A New Co-similarity Measure : Application to Text Mining and Bioinformatics

Syed Fawad Hussain

THÈSE

pour l'obtention du titre de

DOCTEUR de L'UNIVERSITÉ DE GRENOBLE

Spécialité : Informatique

Préparée au laboratoire TIMC-IMAG UMR 5525
École Doctorale : Mathématique, Science et Technologie de l'Information, Informatique

Présentée et soutenue publiquement le 28 septembre 2010 par

Syed Fawad Hussain

Titre: **A New Co-similarity Measure : Application to Text Mining and Bioinformatics**

Une Nouvelle Mesure de Co-Similarité : Applications aux Données Textuelles et Génomique

Directeurs de Thèse :

Mirta B. Gordon et **Gilles Bisson**

Composition du jury:

M	Eric GAUSSIER	(LIG, Grenoble)	Président
M	Marco SAERENS	(IAG, Louvain)	Rapporteur
M	Antoine CORNUEJOLS	(AgroParisTech, Paris)	Rapporteur
M	Juan-Manuel TORRES MORENO	(LIA, Avignon)	Examinateur
Mme	Mirta B. GORDON	(TIMC, Grenoble)	Directrice de thèse
M	Gilles BISSON	(TIMC, Grenoble)	Directeur de thèse

Acknowledgement

I would like to thank all people who have helped and encouraged me during my doctoral study.

My PhD years wouldn't have been nearly as fruitful without my advisors. I, therefore, especially want to thank Gilles Bisson and Mirta Gordon for their patience, their guidance and their motivation during my thesis. They have been really generous with their time and willing to help at all times. I consider myself fortunate to have worked with them and hope that we could continue to collaborate in future.

I would also like to express my gratitude to all the members of the jury for evaluating my PhD work. I would like to thank Prof. Eric Gaussier for accepting to be the president of my jury. I equally wish to thank Marco Saerens and Antoine Cornuejols, for accepting to review my manuscript. There comments and suggestions have been really beneficial in improving the overall presentation of this manuscript. I also thank Juan-Manuel Torress Moreno for accepting to be on my jury.

Many thanks to all my colleagues and members of the AMA team for providing such a wonderful atmosphere and for making me feel at home away from home. Their friendship, support and encouragement have made my years at AMA a really memorable occasion. I also wish to thank my friends at Grenoble for the good times we spent together.

Finally, I would like to thank my family for all their support in my life. I love you all.

Syed Fawad Hussain

Abstract

Clustering is the unsupervised classification of patterns (observations, data items, or feature vectors) into groups (clusters). The clustering problem has been addressed in many contexts and there exist a multitude of different clustering algorithms for different settings. As datasets become larger and more varied, adaptations of existing algorithms are required to maintain the quality of clusters. In this regard, high-dimensional data poses some problems for traditional clustering algorithms known as '*the curse of dimensionality*'.

This thesis proposes a *co-similarity* based algorithm that is based on the concept of distributional semantics using higher-order co-occurrences, which are extracted from the given data. As opposed to co-clustering, where both instance and feature sets are hard clustered, *co-similarity* may be defined as a more '*soft*' approach. The output of the algorithm is two similarity matrices – one for the objects and one for their features. Each of these similarity matrices exploits the similarity of the other, thereby implicitly taking advantage of a co-clustering style approach. Hence, with our method, it becomes possible to use any classical clustering method (*k*-means, Hierarchical clustering …) to co-cluster data.

We explore two applications of our co-similarity measure. In the case of text mining, document similarity is calculated based on word similarity, which in turn is calculated on the basis of document similarity. In this way, not only do we capture the similarity between documents coming from their common words but also the similarity coming from words that are not directly shared by the two documents but that can be considered to be similar. The second application is on gene expression datasets and is an example of co-clustering. We use our proposed method to extract gene clusters that show similar expression levels under a given condition from several cancer datasets (colon cancer, lung cancer, etc).

The approach can also be extended to incorporate prior knowledge from a training dataset for the task of text categorization. Prior category labels coming from data in the training set can be used to influence similarity measures between features (words) to better classify incoming test datasets among the different categories. Thus, the same framework can be used for both clustering and categorization task depending on the amount of prior information available.

Keywords: Clustering, co-clustering, supervised learning, text mining, co-similarity, structural similarity

Résumé

La classification de données (apprentissage non-supervisé) vise à regrouper un ensemble d'observations sous la forme de classes homogènes et contrastées. Lorsque les données sont caractérisées par un grands nombre de propriétés, il devient nécessaire d'adapter les méthodes classique, notamment au niveau des métriques, afin de maintenir des classes pertinentes ; ce phénomène est connu sous le nom de "malédiction de la dimension".

Dans cette thèse nous proposons une mesure de co-similarité basée sur la notion de co-occurrences d'ordre supérieur, directement extraites à partir des données. Dans le cas de l'analyse de texte, par exemple, les similarités entre documents sont calculées en prenant en compte les similarités entre mots, qui simultanément prennent en compte les similarité entre documents. Par cette approche circulaire, nous mettons en correspondance des documents sans mots communs mais juste des mots similaires. Cette approche s'effectue sans nécessiter de thesaurus externe.

En outre, notre méthode peut également être étendu pour tirer partie de connaissances "a priori" pour réaliser des tâches de catégorisation de textes : l'étiquette des documents est utilisée pour influencer les mesures de similarité entre les mots afin de classer de nouvelles données. Ainsi, le même cadre conceptuel, exprimable en terme de la théorie des graphes, peut être utilisé à la fois pour les tâches de classification et de catégorisation en fonction de la quantité d'information initiale.

Nos résultats montrent une amélioration significative de la précision, par rapport à l'état de l'art, pour le co-clustering et la catégorisation sur les jeux de données qui ont été testés.

Mots-Clés: Co-similarité, co-classification, systeme d’apprentissage, fouille de texts, expression génique, co-clustering.

Table of Contents

List of Tables

List of Figures

Introduction Générale

Avec l'avènement de l'age de l'information et l'Internet, en particulier au cours des deux dernières décennies, notre capacité à générer, enregistrer et stocker des données multi-dimensionnelles et non structurées est en augmentation rapide. D'énormes volumes de données sont maintenant disponibles pour les chercheurs dans différents domaines (publications de recherche et bibliothèques en ligne), les biologistes (par exemple: micro puces ou données géniques), en sociologie (données de réseaux sociaux), etc. Pour faire face à cette augmentation du volume de données les tâches d'extraction d'informations pertinentes et d'acquisition des connaissances - telles que la recherche de motifs ou de relations cachées - ont connu un grand essor. Ainsi, le Data Mining (ou fouille de données) se réfère au domaine qui s'intéresse à l'étude formelle de ces problèmes et il englobe un large éventail de techniques dans les domaines des mathématiques, des statistiques et de l'apprentissage machine. Un défi majeur dans l'exploration de ces données est que les informations que nous souhaitons extraire ne sont généralement peu ou pas connues à l'avance.

Les techniques de Data Mining sont donc utiles pour découvrir des modèles et des relations inattendues à partir de données et elles ont reçu une large attention dans divers domaines scientifiques et commerciaux. Ces techniques peuvent être classées en deux grandes façons : l'apprentissage non supervisé ou clustering et l'apprentissage supervisé ou la classification. Le clustering est une approche visant à découvrir et identifier des « motifs » (comme des ensembles caractéristiques, des éléments typiques de données) à partir de la création de groupes (clusters) . Elle vise donc à organiser une collection d'observations ou d'objets en fonction de la similarité qui existe entre eux (généralement les données sont représentées comme un vecteur de données, ou comme un point dans un espace multidimensionnel). Intuitivement, les caractéristiques qui apparaissent au sein d'un cluster valide sont plus semblables les unes aux autres qu'elless ne le sont pour de celles appartenant à un autre cluster.. En pratique, un algorithme de clustering est basée sur une fonction objectif qui essaie à minimiser la distance intra-cluster entre ses objets et de maximiser la distance inter-cluster (on cherche des classes homogènes et contrastées). La FIGURE 1 ci-dessous illustre un regroupement d'objets de données (également appelé les individus, les cas ou les lignes d'un tableau de données). Ici les données peuvent être facilement divisées en 3 groupes différents en fonction de leur mesure de distance.

