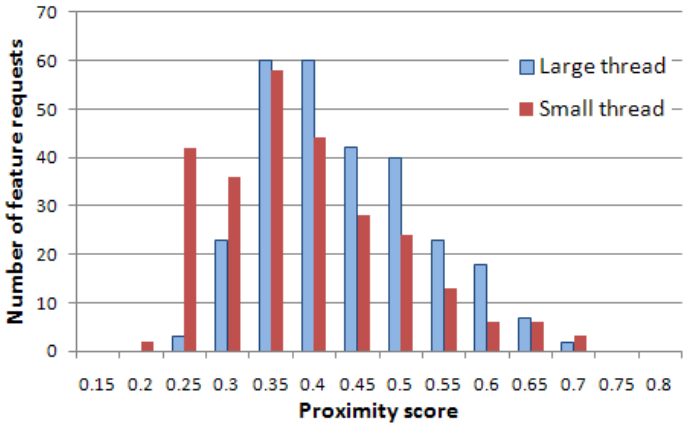


Figure 9.1 A comparison of the extent to which feature requests assigned to individual versus larger threads fit into global topics.

An experiment was designed to determine quantitatively if the users' feature requests and comments that had been placed into new threads, should have been placed into larger pre-existing threads. This experiment was conducted using the data from SugarCRM, an open source customer relationship management system that supports campaign management, email marketing, lead management, marketing analysis, forecasting, quote management, case management and many other features. 1,000 feature requests, contributed by 523 different stakeholders over a two year period and distributed across 309 threads were mined from the open discussion forum. Of these requests, 272 had been placed by users into discussion threads containing two or fewer feature requests, 309 had been placed into groups containing nine or more requests, while the remainder were placed in intermediate sized groups. We clustered all of these feature requests using SPK, and then estimated the degree to which each feature request belonged to a topic by computing the proximity of the request to the centroid of its assigned cluster. We hypothesized that feature requests which represented truly unique topics would have low proximity scores, while those which fit well into a topic would have higher scores. The results for feature requests assigned to small groups containing only one or two feature requests, versus those assigned to larger groups of nine or more requests are graphically depicted in Figure 9.1. A t-test was performed and returned a p-value of 0.0005 which showed that the distributions exhibited a significant difference; nevertheless there is clearly a high degree of overlap between the two distributions, suggesting that many of the features assigned to individual or very small threads could have been placed into larger threads. It is also reasonable to assume that once a stakeholder finds a relevant thread and adds a feature request to it, that their choice of words will be influenced by words found in the existing thread, thereby increasing the similarity of feature requests placed together in shared threads. This phenomenon, which was observed in practice when users often responded directly to previous entries in the thread, could account for some of the differences in proximity scores between the two groupings. In general, the results from the

experiment and our subjective observations suggest that in many cases users made incorrect decisions when they created new threads.

Problems associated with allowing users to define and manage their own threads can be partially addressed through utilizing data-mining techniques to manage both the creation of threads and the placement of new feature requests into threads. However, there are several characteristics of the requirements elicitation domain that make this an interesting and unique data mining problem. As previously stated, the clustering of feature requests is relatively challenging due to a significant amount of background noise, such as spelling errors, poor grammar, slang, long-winded comments, use of non-standard abbreviations, redundancies, inconsistent use of terms, and off topic discussions. Secondly, the requirements elicitation phase of a project is fast-moving and volatile with feature requests arriving as a continual or intermittent stream of ideas. Furthermore, in some projects these feature requests arrive in a random order so that early requests are representative of the full range of potential discussion topics, while in other projects features are explored sequentially by topic and arrive in a more orderly manner. Thirdly, as the generated clusters are used to create and anchor discussion threads, they are highly visible to human users, and to a large extent determine which discussions a user participates in. The clustering algorithm must therefore not only be incremental in order to handle the dynamic environment, but it must also be able to produce relatively stable partitions which minimize unnecessary movement across successive re-clusterings of the data. Finally, the highly interactive nature of the discussion threads provides a natural opportunity for gathering user feedback on the quality of the underlying cluster. However this feedback must be used to improve the quality of the discussion threads without causing excessive volatility across multiple threads.

In summary, the clustering algorithm must deliver high quality clusters despite the incremental arrival of feature requests, and furthermore must incorporate user feedback while minimizing unnecessary movement across clusters. Additionally the algorithm must have a fast running time so as to not interfere with visible forum activities. These objectives give rise to the first question of whether it is possible to increase the stability of the incremental clustering process without negatively affecting its performance or quality, and secondly, the somewhat paradoxical question of whether user feedback can be incorporated in the clustering process in order to change and improve the structure of the clustering, without negatively impacting stability.

The remainder of this chapter is laid out as follows. Section 9.2 introduces the requirements clustering framework and its components. Section 9.3 reports on a series of experiments that were conducted to evaluate whether the requirements clustering algorithm delivers all three objectives of high cluster quality, stability, and high performance. It also describes and evaluates several different approaches for gathering and utilizing user feedback in order to improve cluster quality. Finally Section 9.4 concludes with a discussion of the results and suggestions for future work.

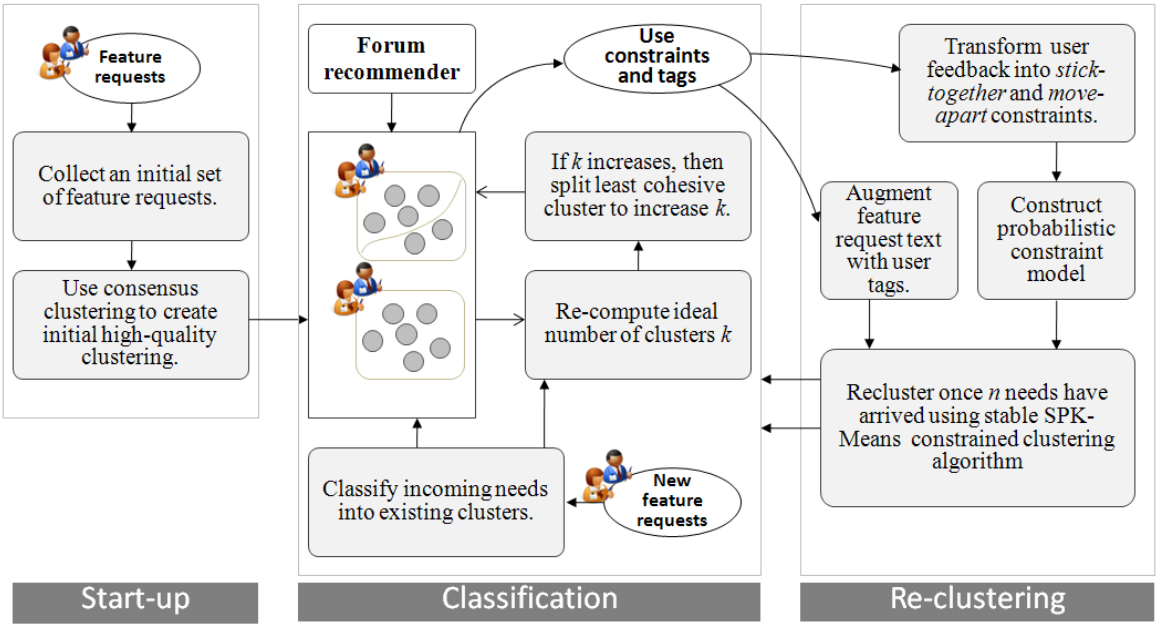


Figure 9.2 The stable, incremental, requirements clustering framework with user constraints [Chuan09].

9.2 The clustering framework

The framework, which is depicted in Figure 9.2, involves three distinct phases of start-up, classification, and re-clustering. During the start-up phase of a project, a web-based collection tool is used to gather an initial set of feature requests from project stakeholders. Once about 50 requests have been received, a robust consensus-based clustering algorithm (discussed in section 7.5) generates the first set of clusters. Subsequent feature requests are then classified into these existing clusters. The framework continually monitors the data to determine when a new topic has been introduced. It then identifies the least cohesive cluster, splits it into two parts, and moves closely related feature requests into the appropriate new cluster. After a predetermined number of new feature requests have arrived, the feature requests are re-clustered using a seed-preserving version of Spherical K-Means that is designed to increase stability while maintaining quality of the clusters. The remainder of this section covers these processes and their underlying algorithms, which are described briefly since most of them have been introduced in the previous chapters.

9.2.1 Preprocessing of feature requests

Feature requests are first preprocessed to remove common words such as “this” and “because” which are not useful for identifying underlying themes. The remaining terms are then stemmed to their grammatical roots, and then weighted by td-idf. The similarity between each pair of request vectors $a = (a_1, a_2, \ldots, a_W)$ and $b = (b_1, b_2, \ldots, b_W)$ is then computed using the Cosine similarity measure:

$$sim(a,b) = \frac{\sum_{i=1}^{W} a_i \cdot b_i}{\sqrt{\sum_{i=1}^{W} a_i^2 \cdot \sum_{i=1}^{W} b_i^2}}.$$

where *W* is the total number of terms in the data set.

9.2.2 Granularity determination

The clustering framework utilizes a technique proposed by Can [Can90] to predict the appropriate number of clusters for the current dataset. Can's approach considers the degree to which each feature request differentiates itself from other requests and the ideal number of clusters *K* is computed as follows: $K = \sum_{x \in D} \sum_{i=1}^{W} \frac{f_{x,i}}{|x|} * \frac{f_{x,i}}{N_i} = \sum_{x \in D} \frac{1}{|x|} \sum_{i=1}^{W} \frac{f_{x,i}^2}{N_i}$, where $f_{x,i}$ is the frequency of term t_i in artifact x, $|x|$ is the length of the artifact, and N_i represents the total occurrence of term t_i .

9.2.3 Clustering algorithm

Following an extensive series of previously conducted experiments discussed in chapter 8, two-stage SPK, which exhibits fast running-times and returns relatively high-quality clustering is adopted as the basic clustering algorithm. To generate the initial clustering and to incorporate of feedback, consensus clustering based on AHC integration of co-association matrix *M* is used to robustly produce a high quality clustering.