Il faut noter que cette tâche de clustering a été pratiquée par les humains depuis des milliers d'années (Willett, 1981; Kural, Robertson, et Jones, 1999), et qu'elle a été en grande partie automatisé dans les dernières décennies en raison de l'avancement de la technologie informatique. On l'utilise maintenant dans une grande variété de domaines, tels que l'astronomie, la physique, la médecine, la biologie, l'archéologie, la géologie, la géographie, la psychologie,

et de le commerce. De nombreux domaines de recherche différents ont contribué à proposer de nouvelles approches (par exemple, la reconnaissance des formes, les statistiques et l'analyse des données, la théorie de l'information, l'apprentissage machine, la bioinformatique et). Dans de nombreux cas, l'objectif du clustering est d'obtenir une meilleure compréhension des données (par exemple, en faisant apparaître une structure "naturelle" sous-jacente des données qui se traduit par l'apparition un regroupement significatif). Dans d'autres cas, la constitution de classes n'est qu'une première étape à des objectifs diverss, telles que l'indexation ou la compression de données. Dans tous les cas le clustering est une démarche « exploratoire » visant à condenser et mieux comprendre les données.

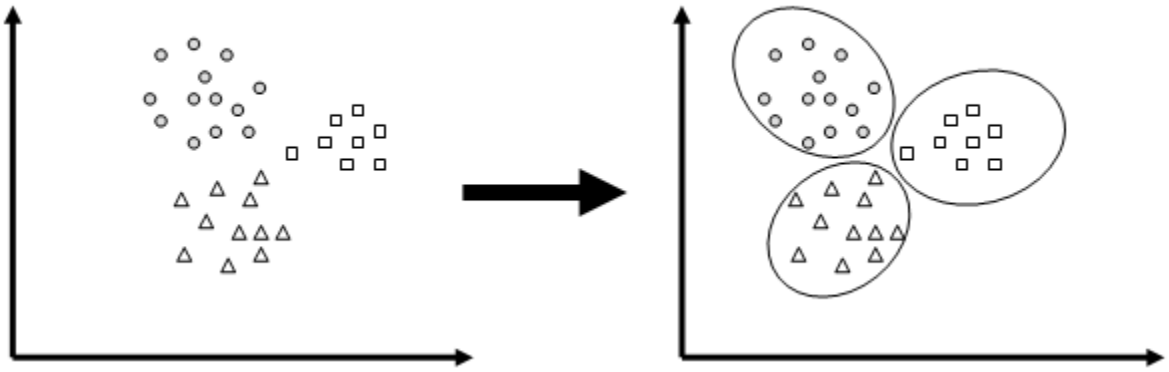


FIGURE 1 Clustering objects

Les données enregistrées dans une base de données, ou comme une matrice de données, peuvent être analysées sous deux angles, à savoir (i) en se référant aux enregistrements (ou lignes), ou (ii), en se référenat aux attributs (ou colonnes). Dans les deux cas, l' objectif est la découverte des motifs récurrents cachés. Considérons une base de données constituées par un ensemble d'objets où les lignes sont les documents et les colonnes sont les mots contenus dans ces documents. L'aproche classique en clustering repose sur le calcul d'une mesure de similarité entre les paires de documents puis sur l'utilisation de ces valeurs pour regrouper les documents afin de former des groupes de documents (partition) ou des groupes de clusters (hiérarchies) qui sont liés les uns aux autres. Ceci est connu comme le « one-way clustering » ou simplement clustering, où l'on essaye de trouver des modèles basés sur une vision globale des données.

Considérons maintenant une base de données contenant des données comme des micro-puces (micro-arrays), où les lignes sont les gènes et les colonnes sont les conditions expérimentales dans lesquelles le degré d'expression des gènes a été mesurés, par exemple selon que le tissu contient une tumeur cancéreuse ou non. Prenons le cas d'un gène qui contrôle la division cellulaire, il est clair qu'un tel gène va s'exprimer fortement dans le cas d'un cancers mais ce degré d'expression n'est pas forcément spécifique d'un cancer ni même d'ailleurs d'un cancer en général. Si l'on analyse ce gène de manière générale, indépendamment des conditions expérimentales on risque donc de ne trouver aucune information pertinente. Dans ce cas, un point de vue plus local, qui considère les valeurs d'expression de gènes visà vis d'un sous-ensemble des conditions, peut être plus approprié.

Par conséquent, travailler simultanément sur les expressions (ou documents) et les gènes (ou vatriables) est

fondamental. Ceci est connu comme la tâche de « biclustering » ou « co-clustering ».

Par ailleurs, en apprentissage supervisé (ou classification), des connaissances a priori sur les données sont utilisées pour entraîner un « classifieur » pour faire des prédiction sur de nouvelles données. Une application classique de l'apprentissage supervisé est la catégorisation de textes. La catégorisation de textes vise à affecter de nouveaux documents texte, non étiquetés, dans l'une des catégories prédéfinies en fonction de leur contenu. Si les textes sont des articles de journaux, les catégories pourraient être des thèmes tels: économie, politique, science, et ainsi de suite. Cette tâche a d'autres applications telles que la classification automatique d'email et de catégorisation de pages Web. Ces applications sont de plus en plus importantes dans la société d'aujourd'hui qui est axée sur la circulation de l'information. La catégorisation de textes est aussi appelée la classification de texte, la catégorisation de documents ou de classification des documents.

Plusieurs approches de catégorisation de documents – comme celles basées sur des mesures de similarité comme le cosinus ou des distances Euclidiennes, etc. - supposent implicitement que les textes de la même catégorie ont une distribution de mots identiques. De telles mesures de similarité sont fondés sur le nombre de mots qui sont partagés entre les deux documents. Considérons les 4 phrases suivantes :

S1: There are many types of *ocean* *waves*.
S2: A swell is a formation of long surface *waves* in the *sea*.
S3: Swelling is usually seen around a broken *knee* or *ankle*.
S4: The *patella bone* is also known as the *knee* cap and articulates with the *femur*.

Il est évident que les deux premières phrases concernent l'océanographie, tandis que la troisième et la quatrième phrase traitent de l'anatomie. Les mots qui pourraient être associés à ces sujets sont en italique alors que les mots qui sont partagés entre les documents sont soulignés. Ici, il pourrait donc être difficile de déterminer quelles phrases forment un cluster. La phrase deux (S2), par exemple, partage un mot chacun avec S1 et S3 respectivement.[1]

Les humains peuvent facilement identifier quelles deux phrases doivent être regroupées, en partie parce que la phrase S1 contient le mot océan et la phrase S2 contient le mot mer. De même, S3 contient les mots du *knee* et *ankle*, et S4 contient les mots *patella bone* et *femur*. Si nous pouvions attribuer des valeurs de similarité entre ces mots, alors nous serions capable de définir quelles paires de phrases devraient être regroupées, même si elles ne partagent pas les mêmes mots.

Si l'on dispose d'assez de documents, il devient possibke d'attribuer automatiquement les valeurs de similarité entre les mots et entre les documents qui contiennent des tels mots. Nous appelerons l'étude de telle similitude la mesure de co-similarité. L'objectif fondamental de cette thèse est donc de proposer une approche fondée sur la co-similarité au co-clustering afind' améliorer le one-way-clustering et le biclustering.

[1] Dans le contexte de fouille de texte, seulment les mots de base sont pris en compte. Par exemple, les mots *swell* et *swelling* sont considérés comme le même mot. En plus, on ne prend pas en compte des mots fréquents tels que *is*, *a*, *to*…

Contribution Principle de la Thèse

Les principales contributions de cette thèse sont les suivantes:

- Nous vous proposons une nouveau mesure de (co-) similarité qui peut être utilisée avec n'importe quel algorithme de clustering comme la Classification Ascendante Hiérarchique, les k-means, les classification par densité, etc.
    - Notre nouvelle mesure de similarité, baptisé χ-Sim, exploite la nature duale de la relation qui existe dans de nombreux ensembles de données entre les objets et les variables. Par exemple, entre les documents et les mots, dans le contexte du Text Mining. Nous mesurons la similarité entre les documents en tenant compte de la similarité entre les mots qui apparaissent dans ces documents et réciproquement.
    - Nos résultats expérimentaux montrent que l'utilisation de cette approche basée sur la co-similarité donne de meilleurs résultats (en termes de précision) que de nombreux algorithmes de clustering et co-clustering existants.
- Nous fournissons une explication en terme de théorie des graphes du fonctionnement de notre algorithme. Le fonctionnement de notre algorithme est basé sur l'idée de produire des valeurs de similarité utilisant les co-occurrences pondérées d'ordre supérieur dans un gaphe bi-partite. Tutefois lorsque l'on considère ces graphes, il faut prendre soin d'éviter les nœuds redondants, qui ont déjà été visités précédemment. Nous proposons une technique pour explorer les chemins (pondérés) jusqu' à l'ordre 3 tout en évitant ces chemins redondants.
- Nous élargissons notre mesure de (co-)similarité afin d'exploiter des connaissances préalables sur les étiquettes des classes pour faire de l''apprentissage supervisé. Nous proposons une double stratégies complémentaires pour exploiter les étiquettes des catégories dans l'ensemble d'apprentissage :
    - maximisation de la similarité des documents appartenant à la même catégorie.
    - minimisation de la similarité des documents ayant des étiquettes différentes.
- Nous fournissons des résultats expérimentaux obtenus à partir de plusieurs bases de données de textes pour évaluer le comportement de notre algorithme et pour pouvoir effectuer une comparaison avec plusieurs autres algorithmes classiques de la littérature.
- Enfin , nous proposons des adaptations de notre algorithme pour qu'il soit applicable à données d'expression génique pour accomplir la tâche de biclustering. Nous testons notre algorithme sur plusieurs bases de données de cancer.

Structure de la Thèse

Le reste de cette thèse est organisé en 5 chapitres.

- Chapter 2 chapitre 2 donne un aperçu des travaux antérieurs dans les domaines du clustering et du co-clustering, principalement dans le contexte de l'exploration de texte..

- Chapter 2 chapitre 3 fournit les détails de notre algorithme et son contexte théorique. Ce chapitre traite également des inconvénients potentiels de l'algorithme et examine les moyens par lesquels ils peuvent être minimisés. Enfin, une extension de l'algorithme à la tâche de l'apprentissage supervisée est également proposées dans ce chapitre.
- Chapter 2 chapitre 4 fournit tous les résultats expérimentaux à la fois pour le clustering de documents et pour leur catégorisation, en faisant l'analyse et la comparaison avec d'autres techniques existantes.
- Chapter 2 chapitre 5 fournit une introduction au domaine de la bioinformatique et les problèmes rencontrés lors du biclustering sur les données d'expression génétique, comme la transformation des données, la sélection génétique, etc. Ensuite nous étudions l'application de l'algorithme proposé pour le biclustering de plusieurs ensembles de données concernant l'expression des gènes et issus du domaine de la bio-informatique.
- Chapter 2 chapitre 6 fournit une conclusion de notre travail et examine les perspectives d'avenir.

Chapter 1
Introduction

1.1. Setting the Scene

With the advent of the information age and the internet, particularly during the last couple of decades, our capacity to generate, record and store multi-dimensional and apparently un-structured data is increasing rapidly. Huge volumes of data are now available to researchers in different fields (research publications and online libraries), biologists (for example micro array and genetic data), sociology (social networks data), ecologists (sensor network data), etc. With the increase in data and its availability, however, comes the task of mining relevant *information* and *knowledge* — such as finding patterns or hidden relationships within the data. Data mining refers to the formal study of these problems and encompasses a broad range of techniques from the fields of mathematics, statistics, and machine learning. A key challenge in data mining is that the information we wish to extract is usually not known or only partially known beforehand.

Data mining techniques are useful for discovering unsuspected patterns and relationships from data and have received wide attention in various scientific as well as commercial fields. These techniques can be categorized in two broad ways – unsupervised learning or clustering and supervised learning or classification.

Clustering is the unsupervised identification of patterns (such as observations, data items, or feature vectors) into groups (clusters). It refers to the organization of a collection of patterns (usually represented as a vector of measurements, or as a point in a multidimensional space) into clusters based on their similarity values. Intuitively, patterns within a valid cluster are more similar to each other than they are to a pattern belonging to a different cluster. The goal of a clustering algorithm thus is to partition the data in such a way that objects that are *similar* in some sense are grouped in the same cluster. Typically, a clustering algorithm has an objective function that tries to minimize the intra-cluster distance between its objects and maximize the inter-cluster distance. FIGURE 1.1 below shows a grouping of data objects (also called observations, individuals, cases, or data rows). The data can easily be divided into 3 different clusters based on their distance measure.

Clustering is a task that has been practiced by humans for thousands of years (Willett 1981; Kural, Robertson, and Jones 1999), which has been mostly automated in the last few decades due to the advancements in computing technology**.** Cluster analysis has been used in a large variety of fields, such as astronomy, physics, medicine, biology, archaeology, geology, geography, psychology, and marketing. Many different research areas contributed new approaches (i.e., pattern recognition, statistics, information retrieval, machine learning, bioinformatics, and data mining). In some cases, the goal of cluster analysis is a better understanding of the data (e.g., learning the "natural" structure of data which should be reflected by a meaningful clustering). In other cases, cluster analysis is merely a first step for different purposes, such as indexing or data compression.

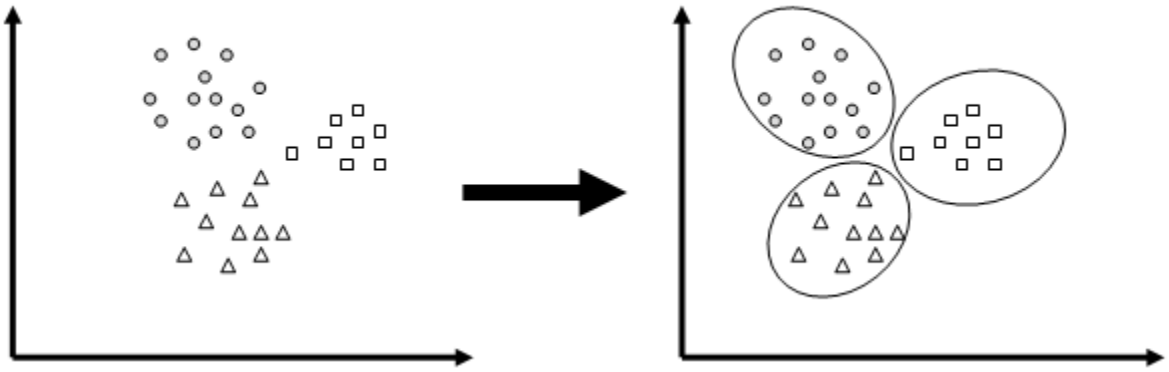


FIGURE 1.1 Clustering objects

The data recorded in a database, such as a data matrix, has traditionally been analyzed from two perspectives, namely (i) with reference to the records (or rows); or (ii), with reference to the attributes (or columns). In both cases, our objective is the discovery of hidden, yet interesting, patterns by comparing either the rows or the columns of the data matrix depending on which clustering we are interested in. Consider a database, for example, a set of documents whose rows are the documents and whose columns are the words contained in those documents. Typically, we calculate the similarity measure between pairs of documents and use these values to group together documents to form groups or clusters of documents which are related together. This is known as one-way clustering or simply clustering and it tries to find patterns based on a global view of the data.

Consider now a database containing micro-array data, where the rows are genes and the columns are the conditions under which the intensity of the genes were measured, such as whether the tissue contains a cancerous tumor or not. Typically, a gene will express itself with higher intensity under certain conditions such as a gene that controls specific functions of cell division. Such a gene may have a high intensity in tissues having a cancerous tumor or even having only a specific type of cancer. In this case, a global view of the gene's expression values might not form any interesting pattern(s). Rather, our interest lies in identifying sets of genes that are over (or under) expressed under certain cancerous conditions. Thus a local view, which considers the expression values of genes

under a subset of the conditions, might be more desirable. Therefore a clustering of both the records and attributes is essential. This is known as *biclustering* or *co-clustering*.