9.2.4 Managing new feature requests

Following the arrival of each new feature request the ideal granularity is recomputed to determine if a new cluster should be added. To add such a cluster in a way that preserves stability of existing clusters while minimizing clustering time, our approach identifies the least cohesive cluster, and then bisects it using SPK with K = 2. The cohesion is measured by the standard SPK objective function as $CH(C_i) = \sum_{x \in C_i} s(x, \mu_i)$ where μ_i represents the centroid of cluster C_i. After the split, feature requests from neighboring clusters are re-evaluated to determine if they exhibit closer proximity to one of the two new centroids than to their own currently assigned centroids. If this is the case, they are reassigned to the relevant cluster. To ensure continued clustering quality, the entire dataset is re-clustered periodically after a fixed number of new feature requests have arrived. Re-clustering is performed using a modified SPKMeans algorithm that we name the Seed-Preserving SPKMeans, designed to minimize the movement of feature requests between clusters, through re-using the current set of centroids as seeds for the new clustering.

9.2.5 Incorporating user feedback

The requirements clustering framework also takes advantage of the interactive nature of the discussion threads in order to gather user feedback about the quality of each of the clusters. Users who feel that a particular feature request does not belong in a given thread can vote for its removal. Likewise, users who want to encourage a set of feature requests to stay together in future re-clusterings can vote to keep them together.

The framework focuses on pair-wise Must-Link (ML) and Cannot-Link (CL) constraints, indicating respectively whether a pair of instances should be placed in the same or in separate clusters. The consensus-based constrained clustering algorithm proposed in Chapter 8 is used to integrate pair-wise constraints.

The feedback provided by the users in a forum is used to generate pair-wise constraints such that any two feature requests that users specify as belonging together in the cluster will be transformed into an ML link, and any pair of feature requests for which one belongs and one does not belong will be transformed into a CL link. However, one potential limitation of this approach is that gathering user feedback within the context of a discussion forum means that only two of four possible scenarios can be captured. These four scenarios could be described as *stick-together*, representing feature requests that are already correctly clustered together, *move-apart* representing feature requests that are incorrectly clustered together, *stay-apart* representing feature requests that are correctly placed in different clusters, and *move-together* representing those that are incorrectly clustered apart and should be placed together. By capturing constraints within the context of a forum, our mechanism captures only *stick-together* and *move-apart* constraints. Experiments reported in the following section evaluate the impact of this limitation on the effectiveness of the constraints to improve quality, and also to determine if in fact, the *stick-together* constraints increase stability. Furthermore, to compensate for the lack of *move-together* links, which are considered relatively informative for improving cluster quality, the requirements framework also allows users to tag the feature requests in order to augment their meaning and further improve the benefits of incremental and constrained re-clustering.

9.3 Experimental evaluation

This section presents a series of experiments that were designed to evaluate the clustering framework's ability to quickly and efficiently deliver cohesive, distinct, and stable clusters.

9.3.1 Validation metrics

Cluster quality was measured using the Normalized Mutual Information (NMI):

$$NMI(P^a, P^b) = \frac{I(P^a, P^b)}{[H(P^a) + H(P^b)]/2}.$$

Two different metrics were used to measure stability. They are described using the following notation. T_a and T_b represent two consecutive incremental stages during clustering, P^a and P^b represent clusterings generated during T_a and T_b respectively, and C_i^a and C_j^b are the clusters in P^a and P^b. For most of the experiments, the well-known Jaccard index (JAC) is used to measure stability. In P^a and P^b, pairs of feature requests fall into 4 classes labeled from a-d respectively as follows: a: those assigned together in both P^a and P^b, b: those assigned together in P^a but not in P^b, c: those assigned together in P^b but not in P^a, and finally d: those assigned separately in both P^a and P^b. JAC is then defined as

$$JAC = a/(a + b + c)$$

A second metric measures the percentage of feature requests moving per iteration (PMPI), and is computed as the number of feature requests that change cluster from one increment to the next over the total number of feature requests in the current dataset. NMPI provides a more intuitive notion of stability than JAC. However, when new seeds are used to re-cluster a dataset there is no simple way to determine whether a feature request moves cluster between P^a and P^b, and so measuring movement is difficult. In contrast, the seed-preserving approach retains the original centroids of P^a in P^b, and so it is possible to determine whether a feature request stays in the same cluster or not.

9.3.2 Improving Stability

The first series of experiments compared the performance, quality, and stability of the seed-preserving approach, versus the standard approach in which new seeds were created at each re-clustering.

Experiment 9.A Validation of seed-preserving approach

Objective

To compare the stability, quality, and performance of the seed-preserving approach versus a non seed-preserving approach

Experimental Method

The basic algorithm follows the incremental re-clustering process described in section 9.2. Each experiment was repeated 10 times, and the average result is reported. In addition, each approach was evaluated for two different arrival orders of feature requests in order to determine if stability was achieved in each of these potential scenarios. In the first case, the ordering of feature requests was randomly selected, while in the second case feature requests were clustered using standard SPK, and then placed into a randomly ordered queue. Feature requests were then randomly selected from the first cluster in the queue until no more remained. They were then randomly selected from the next cluster, and so on. This second case simulated the scenario in which a requirements project explored feature requests in some logical ordering of topics.

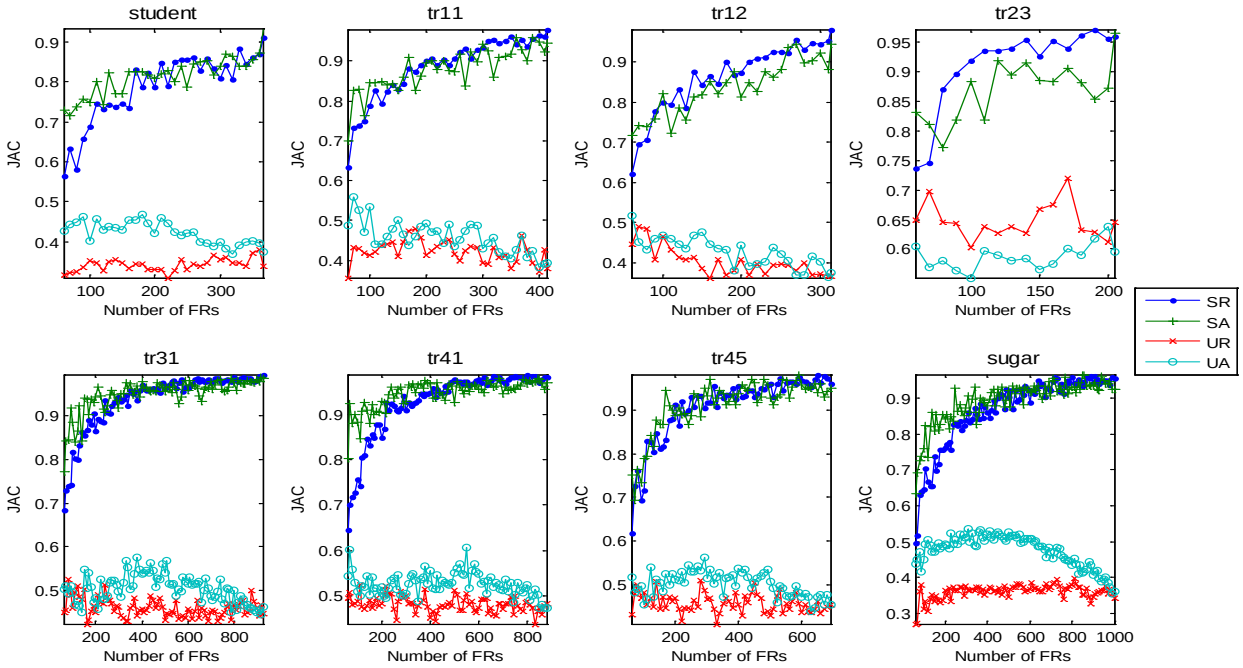


Figure 9.3 Stability of different orderings of feature requests using seed-preserving and standard re-clusterings with increment of 10.

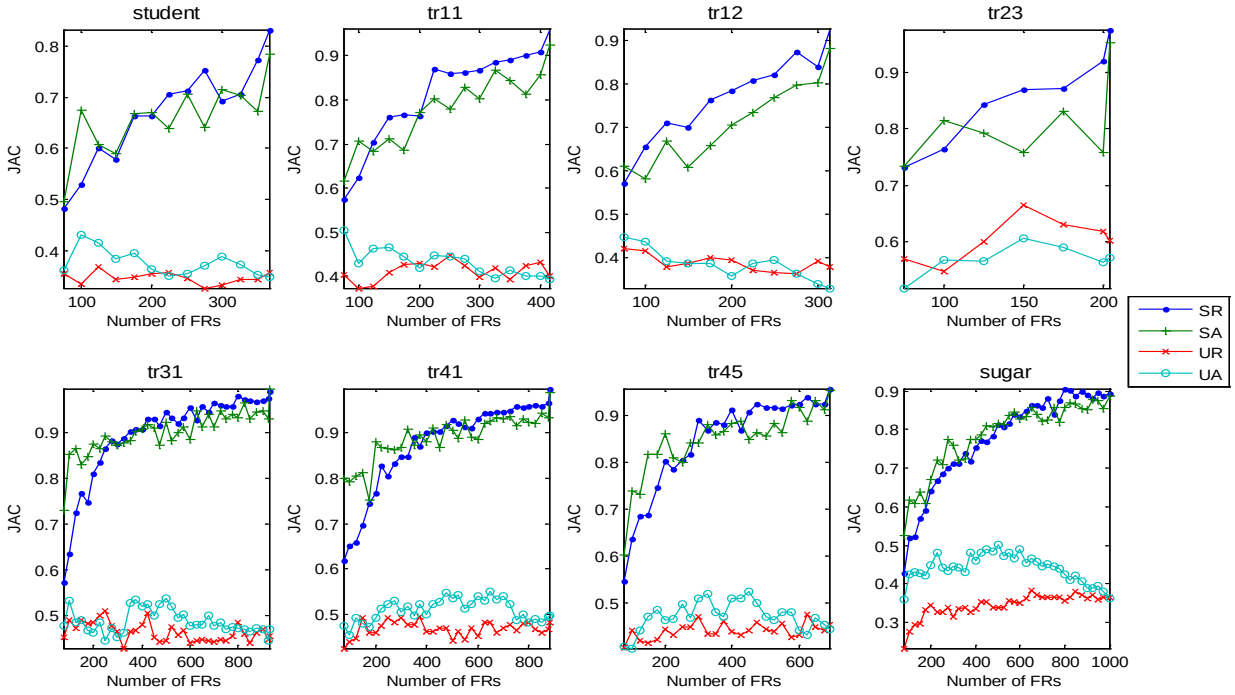


Figure 9.4 Stability of different orderings of feature requests using seed-preserving and standard re-clusterings with increment of 25.