Classification on the other hand is a technique where prior knowledge about the records is used to train a classifier for new records. One such application of supervised learning is text categorization. Text categorization is the task in which new, unlabelled, text documents are categorized into one of predefined categories based on their contents. If the texts are newspaper articles, categories could be, for example, economics, politics, science, and so on. This task has various applications such as automatic email classification and web-page categorization. Those applications are becoming increasingly important in today's information-oriented society. Text categorization is also called text classification, document categorization or document classification.

Several text clustering approaches – such as those based on similarity measures like Cosine, Euclidean, etc – implicitly assumes that texts in the same category have an identical distribution of words. Such similarity measures are based number of documents that are shared between two documents. Consider the following 4 sentences

S1: There are many types of *ocean* *waves*.
S2: A swell is a formation of long surface *waves* in the *sea*.
S3: Swelling is usually seen around a broken *knee* or *ankle*.
S4: The *patella bone* is also known as the *knee* cap and articulates with the *femur*.

It is evident that the first two sentences concern *oceanography*, while the third and forth sentences are taking about *anatomy*. Words that could be associated with these topics are italicized while words that are shared between documents are underlined. In this way, it could be hard to determine which sentences form a cluster. Sentence two (S2), for instance, share one word each with S1 and S3 respectively[2].

Humans could easily identify which two sentences should be grouped together, in part because sentence S1 contains the word *ocean* and sentence S2 contains the word *sea*. Similarly, S3 contains the words *knee* and *ankle* while S4 contains the words *patella*, *bone* and *femur*. If we could assign similarity values between these words, then we would be capable of defining which sentence pair should be grouped together even if they do not share the same words.

Given enough documents, we could automatically assign similarity values between words and between the documents that contain such words. We refer to such a similarity as *co-similarity*. The fundamental aim of this thesis is to provide a *co-similarity* based approach to co-clustering for improving one-way clustering and for biclustering.

1.2. Main Contribution of this Thesis

The main contributions of this thesis are as follows:

[2] In text mining, words are usually compared in their base forms, for example 'swell' and 'swelling' will be considered as the same word; while common words such as 'is', 'a', etc are usually not taken into account. For details see section 4.4

- We propose a new (co-)similarity measure that could be used with any clustering algorithm such as Agglomerative Hierarchical Clustering, *k*-means, etc.
    - Our new similarity measure, christened χ-Sim, exploits the dual nature of relationship that exists in many datasets for example, between documents and words, in text mining. We measure the similarity between documents taking into account the similarity between words that occur in these documents.
    - Our experimental results show that using this co-similarity based approach yields better results (in terms of accuracy) than many clustering and co-clustering algorithms.
- We provide a graph-theoretical explanation of the working of our proposed algorithm. The working of our algorithm is rooted in the concept of generating similarity values based on exploring weighted high-order paths in a bi-partite graph. When considering any such walks, care must be taken to avoid self-repeating nodes which has been previously visited. We provide a technique to explore such (weighted) paths of up to order-3 while avoiding such redundant paths.
- We extend our (co-)similarity measure to exploit available prior knowledge for supervised learning. We propose a two pronged strategy to exploit category labels in the training set to influence similarity learning such that
    - documents in the same category tend to have a higher similarity value.
    - documents in different categories tend to have a lower similarity value.
- We provide experimental results on various text datasets to evaluate the behavior of our proposed algorithm and provide a comparison with several other algorithms.

We apply our proposed algorithm on gene expression data to perform the task of biclustering. We test our algorithm on several cancer datasets to bicluster gene and conditions.

1.3. Structure of the Thesis

The rest of this thesis is organized in 5 chapters.

- Chapter 2 provides an overview of the related work which has been previously done in the area of clustering and co-clustering, mostly related to text mining.
- Chapter 3 provides the details of our proposed algorithm and its theoretical background. This chapter also discusses potential drawbacks of the proposed algorithm and discusses ways in which these could be reduced. Finally, an extension of the proposed algorithm to the supervised task is also given in this chapter.
- Chapter 4 provides all the experimental results for both document clustering and categorization, with analysis and comparison with other techniques.

- Chapter 5 provides an introduction to the bioinformatics domain and the problems of biclustering gene expression data such as data transformation, gene selection, etc and the application of the proposed algorithm on biclustering on several gene expression datasets coming from the bioinformatics domain.
- Chapter 6 provides a conclusion of our work and discusses the future perspectives.

1.4. List of Publications

International Conferences

- Hussain F, Grimal C., Bisson G.: "An improved Co-Similarity Measure for Document Clustering", *9th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 12-14th Dec. 2010, Washington, United States. [To Appear]
- Hussain F. and Bisson G.: "Text Categorization using Word Similarities Based on Higher Order Co-Occurrences", Society for Industrial and Applied Mathematics *International Conference on Data Mining (SDM 2010)*, Columbus, Ohio, April 29-May 1, 2010.
- Bisson G., Hussain F.: "χ-sim: A new similarity measure for the co-clustering task", *7th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 11-13th Dec. 2008, San Diego, United States.

National Conferences

- Hussain S. F. and Bisson G. : "Une approche générique pour la classification supervisée et non-supervisée de documents", *Conference Francophone pour l'Apprentissage Automatique (CAP),* Clermont-Ferrand, France, 17-19 May, 2010.
- Bisson G., Hussain F. : "Co-classification : méthode et validation ", *In 11ème Conférence Francophone sur l'apprentissage automatique (CAp 2009)*, Plate-forme AFIA, Hammamet, Tunisie, 26-29 Mai, 2009. Editions Cépaduès.

Chapter 2
Related Work

The objective of clustering is to partition an unstructured set of objects into clusters (groups). From a machine learning point of view, clustering represents an unsupervised learning technique to search for hidden patterns (clusters) and the outcome represents the data concept. Clustering algorithms can usually be described as hierarchical or partitioning methods. Typically, these methods revolve around the concept of similarity/distance measures between objects such that objects grouped together in a cluster are more similar is some way than objects grouped in different clusters. In certain domains with high-dimensional data, however, the notion of distance is somewhat lost because most objects are only represented by a small subset of these dimensions. Several approaches such as the Latent Semantic Analysis, project such data onto a lower dimension space before trying to determine similarity values between objects. In the last decade, attempts have also been made to simultaneously partition the set of samples and their attributes into co-clusters. The resulting (co-) clusters signify a relationship between a subset of samples in a subset of attributes. Such algorithms employ certain additional information – such as entropy, about the data to enhance the clustering. The task of clustering then can be seen as a compact representation of the data that tries to preserve this additional/auxiliary information as much as possible. In this chapter, we review some of the techniques that have been used for clustering and co-clustering, particularly in the domain of text mining and bio-informatics.

2.1. Data Representation

As introduced in Chapter 1, clustering is the unsupervised classification of patterns (such as observations, data items, or feature vectors) into groups (clusters). We first give a formal definition of the task of clustering below. We will focus on the task of document clustering (unless specifically mentioned otherwise) throughout the rest of this

chapter since this will be a principal application area for our proposed similarity measure.

Let us assume that X is the document set to be clustered, X = $\{x_1, x_2, \ldots, x_m\}$. Each document x_i is an *n*-dimensional vector, where each dimension typically corresponds to an indexing term. A clustering $\hat{\mathrm{X}}$ of *X* into *k* (distinct) sets can be defined as $\hat{\mathrm{X}} = \{\hat{x}_1, \hat{x}_2, ..., \hat{x}_k\}$ so that the following conditions are satisfied:

1. Each cluster $\hat{x}_i$ contains at least one document: $\hat{x}_i \neq \phi$, i=1, …, k
2. The union of all clusters is the set X: $\cup_{i=1}^{k} \hat{x}_i = \mathrm{X}$
3. No two clusters have documents in common: $\hat{x}_i \cap \hat{x}_j = \phi, i \neq j; i, j = 1..k$

The third condition guarantees that clusters do not overlap. In this thesis, we will only deal with single labeled clustering (and categorization), also known as *hard clustering* (as opposed to soft clustering where an object can be part of different clusters with varying levels of confidence).

Several techniques have been used in the literature to index documents, mostly borrowed from Information Retrieval (Manning, Raghavan, and Schütze 2008; Baeza-Yates and Ribeiro-Neto 1999; Jurafsky, J. H Martin, and Kehler 2000; Manning and Schütze 1999). The most commonly used one is the Vector Space Model (and its graphical representation as a bi-partite graph).

2.1.1. Vector Space Model

The Vector Space Model (VSM) was proposed by (G. Salton, Wong, and C. S. Yang 1975). Given a set X of *m* documents, let Y be the set of terms in the document collection whose size is give by *n* (*n* is the size of the corpus dictionary). The dimensions of the vector space usually represent the set of all different words that can be found throughout a document collection, i.e., the vocabulary set. Also let y_i denote a term in the set of terms used to index the set of documents with i=1…n. In the VSM model, for each term y_i there exists a vector $\mathbf{y}_i$ in the vector space that represents it. It then considers the set of all term vectors $\{\mathbf{y}_i\}$ ($1 \leq i \leq n$) to be the generating set of the vector space, thus the space basis. A document vector $\mathbf{x}_i$ is given by

$$\mathbf{x}_i = (y_{i1}, y_{i2}, ..., y_{in}) \tag{2.1}$$

If each $\mathbf{x}_i$ (for i = 1... m) denotes a document vector of the collection, then there exists a linear combination of the term vectors $\{\mathbf{y}_i\}$ which represents each $\mathbf{x}_i$ in the vector space. Once a vector has been defined for each document in the corpus, they can be collected in a document-by-term matrix[3] **A** in which each row represents a document and each column represents a word (or term) in the corpus. The resulting document-by-term matrix **A** whose element A_{ij} denotes the occurrence of a word j in document i as shown below

[3] In the literature, term-by-document matrix is also used especially in the domain of IR. The term-by-document matrix corresponding to **A** is simply the transpose of **A**.

(2.2)
$$\mathbf{A} = \begin{pmatrix} A_{11} & A_{12} & \dots & A_{1n} \\ A_{21} & A_{22} & \dots & A_{2n} \\ \dots & \dots & \dots & \dots \\ A_{m1} & A_{m2} & \dots & A_{mn} \end{pmatrix}$$

When referring to documents and words in the matrix **A**, we will use $\mathbf{a}_i$ to denote a row vector (corresponding to document x_i) and $\mathbf{a}^j$ to denote a column vector (corresponding to word y_j).

The second aspect of the vector space model deals with weighting the terms. The techniques used are borrowed from Information Retrieval domain, where text documents are represented as a set of index terms that are weighted according to their importance for a particular document and the corpus (G. Salton and Lesk 1968; G Salton 1971; Sebastiani 2002; Y. Yang and X. Liu 1999). Various techniques have been proposed in the literature to weight terms, for example a binary-valued vector weighting scheme indicating the presence or absence of the term in the document; or real-valued, indicating the importance of the term in the document. There are multiple approaches for how real-valued weights can be computed such as *tf-idf* (G. Salton and C. Buckley 1988), *term distribution* (Lertnattee and Theeramunkong 2004), or simply the number of occurrence of a word in a document, etc.

The TF-IDF Model The most popular approach used for term weighting in the domain of text clustering, text categorization and information retrieval is the *tf-idf* scheme (G. Salton and C. Buckley 1988). In this approach, the entry A_{ij} is defined as a product of the Term Frequency (*TF*) and the Inverse Document Frequency (*IDF*) given by

(2.3)
$$A_{ij} = TF_{ij} * IDF_j$$

where

(2.4)
$$TF_{ij} = \frac{A_{ij}}{\sum_{k=1..n} A_{ik}} \quad \text{and}$$

(2.5)
$$IDF_j = \log\left(\frac{|\mathrm{X}|}{|\mathbf{a}^j|}\right)$$

TF is a function of the number of occurrences of the particular word in the document divided by the number of words in the entire document. This means that the importance of a term in a document is proportional to the number of times that the term appears in the document. Similarly, IDF ensures that the importance of the term is inversely proportional to the number of times that the term appears in the entire collection. The logarithm in *IDF* is used as it has been shown that a word in a collection of documents follows the Zipf's law (Zipf 1949). Without the logarithmic scale, the IDF function would grow too fast with decreasing number of occurrences of a word in the corpus (Manning and Schütze 1999). If only *IDF* was used to weight terms in a document x_i, then rare terms would dominate a geometric similarity computation: a term that occurs only once in the document collection has a maximum *IDF* value. The product of *TF* and *IDF* ensures that both rare and frequent terms do not over-influence the

similarity measure.

The Boolean Model In the Boolean document model, the representation $\mathbf{x}_i$ of a document $x_i \in X$ is a vector whose j^{th} component indicates if y_j occurs in x_i. An equivalent of the Boolean model is the set model, where a document is represented as a set in which the elements are the document's terms. The Boolean model is defined as

$$\forall i = 1..m, j = 1..n \quad A_{ij} = \begin{cases} 1 & if\ word\ j\ occurs\ in\ document\ i \\ 0 & otherwise \end{cases} \tag{2.6}$$

The Number of occurrences In this case, the entities in A_{ij} corresponds to the number of time word j occurs in document i. As opposed to the Boolean case, by using the number of occurrences, one could give more importance to words that occur multiple times in a document. For instance, the word "χ-Sim" [4] is much more relevant to this thesis (and occurs multiple times throughout the document), than say some other work that simply cites this work. Using the Boolean model, one may loose this importance of different words in documents by assigning just the presence or absence of the word.

Several other weighting schemes have been proposed in the literature, for example, a probabilistic method based on the assumption that informative words are only found in a subset of the documents. The term weighting is then done based on the *divergence from the randomness* theory based on Ponte and Crofts language model (Ponte and Croft 1998). Similarly, Amati and Rijsbergen (Amati and Van Rijsbergen 2002) have proposed to first normalize a document based on the document length and propose an alternative *normalized idf*.

2.1.2. The Bipartite Graph Model

Several clustering algorithms (discussed later in the chapter) adopt a graph-theoretical view of grouping objects. First, let us understand what a graph is and how it is represented. The word *graph* has at least two meanings:

- A graph could refer to the plot of a mathematical function, or
- A collection of points and set of lines connecting some subsets of these points.

We are concerned with the second definition. We define a graph as a collection of *entities* and their *relationships*. The *entities* are usually represented as the vertices (or nodes) of the graph, and their *relationships* as edges (or links). In the case of text, we have two sets of objects – documents and words – which are represented as a *bipartite graph*. A *bipartite graph* is a special graph which can be partitioned into two set of vertices X and Y, such that all edges link a vertex from X to a vertex in Y and no edges are present that link two vertices of the same partition. Equivalently, a bipartite graph is a graph in which no odd-length cycles exist.