Results

Performance

Table 9.1 reports the performance of the seed-preserving incremental approach versus using new seeds for each re-clustering, and clearly demonstrates that the seed-preserving method significantly decreased the running time of the algorithm. For example, the running time of the SUGAR data at increment sizes of 25, was decreased from almost 109 seconds to just under 7 seconds.

Table 9.1 Total Time Spent Clustering for Seed Preserving Clustering versus re-seeding standard (secs).

Dataset	**Increment Size**				**Single clustering**
	1	**10**	**25**	**50**	
STUDENT (366 feature requests)					
Seed Preserving Increments	7.49	0.98	0.62	0.41	0.04
Standard Increments	101.82	10.78	4.74	2.92	0.73
SUGAR (1000 feature requests)					
Seed Preserving Increments	84.54	13.24	6.66	4.27	0.22
Standard Increments	2374.31	249.92	108.62	57.83	6.69

Stability

The experiment used the JAC metric to compare the stability of the seed preserving re-clustering versus standard re-clustering.

Results reported in Figure 9.3 and Figure 9.4 plot the stability of the clusterings measured using the JAC index, against the number of feature requests in the current dataset for seed-preserving clustering with random arrival (SR) and ordered arrival (SA), and standard clustering with random arrival (UR) and ordered arrival (UA). In each of the eight datasets, the seed-preserving approach was demonstrated to outperform the standard method. A comparison of the results from the random arrival versus ordered arrival of feature requests showed that feature requests that arrived randomly initially exhibited lower levels of stability, but then after several increments matched or outperformed the stability of the ordered arrivals. This suggests that in the case of random arrival ordering, early feature requests caused some initial reorganization, but that once the primary topics were discovered, the clusters began to stabilize. A second

observation is that in the case of random arrival order, the clusters stabilized significantly after arrival of approximately 20-30% of the total feature requests. This suggests that the initial instability might be avoided if there were a longer start-up period for gathering feature requests before the initial clusters are formed.

To provide a more intuitive view of stability, Figure 9.5 depicts the PMPI metric showing the percentage of feature requests that changed cluster from one increment to the next for SUGAR and STUDENT datasets. As can be seen from the graph, in early iterations there is movement of 10-13%, while in later iterations of the SUGAR data this decreases to 2-6%. STUDENT did not entirely stabilize, however this is probably due to the small size of the dataset.

Quality

The NMI scores of the final clusterings in comparison to the answer sets were then computed to determine whether the seed-preserving algorithm was detrimental to the quality of the clustering. Again, results are reported for both ordered and random arrival orderings of the feature requests. Results are reported in Table 9.2 for seed-preserving in random order, seed-preserving with ordered arrival, standard with random arrival, and finally standard with ordered arrival. These results showed insignificant differences between the quality of seed-preserving versus non-seed preserving clusterings for both the ordered and random arrival orderings. However, in 75% of the cases tested the seed-preserving approach had slightly higher NMI scores that the traditional clustering method, indicating no general loss in cluster quality when this approach was used.

Table 9.2 Cluster quality, measured using NMI, comparing the seed-preserving versus standard clustering methods for both random and ordered arrivals of feature requests.

	Increment size of 10				Increment size of 25			
Dataset	**Seed preserving random**	**Seed preserving ordered**	**Standard random**	**Standard ordered**	**Seed preserving random**	**Seed preserving ordered**	**Standard random**	**Standard ordered**
STUDENT	0.593	0.590	0.597	0.590	0.592	0.580	0.589	0.589
tr11	0.602	0.593	0.589	0.590	0.615	0.596	0.604	0.594
tr12	0.581	0.584	0.576	0.567	0.578	0.582	0.585	0.579
tr23	0.476	0.470	0.464	0.471	0.465	0.482	0.477	0.481
tr31	0.512	0.520	0.511	0.512	0.517	0.517	0.512	0.510
tr41	0.615	0.614	0.604	0.611	0.609	0.607	0.607	0.609
tr45	0.649	0.654	0.64	0.638	0.653	0.651	0.638	0.640
SUGAR	0.559	0.557	0.554	0.556	0.560	0.55	0.553	0.554

Conclusions

These results demonstrated that the seed-preserving algorithm effectively maintained the quality and performance of the clustering, while significantly improving stability.

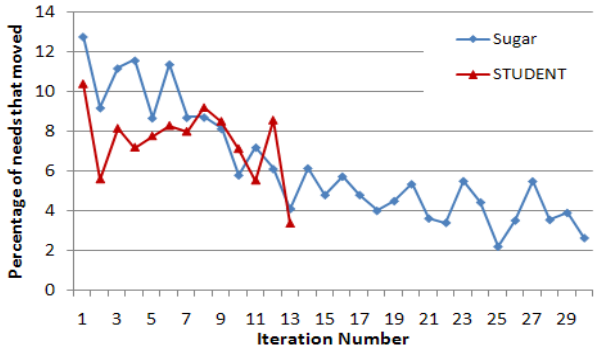


Figure 9.5 Percentage of Feature Requests moved per iteration.

9.3.3 Improving quality through user feedback

The second series of experiments were designed to evaluate whether user feedback could be used to increase the quality of the clusters without negatively impacting quality. As previously explained, the requirements gathering forum provides a natural context for gathering feedback from users in an incremental fashion as users interact with the discussion forums. However, this feedback creates a tension in the clustering process. On one hand, the user feedback should cause restructuring of unfocused clusters or misplaced feature requests, but on the other hand, unnecessary change, especially change which negatively impacts cluster quality, should be minimized.

This series of experiments therefore studied the impact of pair-wise constraints inserted during the process of incremental clustering. The impact of these constraints was again evaluated by measuring stability and cluster quality.

Experiment 9.B Validation of seed-preserving approach in constrained clustering

Objective

To study the stability, quality, and performance of the proposed approach used in constrained clustering

Experimental Method

The increment size was fixed at 25, feature requests arrived in random order, and 25 constraints were inserted at each increment.

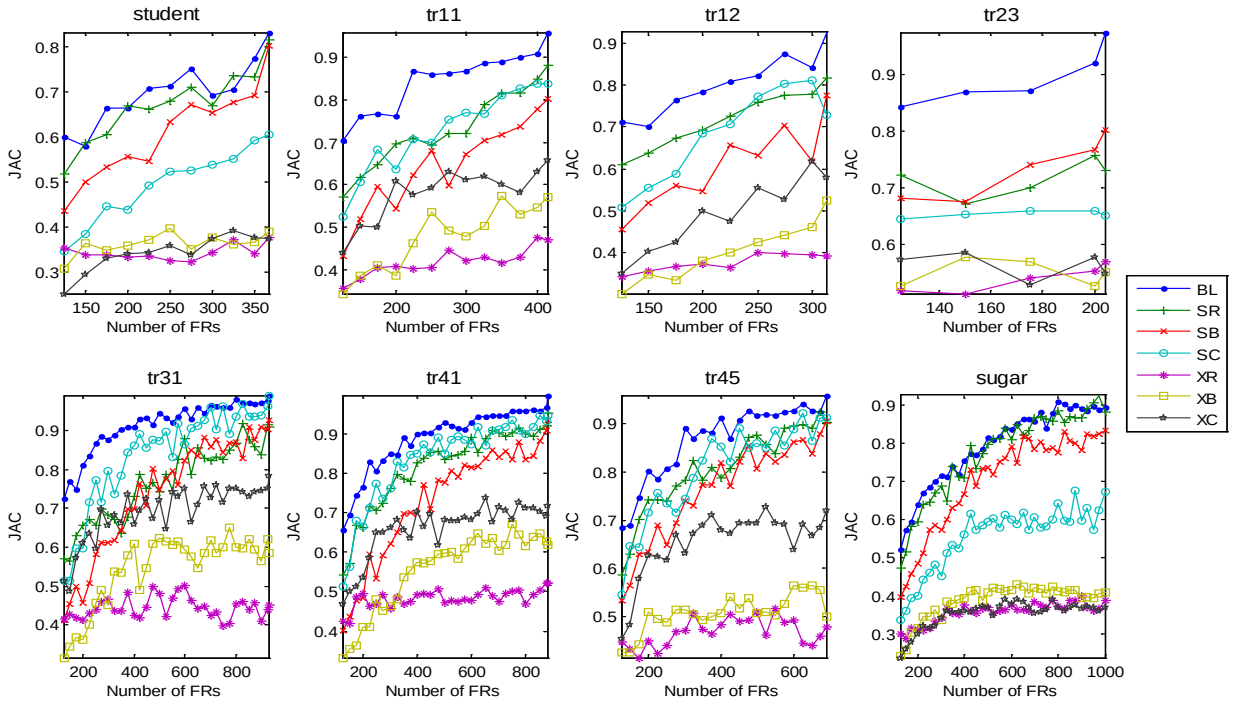


Figure 9.6 Stability of seed-preserving re-clustering algorithm using different constraint generation methods at increment sizes of 25.