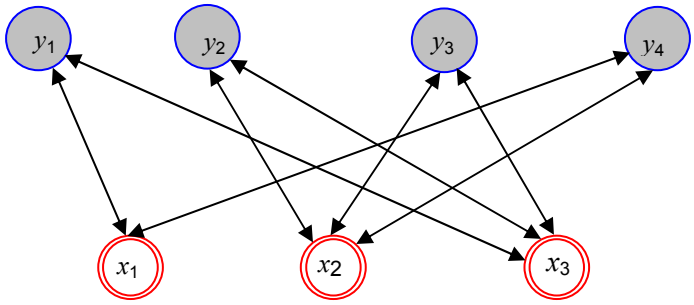
A bipartite graph is usually defined by G = (X,Y,E) where X=$\{x_1,\ldots x_m\}$ and Y=$\{y_1,\ldots,y_n\}$ are two sets of vertices and E is a set of edges $\{(x_i,y_j);\ x_i \in X,\ y_j \in Y\}$. In our case, the two sets of vertices X and Y represent the document set and word set respectively. An edge signifies an association between a document and a word. By putting positive weights on the edges, we can identify the strength of this association. It is straightforward to identify

[4] χ-Sim is the name we give to a new co-similarity measure that we will propose in chapter 3.

a clear relationship between a bipartite graph and a matrix. This is shown in FIGURE 2.1.

	y_1	y_2	y_3	y_4
x_1	1	0	0	1
x_2	0	1	1	1
x_3	1	1	1	0

(a)



(b)

FIGURE 2.1 Representing 3 documents (x_1-x_3) and 4 words (y_1-y_4) using (a) a vector space model, and (b) a bipartite graph model

The two sets of vertices correspond to the rows and columns of the matrix **A** while the edges correspond to the entries A_{ij}. Similarly, bipartite graph can be either weighted with a weight assigned to each edge or simply binary, indicating the presence of absence of an edge as seen previously for matrix entries A_{ij}. Bipartite graphs are usually represented using an adjacency matrix. Given a *document-by-term* matrix **A**, the corresponding adjacency matrix **M** for the bipartite graph G is given by

$$\mathbf{M} = \begin{bmatrix} 0 & \mathbf{A} \\ \mathbf{A}^{\mathrm{T}} & 0 \end{bmatrix} \tag{2.7}$$

where we have ordered the vertices such that the first m vertices index the documents while the last n index the words and $\mathbf{A}^{\mathrm{T}}$ denotes the transpose of the matrix **A**. The dimensions of **M** are $m+n$ by $m+n$. As shall be seen in the later part of this chapter (section 2.5), such representation is sometimes useful when using graph theoretical methods to cluster documents and words simultaneously that use several matrix manipulation techniques.

2.2. Document Similarity Measures

Recall that the objective of clustering is to partition an unstructured set of objects into clusters (groups). Most clustering algorithms use a similarity (or dissimilarity) measure between the objects. A large number of such measures that quantify the resemblance between objects have been proposed in the literature. We introduce here a formal definition of a distance measure. Using the VSM presented above where X is the set of documents to be clustered, then a distance measure is a function *dist*: $\mathrm{X} \times \mathrm{X} \rightarrow \Re$, where $\Re$ is the set of nonnegative real numbers. Such a function *dist*, in general, satisfies the following axioms:

1. $dist\,(x_i,x_i) = 0$ (reflexivity)

2. $dist\ (x_i,x_j) \geq 0$ (non-negativity)
3. $dist\ (x_i,x_j) = dist\ (x_j,x_i)$ (symmetry)
4. $dist\ (x_i,x_j) + dist\ (x_j,x_k) \geq dist\ (x_i,x_k)$ (triangular inequality)
5. $dist\ (x_i,x_j) \in [0,1]$ (normalization)

A function that satisfies the first four axioms is called a metric (Duda, Hart, and Stork 2001). The axiom 5 is usually satisfied by a large number of distance measures but is not a necessity for being a similarity or distance metric. A similarity measure *Sim(,)* can similarly be defined as a function *Sim*: $X \times X \rightarrow \Re$. The use of either types of similarity or distance measures is usually problem dependent.

Sneath and Sokal (Sneath and Sokal 1973) categorize such measures in four main classes: association (or *similarity)*, *dissimilarity*, *probabilistic*, and *correlation* coefficients. The first two usually belong to more generic sets of *geometric* measures and to the *set theory*. Document clustering has mostly utilized geometric measures which we discuss below. Probabilistic measures have mostly been used in Information Retrieval where probability measures are used to rank documents according to a given query, while correlation coefficients have mostly been employed in the domain of bioinformatics to group genes based on their profiles over certain conditions.

Geometric Measures

The category of measures is usually used with the vector model under Euclidean geometry and has been popularly employed when comparing continuous data. Given the index vectors for two documents, it is possible to compute the similarity coefficient between them, $Sim(\mathbf{a}_i,\mathbf{a}_j)$ which represents the degree of similarity in the corresponding terms and term weights. For instances, the Minkowski distance measure is defined as

$$Dist(\mathbf{a}_i,\mathbf{a}_j) = \sum_{k=1}^{n}\left(\left|A_{ik} - A_{jk}\right|^{1/p}\right)^{p} \tag{2.8}$$

Perhaps the most popular distance metric is the *Euclidean distance* which is the special case of the Minkowski distance with p=2. The Euclidean distance works well when the dataset have compact or isolated clusters (A. K Jain, Duin, and Mao 2000). The Euclidean distance has a drawback in that it is not scale invariant. This implies that the largest-valued features tend to dominate over other features. Another potential drawback arises when comparing two unequal sized objects, such as two document vectors of different lengths. As a result, the most common measure of similarity used in text mining is the Cosine measure or the Cosine of the angle between the two given documents:

$$Sim(\mathbf{a}_i,\mathbf{a}_j) = \frac{\mathbf{a}_i^{T}\mathbf{a}_j}{\left\|\mathbf{a}_i\right\|\left\|\mathbf{a}_j\right\|} = \frac{1}{\left\|\mathbf{a}_i\right\|\left\|\mathbf{a}_j\right\|}\sum_{k=1}^{n} A_{ik}\cdot A_{jk} \tag{2.9}$$

This is intuitively appealing: two texts of different sizes covering the same topics are similar in content when the measure of angle between their vectors is small. The Cosine measure is not affected by the size of the documents i.e. it is scale invariant. It merely considers the proportions of the words in the document (the normalized vectors). Thus, documents are regarded as equal when they use similar proportion of words irrespective of their lengths. It

should be noted that if the vectors x_i and x_j are normalized to the unit norm ($||x_i||=||x_j||=1$), then the Euclidean measures becomes scale invariant and is complementary to the dot product since $(||x_i-x_j||)^2 = (x_i-x_j)^T(x_i-x_j) = ||x_i||^2 + ||x_j||^2 - 2\cos(x_i,x_j)$ where $(x_i-x_j)^T$ is the transpose of (x_i-x_j).

Strehl and Gosh (Strehl 2002) proposed a similarity measure $Sim\left(x_i, x_j\right) = e^{-\left\|x_i - x_j\right\|^2}$ which is based on the Euclidean distance and has been used in *k*-means clustering. A summary of most commonly used measures can be found in (R. Xu and Wunsch 2005).

Several other similarity (and distance) measures have been proposed in the literature. A discussion about some of the classical measures can also be found, for example, in (A. K. Jain, Murty, and Flynn 1999; Diday and Simon 1976; Ichino and Yaguchi 1994) among others. Similarity measures are central to most clustering algorithms. We conclude this section with a brief note on the utilization of these measures. Given the large number of measures available, the question naturally arises of the choice of the most appropriate one(s) for the purpose of document clustering. We first consider the normalized and non-normalized measures. Van Rijsbergen (Van Rijsbergen 1979) advised against the use of any measure that is not normalized by the length of the document vectors under comparison. (Willett 1983) performed experiments on different measures to determine inter document similarity using 4 similarity measures (inner product, Tanimoto coefficient, Cosine coefficient, and the overlap coefficient) and five term weighting schemes. Experimental results confirmed the poor effectiveness of non-normalized measures. Similarly, (Griffiths, Robinson, and Willett 1984) compared the Hamming distance and Dice coefficient and found the former (which is not normalized) inferior to the later.

In most such comparison analysis, especially using the hierarchical clustering algorithm, the Cosine similarity measure was reported to perform better. (Kirriemuir and Willett 1995) applied hierarchical clustering using Cosine, Jaccard and normalized Euclidian distance measures to the output of database search. Their reported results also suggest that the Cosine and Jaccard coefficient were found to be superior in their study.

2.3. Clustering Methods

The clustering problem has been addressed in many contexts and there exist a multitude of different clustering algorithms for different settings (A. K. Jain et al. 1999; Berkhin 2006; Buhmann 2003; R. Xu and Wunsch 2005). This reflects its broad appeal and usefulness as an important step in data analysis. As pointed out in Backer and Jain (Backer and A. K. Jain 2009),

"*In cluster analysis, a group of objects is split up into a number of more or less homogeneous subgroups on the basis of an often subjectively chosen measure of similarity (i.e., chosen subjectively based on its ability to create "interesting" clusters), such that the similarity between objects within a subgroup is larger than the similarity between objects belonging to other subgroups*" (Backer and A. K. Jain 2009).

However, the idea of what an ideal clustering result should look like varies between applications and might be even different between users. Clustering algorithms may be divided into groups on several grounds, see for example (A. K. Jain et al. 1999). A rough but widely used such division is to classify clustering algorithms as *hierarchical*

algorithms that produce a hierarchy of clusters, and *partitioning* algorithms that give a flat partition of the set (Everitt, Landau, and Leese 2001; A. K. Jain et al. 1999; R. Xu and Wunsch 2005; Berkhin 2006). Hierarchical algorithms form new clusters using previously established ones while partitioning algorithms determine all clusters at once. Clustering methods usually employ a similarity matrix. They do not care how this matrix is calculated, since they perform the clustering process assuming that the matrix has been calculated in some way.

A vast amount of algorithms and their variants have been proposed in the literature. In this section, we review some of the widely used hierarchical and partition algorithms and stress a common drawback of such algorithms when applied high-dimensional data. A survey of all the algorithms is not the goal of this thesis and no such attempt has been made. Several attempts have been made previously by different authors to provide a survey of various popular clustering algorithms and wherever possible, we provide a reference for readers who wish to read more details about these algorithms. Instead, we will provide a brief introduction to popularly used clustering algorithms in this section to focus more on alternate approaches that have been proposed for high-dimensional data in the next section.

2.3.1. Hierarchical Clustering

Hierarchical clustering methods result in tree-like classifications in which small clusters of objects (i.e. documents) that are found to be strongly similar to each other are nested within larger clusters that contain less similar objects. Hierarchical methods are divided into two broad categories, *agglomerative* and *divisive* (A. K. Jain et al. 1999). An agglomerative hierarchical strategy proceeds through a series of (|X|-1) merges, for a collection of |X| documents, and results in clustering building from the bottom to the top of the structure. In a divisive strategy, on the other hand, a single initial clustering is subdivided into progressively smaller groups of documents (Van Rijsbergen 1979). A hierarchical algorithm yields a *dendrogram* representing the nested grouping of patterns and similarity levels at which groupings change. The *dendrogram* can be cut at different levels to achieve several clusterings as shown in FIGURE 2.2.

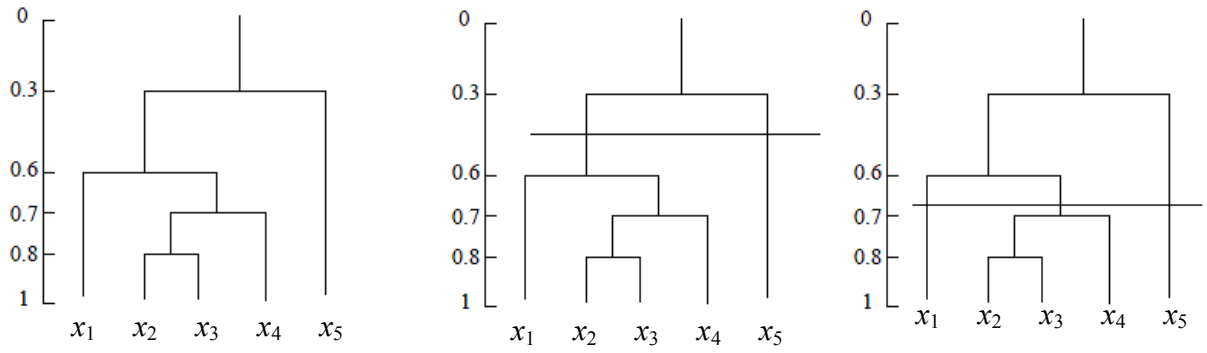


FIGURE 2.2 A dendrogram showing different clustering of 5 documents $x_1..x_5$

We will concentrate here on the agglomerative approaches as they have been popularly used in clustering, especially document clustering. Hierarchic agglomerative methods usually follow the following generic procedure (Murtagh 1983):

1. Determine all inter-document similarities,
2. Form a new cluster from the two closest (most similar) objects or clusters,
3. Redefine the similarities between the new cluster and all other objects or clusters, leaving all other similarities unchanged,
4. Repeat steps 2 and 3 until all objects are in one cluster.

The various agglomerative methods proposed usually differ on the way that they implement step 3 of the above procedure. At each step t of the clustering process, the size of the similarity matrix **S** (which initially is X by X) becomes (X-t) by (X-t). The matrix $\mathbf{S}^t$ of step t of the process is derived from the matrix $\mathbf{S}^{t-1}$ by deleting the two rows and columns that correspond to the newly merged documents (or clusters), and by adding a new row and column that contain the new similarities between the newly formed cluster and all unaffected (from step t of the process) documents or clusters.

Table 2-1 Summary of various linkage algorithms

Linkage	Method	Comments	Reference
Single	Combine clusters with closest distance between any members	- Easy to implement -Susceptible to outliers - might form loose clusters	(Cormack 1971)
Complete	Combine clusters with the smaller farthest distance. Sometimes called farthest neighbor or maximum algorithm.	- Sensitive to outliers - Good when there are tightly bound, small clusters - May suffer from space dilution when the size of clusters grow. Not suitable for elongated clusters.	(Lance and Williams 1967)
Average	Combines clusters based on their average distance measure between all pairs of objects	- Sensitive to outliers with might affect the average - When a large cluster is merged with a small one, the properties of the smaller one are usually lost - Could be both weighted or un-weighted	(Murtagh 1983)
Centroid	Similar to Average but builds a prototype (centroid) to represent each cluster.	- Same as with Average linkage - Need to re-calculate cluster centroid at each iteration.	(Murtagh 1983)
Ward's	Combine clusters that result in minimum sum of squared distances	- Considers overall cluster objects before merging - May form spherical clusters, thus making it non-suitable for highly skewed clusters	(Ward Jr 1963)

Several linkages are used to combine clusters at each iteration, such as the single linkage, complete linkage, Ward's linkage, etc. The methods have been summarized in Table 2-1 while a graphical interpretation of some of

the linkage methods appear in Figure 2.3. In the single linkage algorithm (Figure 2.3a), a new cluster is formed in step by merging two clusters that have the closest distance from any member of the first cluster to any member of the second cluster. In the complete linkage clustering algorithm (Figure 2.3b), two clusters are merged by considering the farthest distance from any member of the first cluster to any member of the second cluster. The average linkage merge clusters with the lowest average distance between elements of the clusters (Figure 2.3c), while the Centroid linkage first builds a "representative" or "centroid" (for example by taking the mean or the median) element for each cluster and then merges clusters whose centroids are nearest (Figure 2.3d).