Three different methods of constraint generation were explored. The first approach served as a baseline and defined constraints by randomly selecting two feature requests and generating an ML if both artifacts were assigned together in the answer clustering, and a CL otherwise. The second method used bounded constraint generation, for which borderline constraints were selected using the method previously described in section 9.3. This approach is based on the notion that selecting constraints that lie at the boundaries of clusters will maximize the benefits of constrained clustering. Boundary constraints are found by clustering the entire dataset using the consensus clustering method, and then selecting pairs of feature requests with scores in the co-association matrix within a given interval window [a, b]. Based on prior experimentation, the window was set to [0.1, 0.5] for this experiment.

The third method selected constraints only if both feature requests had been placed in the same cluster. This simulated the more realistic case for the requirements domain, in which feedback was elicited from users during their real-time interactions with the discussion threads. As previously discussed, this approach limits the potential constraints to *stick-together*, and *move-apart*, while failing to capture *move-together*, and *stay-apart* constraints.

These three approaches were implemented for both seed-preserving and standard re-clustering methods, resulting in the following six cases: seed-preserving random (SR), seed-preserving boundary (SB), seed-preserving cluster-based (SC), standard random (XR), standard boundary (XB), and standard cluster-based (XC) respectively.

Results

Performance

Incorporating constraints into the re-clustering process increased the runtime of each algorithm. For example the total clustering time for STUDENT using the seed-preserving algorithm and random arrival order increased from 0.62 seconds to 10.28 seconds, and in SUGAR data from 6.66 seconds to almost 262 seconds. As this time is dispersed across multiple clusterings this level of increase in running time is not problematic within the requirements domain.

Stability

Stability results are reported in Figure 9.6 and show that in general the seed-preserving method outperformed the traditional algorithm when user feedback was incorporated into the re-clustering process. The bounded constraint selection technique outperformed the random approach when the standard clustering algorithm was used, but interestingly failed to outperform the random approach when the seed-preserving algorithm was used. This observation might be explained by the fact that boundary conditions are more stable in the seed-preserving approach, and so only considering boundary constraints failed to unearth some of the more important structural changes that needed to occur.

A second interesting observation is that the cluster-based method that utilized only *stick-together* and *move-apart* constraints returned mixed results. In certain datasets, such as TR31 and TR41 it clearly outperformed the other constraint selection techniques, while in other datasets, notably including the two sets of feature requests SUGAR and STUDENT, it performed significantly worse than both of the other two methods.

Figure 9.6 also includes a baseline plot labeled (BL), which represents the case of the seed-preserving clustering with random arrival ordering of feature requests without constraints. In all eight datasets, the use of constraints decreased the stability of the incremental re-clustering; however this decrease was significantly less marked when the seed-preserving algorithm was used. In other words, although stability decreased when user constraints were incorporated, the seed-preserving algorithm still maintained relatively high levels of stability in all three cases.

Quality

The NMI scores of the final clusterings in comparison to the answer sets were computed to determine whether the use of constraints improved the quality of the clustering. Results are reported in Table 9.3. The first column represents the baseline case of seed-preserving clustering with random ordered arrival of feature requests but no user feedback. In every single dataset, all of the user constraint techniques returned higher NMI scores than this baseline case. A comparison of seed-preserving, randomly ordered algorithms showed an average NMI increase of 11.3% across all eight datasets. In general, the bounded and cluster-based constraints led to greater quality improvements than the random approach.

Table 9.3 NMI Scores comparing the quality achieved from various constraint generation techniques.

Data set name	Baseline	Seed preserving random	Seed preserving bounded	Seed preserving cluster-based	Standard random	Standard bounded	Standard cluster-based
STUDENT	0.592	0.610	0.650	0.656	0.611	0.636	0.642
tr11	0.615	0.641	0.700	0.675	0.624	0.702	0.683
tr12	0.578	0.610	0.692	0.682	0.622	0.676	0.681
tr23	0.465	0.499	0.509	0.532	0.523	0.504	0.539
tr31	0.517	0.547	0.582	0.568	0.542	0.583	0.573
tr41	0.609	0.639	0.698	0.670	0.633	0.693	0.672
tr45	0.653	0.682	0.719	0.724	0.673	0.721	0.718
SUGAR	0.560	0.571	0.611	0.598	0.571	0.609	0.608

Conclusions

These results demonstrate that the *stick-together* and *move-apart* constraints performed unexpectedly well in improving the quality of the clusters, but also had some negative impact on the stability of the incremental clustering algorithm. Despite the slightly higher stability of the bounded approach, it has two major disadvantages including a time-consuming pre-processing stage, and the need for users to participate in a series of dedicated training sessions. In contrast, the cluster-based technique requires no special pre-processing and collects feedback from users within the natural context of their regular forum activities.

9.3.4 Improving quality through user tags

The final series of experiments were designed to evaluate whether user tags might help increase the quality of the clusterings. Tags are contributed by the user, and used to augment knowledge of the feature requests. Theoretically they can be used to disambiguate the meaning of a feature request and provide guidance to the clustering algorithm for use in future re-clusterings. As the use of tags does not significantly impact performance, this experiment is evaluated in respect to quality and stability only.

Experiment 9.C Utilizing tags in improving clustering

Objective

To validate the use of tags to improve the quality of clustering

Experimental Method

Unlike user constraints, which can be simulated by selecting pairs of features from the answer set, tags cannot be simulated. A GUI was therefore developed that displayed feature requests within their clusters. A group of five DePaul MS students and faculty were each asked to look through ten randomly assigned clusters from the STUDENT dataset and to tag feature requests. They were instructed that these tags would be used to help improve the placement of feature requests into correct clusters. They were further instructed to only add tags that described the content of the feature requests and not to add emotive-style tags such as "I agree with this request." This enabled the experiment to focus on whether content-based tags could improve the clustering, rather than on filtering out non-useful tag information.

As a result, 141 of the 366 feature requests were tagged with additional information. Some feature requests were tagged by more than one person, and on average each of the tagged feature requests were assigned 9.2 distinct terms. Terms from each tag were then added to the text of the feature request, and subsequent incremental reclusterings were based on the combined text of the feature request plus the user contributed tags.

Results

Stability

As in previous experiments, stability was measured using the JACARD metric. Results are reported in Figure 9.7 and indicate that there was almost no change in stability when tags were combined with each of the three individual constraint generation methods.

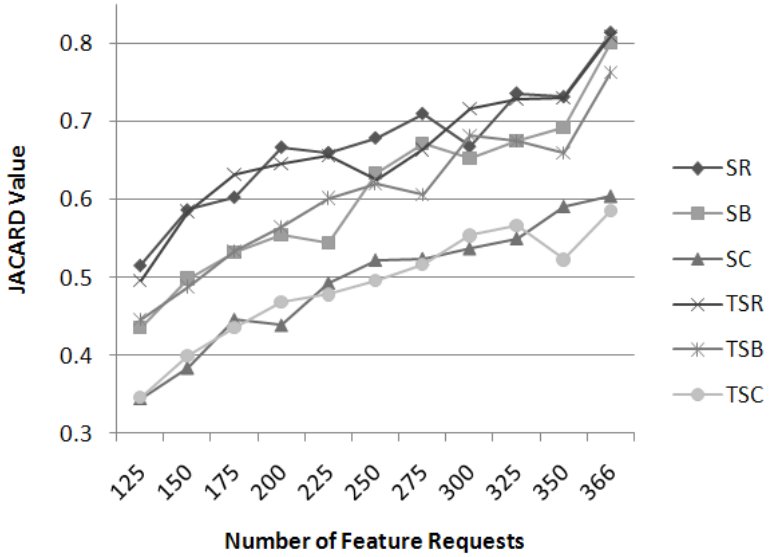
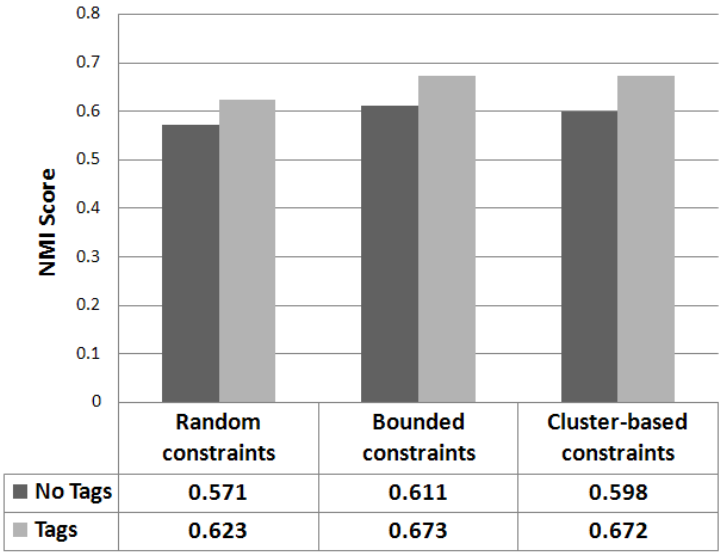


Figure 9.7 Stability of STUDENT data when user tags are added.

Quality

NMI scores were computed for each of the final clusterings produced using tagged feature requests. Results are reported in Figure 9.8, and indicated that the use of tags led to a significant

improvement in cluster quality. For example, the NMI score for cluster-based constraints improved by 12.4% when tagging was used.



	Random constraints	Bounded constraints	Cluster-based constraints
No Tags	0.571	0.611	0.598
Tags	0.623	0.673	0.672

Figure 9.8 NMI scores of tagged FRs clustered incrementally using various constraint mechanisms.

9.4 Conclusion

This chapter described a seed-preserving approach to incremental clustering, and demonstrated that it can deliver high quality, and relatively stable clusters throughout the incremental re-clustering process. Two different user feedback mechanisms based on the use of pair-wise constraints and tagging were also described, both of which can be easily implemented in a requirements gathering forum. Experimental results demonstrated that incorporating user feedback in the form of *stick-together* and *move-apart* links, and augmenting the feature requests with user tags increases the quality of the generated clusters while maintaining much of the stability provided by the seed-preserving clustering algorithm.

The results from this study will be used to guide the choice of clustering and feedback techniques in future requirements gathering forums and wikis. Future work will focus on developing more refined models that enable greater levels of stability and take into consideration additional factors related to user feedback, such as the consensus built by stakeholders concerning both the pair-wise constraints and tags assigned to feature requests.

CHAPTER 10. APPLICATION OF CLUSTERING IN REQUIREMENTS ENGINEERING

In previous chapters, various approaches have been proposed to generate good requirement clusters. This chapter will present concrete examples on how clustering can be used to provide automated support to two requirement engineering tasks: early aspect identification and automated requirements traceability.

10.1 Cluster-based detection of early aspects

10.1.1 Introduction

Aspects represent cross-cutting concerns that are dispersed across the primary decomposition of a system [Duan07a]. For example, at the implementation level, distribution and synchronization code cannot be encapsulated in a single class and is typically spread across several classes. The aspect oriented languages, such as AspectJ [Laddad03], try to modulize these cross cutting concerns by letting programmers define aspect units such as cut points and joint points, and then weaving back these units into the code at compile time [Rashid02, Rashid03]. . At the requirement level, the cross-cutting functionalities or system quality constraints that have chances to be implemented as aspects are called early aspects. The identification of early aspects enables architects to make an early decision as to whether identified cross-cutting concerns should be modeled as aspects or not; thereby minimizing the need for expensive refactoring efforts later in the software development lifecycle.

Several approaches have been suggested for mining early aspects. For example, Rosenhainer proposed a method that required an analyst to inspect the requirements in order to manually identify potential candidate aspects [Rosenr04]. Information retrieval methods were then used to search for related requirements [Antoniol02]. Although numerous researchers have described effective methods for implementing such artifact based searches [Hayes06, Cleland-Huang05b, Marcus03], manually searching for an initial candidate aspect can be labor intensive and error prone.

The Theme/Doc method [Baniassad04, Baniassad06] provides semi-automated support for early aspect mining. An analyst first parses the requirements specification to identify keywords, which are then used by the tool to generate a visual representation of the relationships between behaviors. This view is used by the analyst to identify candidate aspects. Again, the approach is rather labor intensive, as the analyst needs to perform a preliminary manual search for keywords in addition to a later analysis of the candidate concerns. In more recent work Kit et al, have incorporated the use of Latent Semantic Analysis (LSA) to provide more effective support for identifying themes that anchor the aspect discovery process [Kit06].

Sampaio et al. proposed a method that uses natural language processing to first identify viewpoints based on nouns occurring in the specification and then to find actions (i.e. verbs) that

occur across multiple viewpoints [Sampaio05]. This approach can identify unique aspect types, but requires significant user feedback to evaluate viewpoints and assess the feasibility of the candidate aspects. It may also miss aspects if seemingly unrelated action verbs represent the same concern across different viewpoints.

Cleland-Huang et al [Cleland-Huang06, Cleland-Huang07a] used a training set of pre-classified aspects to discover weighted indicator terms that could be used to detect and classify aspects in new document sets. However, their approach only detects aspects for which the classifier has previously been trained and is therefore unable to detect aspects that are unique to individual projects.

10.1.2 Aspect Clustering Engine (ACE)

Although the above approaches are rather labor intensive, they support analysts in the task of identifying potential aspects during early stages of the software development lifecycle. In the author's previous paper [Duan07a] a technique called the Aspect Clustering Engine (ACE) was proposed to detect candidate early aspects in a more automated fashion. There are two essential components of ACE. The first involves concern identification and includes the steps of hierarchical clustering of requirements and selection of the most cohesive clusters as dominant system concerns. The second involves evaluating the degree to which each candidate aspect cross-cuts other clusters. This is measured through two metrics designed to evaluate the degree to which each cluster cross-cuts the other ones. A particular difficulty is that the clusters often end up with trivial or multiple topics, a phenomenon that hides many significant but "recessive" concerns. To tackle this, a heuristic method named dominant term elimination was proposed and implemented in proof of concept form.

For example, consider a set of requirements $R = \{r_1, r_2, \ldots r_n\}$, and a set of terms $T = \{a,b,c,\ldots..,z\}$, such that $r_1 = \{a,b,c,d,\}$, $r_2=\{a,a,b,d\}$, $r_3=\{d,f,g,g\}$, $r_4=\{c,f,f,g\}$, $r_5=\{c,h,i,j,j\}$, and $r_6=\{j,j,k,k\}$. Following the initial clustering phase a set of clusters $C = \{c_1, c_2, \ldots, c_n\}$ is formed as follows $c_1 = \{r_1, r_2\}$, $c_2 = \{r_3, r_4\}$, and $c_3 = \{r_5, r_6\}$. Observation suggests that cluster c_1 is formed around the terms a and b, c_2 is clustered around f and g; while c_3 is formed around the term j. Even though term c occurs in multiple requirements, these requirements are dispersed across each of the three clusters due to the presence of more dominant terms. The temporary removal of dominant terms results in the formation of new clusters representing less dominant themes. For example in this case, a new cluster $c_4 = \{r_1, r_4, r_5\}$ was formed around the term c. ACE therefore places certain requirements, such as r_1, r_4, and r_5 into two disparate clusters, representing functional and cross-cutting decompositions of the system respectively.

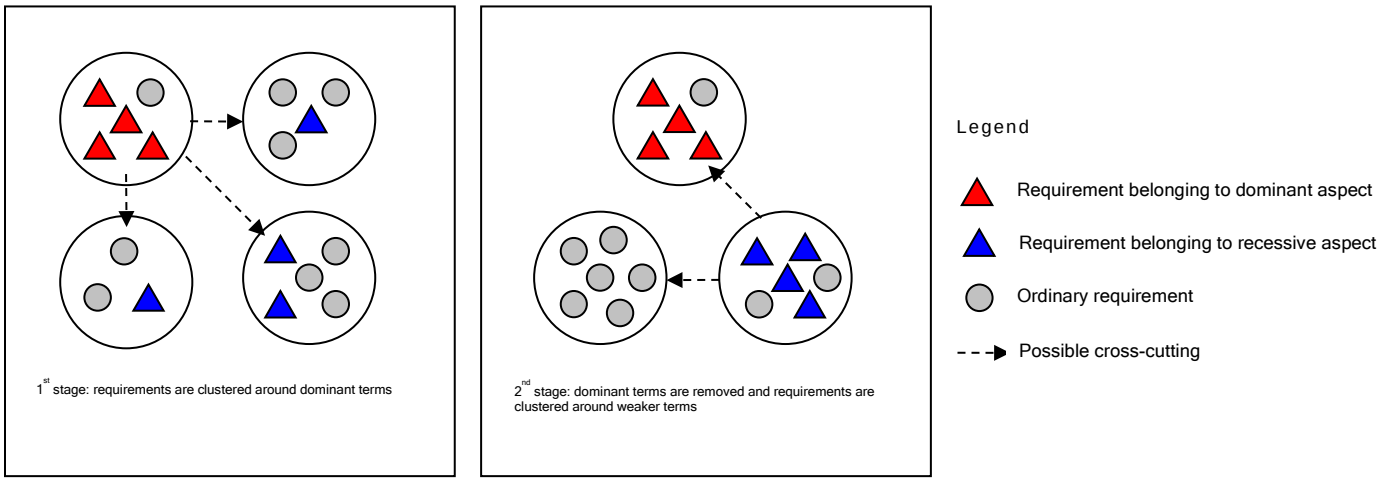


Figure 10.1 Detecting aspects by clustering around strong and weak terms.

The results of ACE highly rely on the quality of clusters. For example, in the earlier work [Duan07a] clusters were generated by hierarchical clustering algorithms, therefore exhibiting relatively low cohesion (see the comparison between hierarchical clustering algorithm and other algorithms in section 9.X), which resulted in returned sub-optimal candidate aspect set. The research from chapter 8, 9, and 10 can effectively improve cluster quality and then enhance the results of using ACE.

10.2 Cluster-based support for automated requirements traceability

10.2.1 Introduction

Requirements traceability is defined as the ability to describe and follow the life of a requirement, in both a forward and backward direction [Gotel95]. It supports a wide variety of activities throughout the software development life cycle, such as the validation & verification of requirements, impact analysis and regression test selection, requirements rationales comprehension, and the retrieval of supplemental information from supporting documents [Gotel94]. In practice, traceability links are typically stored in a requirements trace matrix (RTM) that is constructed manually by analysts. Unfortunately, building and maintaining complete and accurate trace matrices is arduous and labor intensive, and therefore practitioners often fail to implement consistent and effective traceability processes [Cleland-Huang07b].

One solution to the problem of maintaining an RTM is to utilize information retrieval methods such as the Cosine coefficient in vector space model (VSM) [Hayes06], probabilistic network model (PN) [Cleland-Huang05], latent semantic indexing (LSI) [Marcus05], Keyword extraction [Dekhtyar07], or Latent Dirichlet Allocation (LDA) [Dekhtyar07] to automatically generate traceability links. These methods compute similarity scores between pairs of artifacts,

and then those scoring above a certain threshold value are considered candidate links. The tools based on those retrieval methods include Poirot [Lin06], RETRO [Hayes06], and ADAMS [DeLucia06].

Most automated traceability algorithms and tools assume a straightforward and simple model in which all related artifacts are retrieved and displayed to the users as a sequential list, typically ranked in order of similarity score [Hayes06, Lin06], as shown in Figure 10.2. This simple model has serious drawbacks: the retrieved list is unstructured, hard to understand, and more importantly, isolates the retrieved artifacts from their context. These limitations hinder comprehension and evaluation of the generated links.

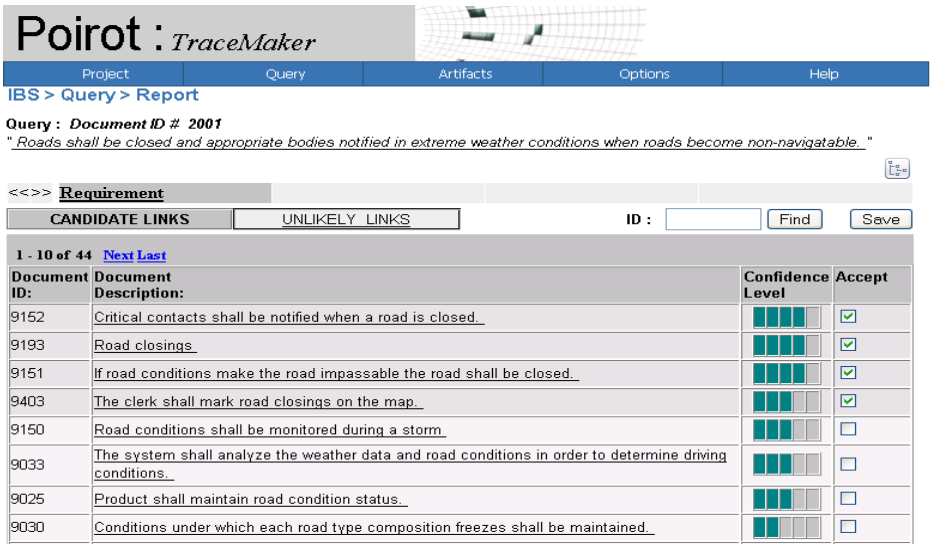


Figure 10.2 Results browsing in typical automated tracing tasks.

10.2.2 Cluster-based trace browsing

Requirements clustering was used to organize traceable requirements in order to increase the understandability of the results and minimize the effort needed to retrieve and evaluate all of the correct links [Duan07b]. This cluster-based trace presentation technique includes the following steps:

1. Create individual clusterings for each set of artifacts. i.e. separately cluster all requirements, all java classes, all test cases etc.
2. When a query is issued against the a set of artifacts, then compute probability scores utilizing the standard probabilistic network algorithm described in numerous papers [Cleland-Huang05a, Cleland-Huang05b, Cleland-Huang06].
3. Group candidate links according to previously generated clusters.
4. Generate a meaningful name for each cluster.

5. Rank the clusters according to their relevance to the query.
6. Display only those elements from each cluster which exhibit a similarity score higher than the standard threshold used to determine candidate traceability links. Make additional cluster elements visible only on demand by the user.

10.2.2.1 Clustering algorithms and granularity determination

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.

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 the widely used K-means most of the time, as already demonstrated by Steinbach et al. [Steinbach00] and Zhao [Zhao02]. However, assumption in bisecting algorithms may embed outliers in clusters, therefore reducing the cohesion of the clustering.

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 10.3 and 2.8 show their score curves and highlight the significant turning points for 3 data sets.

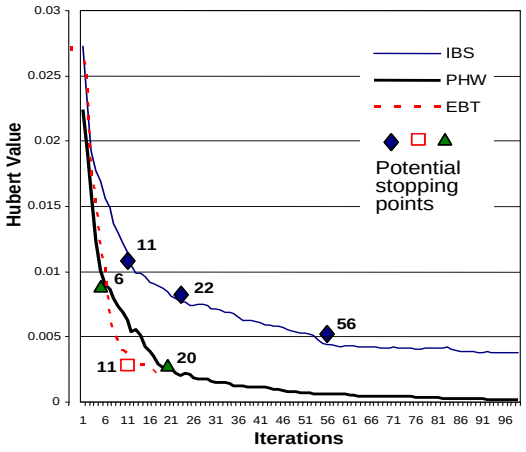
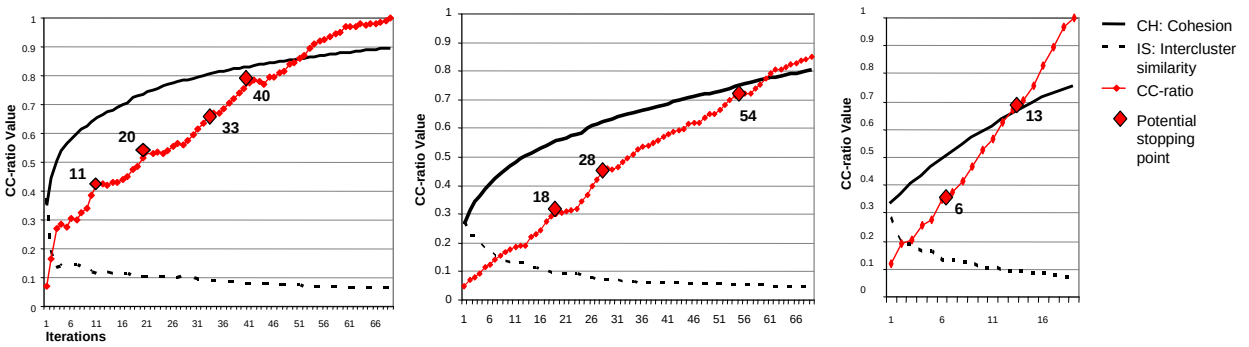


Figure 10.3 Hubert's index against three datasets.



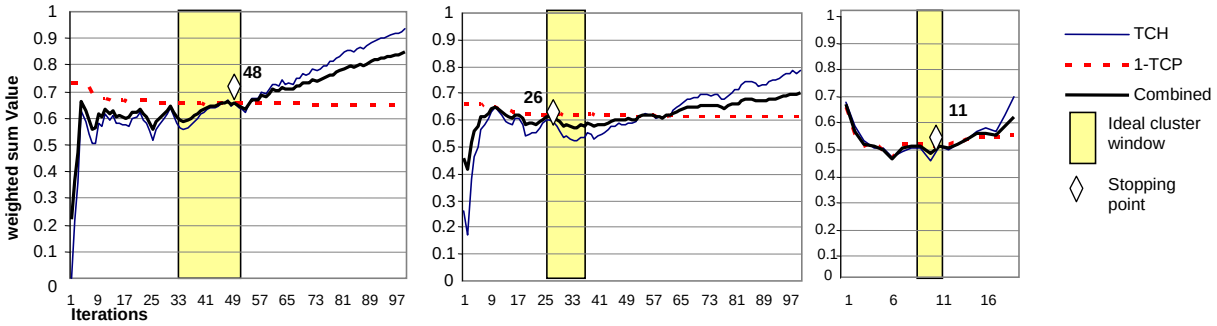
a. Public Health Watcher System. b. Ice Breaker System. c. Event Based Traceability

Figure 10.4 CC index against three datasets.

As pointed out in the previous section, 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. 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 10.5, was then selected.



a. Public Health Watcher System. b. Ice Breaker System. c. Event Based Traceability

Figure 10.5 Theme cohesion and coupling for three datasets showing ideal cluster window.

10.2.2.2 Ranking clusters

The cluster ranks results were ranked as below:

$$R(c_i) = \alpha \frac{|links(c_i)|}{L} + (1-\alpha)\left(\frac{\sum_{d \in links(c_i)} (L - order(d))}{\sum_{i=1}^{L} i}\right)$$

where L represents the number of retrieved links, links(c_i) is the set of retrieved links in cluster c_i, order(d) is the ordered position number of d within the list of retrieved links, and α is the weight assigned to the first component. Intuitively this formula takes into account two primary factors. The first component computes the proportion of total retrieved links that are found in cluster c_i, while the second component takes into consideration the ranking of those links so that a cluster with more highly ranked links would be considered before one with only lowly ranked links. The weighting for α in this initial experiment was determined through analyst evaluation.

There are several significant benefits to clustering trace results. First, as depicted in Figure 10.6, traces are presented to the analyst as part of a meaningful group. The analyst has three options: to accept all the candidate links in the cluster, to reject all of the candidate links, or to make individual decisions. From a usability perspective this enables the analyst to assess similar artifacts at the same time, which can reduce overall 'thinking' time, and coordinate the physical inspection of supplemental documents needed to make a decision. As an added bonus, the number of decision points can be reduced when the analyst is able to evaluate and accept or reject an entire cluster as a consolidated group.

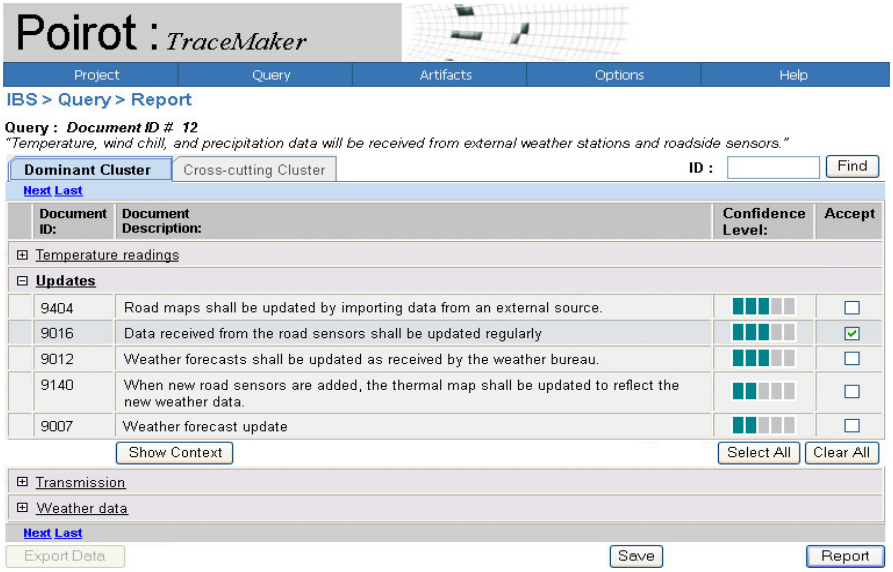


Figure 10.6 Cluster-based trace browsing.

For example the first three clusters in the query depicted in Figure 10.6 include the following candidate links:

Cluster: 14 - Temperature readings *X* (Reject all)

- The system shall report on precipitation readings. (2nd)
- Road temperature readings shall be recorded. (6th)
- The system shall report on road temperature readings. (8th)
- The system shall report on air temperature readings. (9th)

Cluster: 26 - Updates

- Road maps shall be updated by importing data from an external source. (4th) *X*
- **Data received from the road sensors shall be updated regularly (5th)** ✓
- Weather forecasts shall be updated as received by the weather bureau. (10th) *X*
- When new road sensors are added, the thermal map shall be updated to reflect the new weather data. (25th) *X*
- Weather forecast update. (26th) *X*

Cluster: 43 – Transmission ✓ (Accept all)

- **The system shall validate that all road sensors are functioning correctly and transmitting data. (19th)**
- **The system shall receive transmission from road sensors. (3rd)**

Correct links are highlighted in gray, and each requirement's ranking in the candidate list is also shown. In this example the user can reject all artifacts in cluster 14, accept all those in cluster 43, and make individual decisions for those in cluster 26.

Table 10.1 Evaluation of Cluster-Based Traceability.

Goal	Description	No. of links		Number of Clusters		Decision points (clicks)	Effort reduction
		Candidate	True	Total	With true links		
BG1	Roads shall be closed and appropriate bodies notified in extreme weather conditions when roads become non-navigatable.	28	6	14	3	15	**46%**
BG2	Operational costs shall be minimized through efficient route planning.	11	5	6	4	8	**27%**
BG3	Operational costs shall be minimized through efficient management of de-icing inventories.	23	9	11	2	11	**52%**
BG4	The system shall coordinate the maintenance of a fleet of trucks.	44	8	14	4	22	**50%**
BG5	District maps shall be imported and updated from external sources.	14	4	8	2	11	**21%**

To evaluate the effectiveness of trace-base clustering in respect to trace queries issued against requirements, a small analysis was performed in which five business requirements were traced to the lower level requirements of the Ice Breaker System. Results are shown in Table 10.1. As an example, consider the first business goal (BG1), for which 28 candidate links were generated. Six of these links represented true links while 22 represented false links, yielding a precision value of 21.4%. 14 clusters were presented to the user, each of which contained one or more candidate links. Correct links, as validated through an answer set, were found in three of these clusters. In fact for this particular query, four of the good links were found in a single cluster entitled "Road Closings" and that cluster contained no incorrect candidate links. By simply examining the cluster title and glancing over the requirements contained in the cluster, the analyst could quickly determine the correctness of the links and accept them as a group with a single click of a button. The final column in Table 10.1, analyzes the minimum number of decisions (estimated by the number of clicks) needed to accept and reject all of the candidate links. The number of clicks are compared to the number that would have been needed to evaluate traces in a non-clustered results set. As reported in Table 10.1, the number of decision points, reflected in the column entitled "effort reduction", is significantly reduced in each of the queries when clustering is used to organize the results.

As in the ACE's utilization of clustering discussed in the last section, the success of the cluster-based trace presentation is dependent upon the ability to generate high quality clusters.

10.3 Other Requirements Clustering Activities

The previous two sections have introduced specific requirements engineering tasks that are dependent upon availability of high quality clusters. Although not elaborated on in this

dissertation, there are numerous other tasks that could benefit from clustering. These include dynamic structuring and evaluation of a software requirements specification, and feature identification in product line development, etc. The evaluation of a requirement specification includes the tasks of checking the specification's completeness, consistency, and other important attributes. Usually conducted manually in practice, the evaluation process is time-consuming and error-prone. Automated clustering can facilitate this process by, for example, pulling together requirements that have similar themes or are cross cut by the same aspects, therefore making detection of discrepancies or conflicts among requirements much easier. Similarly, in software product line engineering (SPL), based on textual information or certain semantic attributes assigned to requirements, clustering can be used to quickly uncover primary themes and to organize and structure requirements. This complements domain analyses, one of the core activities of SPL engineering methods that require human analysts to manually identify requirement elements and structures [Niu08].

CHAPTER 11. CONCLUSIONS AND FUTURE WORK

This dissertation discusses the clustering of software requirements written in natural language to specify the functionalities and constraints of systems. Unlike other textual artifacts such as articles and research papers, most requirements are highly terse and noisy, therefore posing a serious challenge to producing cohesive requirements clusters. This dissertation addressed this challenge through two distinct lines of research. The first consisted of a study of cluster validation metrics and an empirical evaluation of existing traditional and enhanced clustering algorithms. Several hybrid approaches were also considered. The second line of research focused on improving requirements clustering by incorporating user feedback. Additionally the dissertation studied the problem of incremental requirements clustering to accommodate scenarios in which requirements are generated and clustered in an incremental way.

Study of cluster validation metrics

There are two purposes in studying cluster validation metrics: to identify clustering comparison metrics that have strong discriminate ability, and to detect non clustering comparison metrics that can measure the quality of a clustering without referencing to answer clusterings. Experiment 6A and 6B were designed for the first purpose. Five CCMs were chosen and their baseline distributions, which were comprised of scores calculated between two random clusterings, were inspected. The results from Experiment 6.A indicated that all chosen metrics have symmetric distributions in measuring agreement between random clusterings. It was found in Experiment 6.B that NMI shows a much better discriminate ability than other CCMs therefore it was used throughout the remainder of the dissertation to measure clustering quality. Those findings provided guidelines on how to appreciate the significance of a score calculated by a certain metric. The research for the second purpose examined the correlation between CCMs and a few non-CCM metrics in order to choose the non-CCM quality metrics that exhibited the high correlations with CCM. Unfortunately, as shown in Experiment 6.D, only moderate positive or negative correlations were found between CCMs and non-CCMS, meaning that without an answer clustering, measuring clustering quality by use of non-comparison metrics is not accurate.

Evaluation of existing traditional algorithms in clustering requirements

In Experiment 7.A and Experiment 7.B, four agglomerative hierarchical clustering (AHC) algorithms, a divisive hierarchical clustering, bisecting hierarchical algorithm, and four variation of spherical K-means clustering (SPK) were experimentally evaluated. It was found that proximity-based hierarchical clustering algorithms are not effective in directly clustering high-dimensional and noisy documents such as requirements, whereas a two-stage spherical K-means (SPK) algorithm gave the best clustering on eight data sets. Moreover, through correlation analyses it was suggested that the smaller the average size of documents, or less rich the vocabulary of a data set, the more beneficial the incremental optimization becomes when using

spherical K-means. Since terseness is a characteristic of most requirements data sets, it is highly recommended to include incremental optimization in K-means clustering of software requirements.

Evaluation of enhanced clustering algorithms in requirements problem

Several enhanced clustering methods were evaluated. Experiment 7.D studied the effectiveness of indexing requirements by SVD or PCA in clustering requirements. It was found that in general SVD outperformed PCA, but the effects of both varied across different data sets. Given the huge increase of computational cost of using PCA and SVD both before and during clustering, the approach of clustering after matrix decomposition is therefore not recommended when clustering software requirements.

Two methods to enhance SPK were investigated. A sampling method was proposed to utilize the trained units from SOM without degrading the quality of the SOM. After training a Euclidean distance based SOM, the method sampled K units from the SOM array as initial centroids for SPK, each of which was randomly selected from K areas partitioned from a SOM. As indicated in Experiment 7.E, the effects of SOM-seeded SPK varied for different data sets; however even in the cases that showed improvement, this improvement was minor. Another enhancement method is pipeline hybrid clustering, where SPK is applied to a set of intermediate clusters that are generated from AHC algorithms. Experiment 7.G showed most data sets exhibited a varied extent of improvement. An analysis was also provided to explain the cause of the improvement. As Figure 7.9 and 7.11 suggested, compared with AHC algorithms, SPK is good at separating dissimilar instances, but it sometimes fails to assign similar instances to the same groups, which partly explains why the combination of AHC and SPK achieved certain improvements.

The consensus clustering was also evaluated. Experiment 7.H studied in sub-sampling clustering ensemble generation, what proportion of the ensemble size should be used in order to generate the best possible clustering. It was found that very low scores of α produced worse results than higher scores of α but the differences between different α scores were not substantial for values over 0.6. Also, clustering results were less sensitive to the changes of size of ensemble R than to α. Next the effectiveness of consensus clustering was compared with that of the standard two-stage SPK in Experiment 7.I. Consensus clustering results were always above the mean obtained using SPK, an observation that is highly significant because it means that consensus clustering is more consistent and robust than basic SPK. A further analysis indicated all of the datasets had similar distributions of ML and CLs for the original proximity matrices, but the scores in the co-association matrix provided a clearer differentiation between MLs and CLs by boosting ML scores. This implies that the pair-wise scores compiled from the consensus algorithm tend to approach their “true” similarity scores.

Constrained requirement clustering

As pointed out at the end of Chapter 7, the terseness not only makes it difficult to cluster requirements using basic clustering algorithm, but also constrains the effect of various enhancement approaches. In other words, due to the characteristic of software requirements, there is a limit to what unsupervised clustering techniques can achieve. The second line of research studied ways to improve requirements clustering by incorporating user feedback. The type of feedback studied included pair-wise constraints, which come in two types, ML and CL, respectively suggesting whether a pair of artifacts should be assigned together or not. A new consensus-based constrained clustering framework was proposed. It utilizes consensus clustering in two phases. First, rather than selecting constraints randomly, the framework chooses only constraints with moderate scores in the co-association matrix; the constraints generated in this way tend to be more informative. Second, from a certain set of constraints, the framework generates multiple constrained clusterings and then combines them to produce the final constrained clustering. Experiment 8.A and 8.B evaluated this framework on eight data sets with numbers of constraints up to 1000. It was observed that the results of consensus clustering of constrained partitions improved significantly over the results without consensus clustering. The bounded constraint generations were also found to outperform random generation in most data sets. It should be noted this framework was found to be more effective in improving requirements clusters than in improving normal document clusters. In requirements engineering applications in which lots of user interactions can be expected, this framework can further improve the quality of clusters.

Incremental requirements clustering

In some applications such as automated requirements forum organization, since requirements tend to arrive incrementally, the clustering should be able to quickly and effectively cluster the new artifacts in an incremental fashion. Moreover, in the requirements forum, stakeholders are likely to value stability of clusters, so that their own requirements are not continually moved from one cluster to another. A new incremental clustering framework was proposed to address these needs. While utilizing a number of techniques introduced in previous chapters, the framework has a core strategy that adds seed-preserving into the basic SPK algorithm. In particular, each time a new cluster is created, all existing centroids are retained except the one whose corresponding cluster shows the least cohesion, then two new centroids are generated from the split of the least cohesive cluster using 2-means SPK. When clusters are periodically regenerated, the existing seeds are used to seed the new clustering algorithm.

Experiment 9.A validated this framework by measuring three attributes: quality of clustering measured using NMI, stability between two consecutive clusterings measured using the Jaccard metric, and time consumption in seconds. Compared with entirely re-clustering, the seed-preserving approach produced clustering of comparable quality, achieved far better stability, and spent less time. Experiment 9.B evaluated the same attributes when user constraints were provided at each increment. Three constraint generation methods were explored, including random generation, bounded generation, and cluster-based generation, which selected only

constraints for which both feature requests had been placed in the same cluster. This simulated the more realistic case for the requirements domain, in which feedback was elicited from users during their real-time interactions with the discussion threads. Incorporating constraints into the re-clustering process slightly increased the runtime of each algorithm. The bounded and cluster-based constraints led to greater quality improvements than the random approach. The results of stability were interesting. The bounded constraint selection technique outperformed the random approach when the standard clustering algorithm was used, but interestingly failed to outperform the random approach when the seed-preserving algorithm was used, whereas the cluster-based method returned mixed results.

Experiment 9.C was designed to evaluate whether user tags might help increase the quality of the clusterings on data set STUDENT. A GUI was developed that displayed feature requests within their clusters, and five DePaul MS students and faculty were asked to tag the needs in ten randomly assigned clusters. Terms from each tag were then added to the text of the feature request, and subsequent incremental reclusterings were based on the combined text of the feature request plus the user contributed tags. While no pronouced change in stability and speed were observed, the use of tags led to a significant improvement of 12.4% in cluster quality as measured by the NMI metric.

Application scenarios of requirement clustering

Finally, two applications of requirements clustering were described to illustrate the automation support clustering can potentially provide. A technique called the Aspect Clustering Engine (ACE) was proposed to detect candidate early aspects. The core step of ACE is to use clustering to discover dominant and recessive themes. In automated tracing, clustering can be used to organize returned links so as to enhance the understanding of the links and then to facilitate the evaluation of the links. A small analysis was performed to evaluate the effectiveness of trace-base clustering. Five business requirements were traced to the lower level requirements of the Ice Breaker System, and the number of decision points was significantly reduced in each of the queries when clustering is used to organize the results.

In summary, the research presented in this dissertation investigates state-of-art methods to cluster software requirements and suggests new ways to improve cluster quality. A number of areas are expected to be further addressed by future research. First, substantial work is needed in building more requirements data and their answer clustering sets. This will facilitate a statistically strict comparison between requirements data sets and well-know document data sets (such as TREC) and then the development of better requirements clustering framework. Second, the work on feedback collection and integration described in Chapter 10 can have several extensions. For instance, in addition to providing tags for content enrichment, the set of feedback can also be used to generate pair-wise constraints, and these constraints can be utilized by the constrained clustering framework that is proposed in Chapter 9. Third, with the majority of this dissertation about how to cluster, numerous potential research can be conducted to evaluate the use of clustering across various requirements related applications. As an example, in aspects

detection, given that requirements are inherently multi-topical, soft clustering by probabilistic models such as PLSA or LDA could complement the results from using crisp clustering, leading to a more comprehensive set of candidate aspects. The results from this dissertation, which provide insights into clustering software requirements with a clear appreciation of requirements' characteristics, are expected to make clustering play an increasingly important role in providing effective automation support to RE activities.

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.

[Basu02] Basu, S., Banerjee, A., and Mooney, R. J. 2002. Semi-supervised Clustering by Seeding. In Proceedings of the Nineteenth international Conference on Machine Learning (July 08 - 12, 2002). C. Sammut and A. G. Hoffmann, Eds. Morgan Kaufmann Publishers, San Francisco, CA, 27-34.

[Basu04a] Basu, S., Banerjee, A., and Mooney, R. J. 2004. Active semisupervision for pairwise constrained clustering. In Proc. of the 4th SIAM International Conference on Data Mining (Orlando, FL, 2004), pp. 333-344.

[Basu04b] Basu, S., Bilenko, M., and Mooney, R. J. 2004. A probabilistic framework for semi-supervised clustering. In Proceedings of the Tenth ACM SIGKDD international Conference on Knowledge Discovery and Data Mining (Seattle, WA, USA, August 22 - 25, 2004). KDD '04. ACM, New York, NY, 59-68.

[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.

[Bilenko06] Bilenko, M., Basu, S., and Mooney, R. J. 2004. Integrating constraints and metric learning in semi-supervised clustering. In Proceedings of the Twenty-First international Conference on Machine Learning (Banff, Alberta, Canada, July 04 - 08, 2004). ICML '04, vol. 69. ACM, New York, NY, 11.

[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.

[Castro08] 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.

[Cleland-Huang09] Cleland-Huang, J., Dumitru, H., Duan, C., and Castro-Herrera, C. 2009. Automated Support for Managing Feature Requests in Open Forums. To appear in Communications of the ACM, 2009.

[Cohn03] Cohn, D., Caruana R., and McCallum, A. 2003. Semi-supervised clustering with user feedback. Technical Report TR2003-1892, Cornell University, 2003.

[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.

[Davidson05] Davidson, I. and Ravi, S.S. 2005. Hierarchical clustering with constraints: theory and practice. In: Proc. 9th European principles and practice of KDD (PKDD'05). Porto, Portugal pp 59-70.

[Davidson06a] Davidson I. and Ravi S.S. 2006. Identifying and Generating Easy Sets of Constraints For Clustering, 21st AAAI Conference, 2006.

[Davidson06b] Davidson I., Wagstaff, K., and Basu, S. 2006. Measuring Constraint-Set Utility for Partitional Clustering Algorithms, In Proceeding of ECML/PKDD, 2006.

[Davidson07] Davidson, I. and Ravi, S. S. 2007. Intractability and clustering with constraints. In Proceedings of the 24th international Conference on Machine Learning (Corvalis, Oregon, June 20 - 24, 2007). Z. Ghahramani, Ed. ICML '07, vol. 227. ACM, New York, NY, 201-208.

[Decker07] Decker, B., Ras, E., Rech, J., Jaubert, P., and Rieth, M. 2007. Wiki-Based Stakeholder Participation in Requirements Engineering. *IEEE Software*. 24, 2 (Mar. 2007), 28-35.

[Deer90] 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.

[Duan08] Duan, C., Cleland-Huang, J., and Mobasher, B. 2008. A consensus based approach to constrained clustering of software requirements. In *Proceeding of the 17th ACM Conference on information and Knowledge Management* (Napa Valley, California, USA, October 26 - 30, 2008). CIKM '08. ACM, New York, NY, 1073-1082.

[Duan09] Duan, C., Dumitru, H., Cleland-Huang J., and Mobasher, B. 2009. Incremental Requirement Clustering in a Dynamic Environment: Balancing Stability and Change. Submitted to KDD ’09.

[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.

[Greene07] Greene, D. 2007. Constraint Selection by Committee: An Ensemble Approach to Identifying Informative Constraints for Semi-Supervised Clustering. In Proc. 18th European Conf. on Machine Learning (ECML'07), 140-151, Springer.

[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).

[Hook01] Hook, S., Prata, F., and Schladow, G. 2001. Progress Report. http://eospso.gsfc.nasa.gov/ftp_docs/validation

[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.

[Laddad03] Laddad, R. 2003. *AspectJ in Action: Practical Aspect-Oriented Programming*. Manning Publications.

[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).

[Li07] Li, T., Ding C., and Jordan M.I. 2007. Solving Consensus and Semi-supervised Clustering Problems Using Nonnegative Matrix Factorization. ICDM 2007: 577-582.

[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.

[McLach00] 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.

[Niu08] Niu, N. and Easterbrook, S. 2008. On-Demand Cluster Analysis for Product Line Functional Requirements. In *Proceedings of the 2008 12th international Software Product Line Conference* (September 08 - 12, 2008). International Conference on Software Product Line. IEEE Computer Society, Washington, DC, 87-96.

[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.

[Rosenr04] Rosenhainer, 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.

[Standish94] Standish Group International. The Chaos Report; 1994, www.standishgroup.com<http://www.standishgroup.com>.

[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.

[Tang07] Tang, W., Xiong, H., Zhong, S., and Wu, J. 2007. Enhancing semi-supervised clustering: a feature projection perspective. In Proceedings of the 13th ACM SIGKDD international Conference on Knowledge Discovery and Data Mining (San Jose, California, USA, August 12 - 15, 2007). KDD '07. ACM, New York, NY, 707-716.

[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.

[Wagstaff01] Wagstaff, K., Cardie, C., Rogers, S., and Schrödl, S. 2001. Constrained K-means Clustering with Background Knowledge. In Proceedings of the Eighteenth international Conference on Machine Learning (June 28 - July 01, 2001). C. E. Brodley and A. P. Danyluk, Eds. Morgan Kaufmann Publishers, San Francisco, CA, 577-584.

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

[Wnuk09] Wnuk, K., Regnell B., and Schrewelius C. 2009. Architecting and Coordinating Thousands of Requirements – An Industrial Case Study. The 15th International Working Conference on Requirements Engineering: Foundation for Software Quality

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

[Xing03] Xing, E. P., Ng, A. Y., Jordan, M. I., and Russell, S. (2003) Distance metric learning, with application to clustering with side-information. *Advances in Neural Information Processing Systems* 15.

[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).

[Yan06] Yan B. and Domeniconi, C. 2006. Subspace Metric Ensembles for Semi-supervised Clustering of High Dimensional Data. In Proceedings of the *17th European Conference on Machine Learning*, Berlin, Germany, September 18-22, 2006.

[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.

[Zipf49] Zipf, G. K. 1949. *Human Behavior and the Principle of Least-Effort*. Addison Wesley.