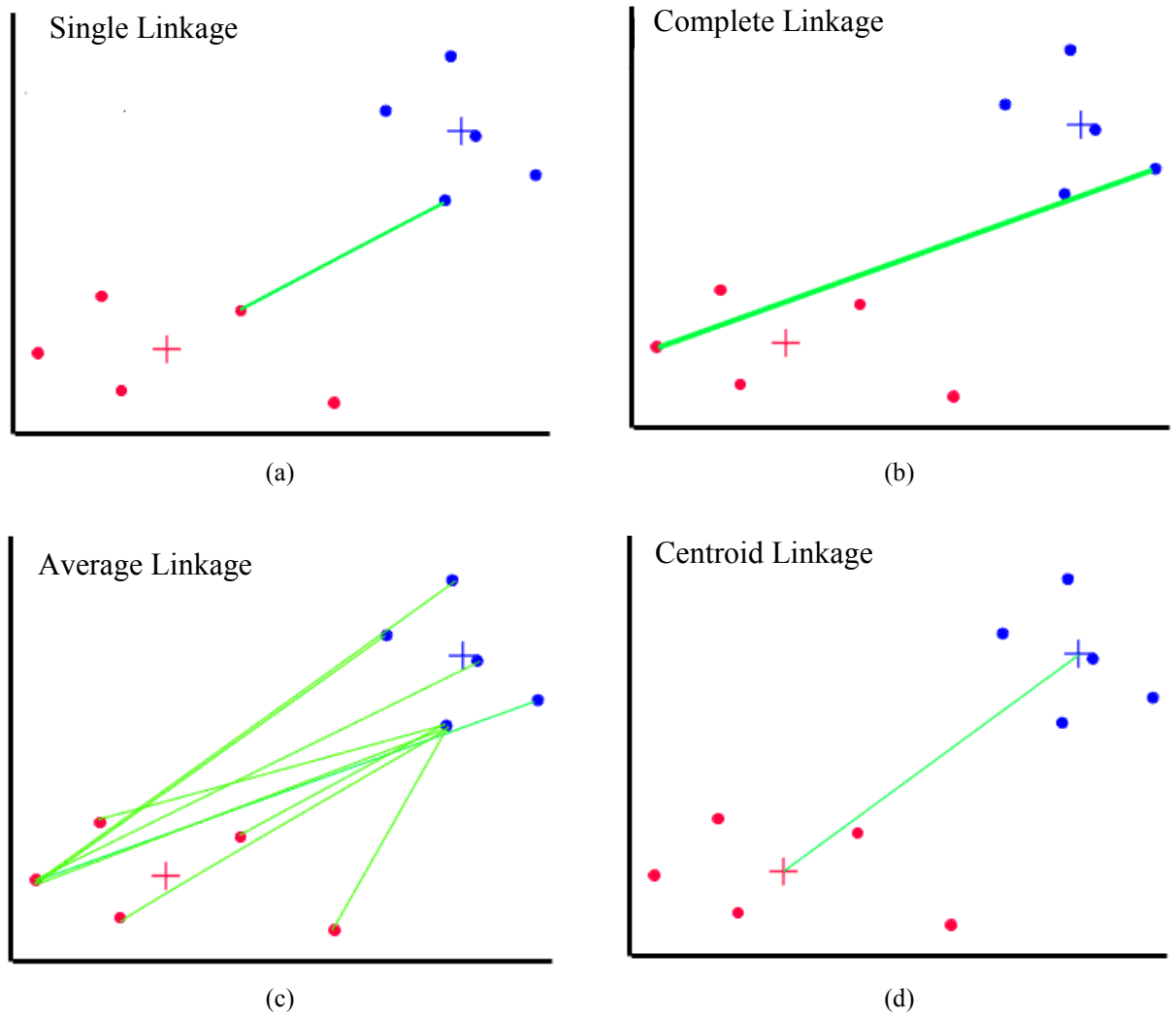


Figure 2.3 Various Linkage algorithms

The Ward's (Ward Jr 1963) linkage merges two clusters so as to minimize an objective function that reflects the investigator's interest in the particular problem. Ward illustrated this method with an error sum of squares objective function, and Wishart (Wishart 1969) showed how Ward's method can be implemented through updating a matrix of squared Euclidean distances between cluster centroids. Lance and Williams (Lance and Williams 1967) proposed a

special recurrence formula that is used in the computation of many agglomerative hierarchical clustering algorithms. Their formula (see Appendix I for details) provides updating rules by expressing a linkage metric between the union of the two clusters and the third cluster in terms of underlying components. Thus, manipulation using similarity (or distance) measures become computationally feasible. In fact, under certain conditions such as reducibility condition (Olson 1995) linkage based algorithms have a complexity of $O(m^2)$. A survey of linkage metrics can be found in (Murtagh 1983; Day and Edelsbrunner 1984).

Other Hierarchical Algorithms

Several other hierarchical clustering algorithms have been proposed in the literature. For example, the agglomerative hierarchical clustering algorithm CURE (Clustering Using Representatives) was proposed by Guha et al. (Guha, Rastogi, and Shim 2001). Instead of using a single centroid to represent a cluster like in the linkage based-approaches presented above, CURE chooses a constant number of cluster points to represent each cluster. The similarity is then based on the closest pair of representing points between the two clusters. As a result, CURE is able to find clusters of arbitrary shapes and sizes since each cluster is represented via multiple points and does not suffer from the issues mentioned in the earlier linkage methods. The choice of these "representative" points however is usually not trivial.

Another similar algorithm is CHAMELEON (Karypis, E. H Han, and V. Kumar 1999) which tries to find clusters in the data using a two-phase approach. First, it generates a k nearest neighbors (k-NN) graph (containing links between a point and its k-NN) and uses a graph-partitioning algorithm to cluster the data objects into a large number of clusters. Similarly, BIRCH (Balanced Iterative Reducing Clustering using Hierarchies) is another approach proposed by Zhang et al. (T. Zhang, Ramakrishnan, and Livny 1996) that builds a height-balanced data structure called a CF-Tree while scanning the data. It is based on two parameters the Branching factor β and the threshold T which refers to the maximum diameter of a cluster. At each stage when a new data object is found, the tree is traversed by choosing the nearest node at each level and the object placed on a leaf node if it satisfies the threshold condition. BIRCH has an advantage in that it can create a clustering tree with one scan (though subsequent scans are usually needed to improve the result). However, since it is based on a threshold condition T, it may not work well when clusters are not spherical. Additionally, the clustering also depends on the order of the input and the same data may result in different clustering on subsequent runs.

Some of the advantages of using hierarchical algorithms include the following

- provides a hierarchy of the clusters
- ease of visualization and interpretation
- flexibility in the granularity of clusters (by using a cut through the dendrogram).

But at the same time, Hierarchical clustering also suffers from a few disadvantages, such as

- they are usually computationally expensive for large datasets
- most hierarchical algorithms cannot improve a previous assignment since they do not revisit a node after assigning an object.

2.3.2. Partition Based Clustering

In contrast to the hierarchical clustering algorithms, partitions based clustering methods starts by clustering the whole dataset into *k* partitions. Once the data objects are initially partitioned into *k*-clusters, several heuristics are then used to refine the clustering based on some objective function. Hence, unlike the hierarchical clustering algorithms, partitioned based algorithms are relatively fast and objects that have been assigned to a partition are revisit and may be re-assigned to iteratively improve the clustering.

Most partition based clustering algorithms starts with a definition of an objective function that is to be iteratively optimized. Linkage metrics such as pair-wise similarity (or dissimilarity) measures provide such a natural function that can be used to measure the intra- and inter-class similarities. Using an iterative approach to optimize such a clustering would be computationally prohibitive, hence a "representative" or "prototype" of a cluster is chosen instead. Therefore, instead of comparing each object against every other object, we only compare it against a set of *k* prototype objects of each cluster. Perhaps the most widely used partition based algorithm is the *k*-means using the squared error criterion (MacQueen 1966). The *k*-means algorithm tries to partition the objects in the dataset into *k* subsets such that all points in a given subset are closest to a given center or prototype. It starts by randomly selecting a set of *k* instances as "representatives" of clusters and assigning the rest of the objects based on some distance criteria (such as sum of squared errors). A new centroid (such as the mean) is then recalculated for each cluster to be used as *k* prototype points and the process is repeated until a termination criterion (usually a threshold or number of iterations) is met. If we represent $\mathbf{m}_i$ as the mean of each cluster $\hat{x}_i$, then the sum of squared error is given by

$$SSE = \sum_{i \in k} \sum_{j \in \hat{x}_i} \left\| \mathbf{a}_j - \mathbf{m}_i \right\|^2 \tag{2.10}$$

The *k*-means algorithm has its advantages in that it is simple to implement and fast. But it suffers from a number of drawbacks, for example it is sensitive to outliers which can effect the mean value. The *k*-medoids method is a variation of the *k*-means where the cluster is represented by one of its points. This has a few advantages over the *k*-means as mediods have an embedded resistance against outliers (hence are less affected). Various propositions have been made to select suitable initial partitions and using different distance measures, see for example (Berkhin 2006). However, *k*-means based algorithm suffers from number drawbacks such as

1. There is no universal method to identify the number of partitions beforehand,
2 The iteratively optimal procedure of *k*-means does not guarantee convergences towards a global optimum,
3. The *k*-means algorithm remains sensitive to outliers, and
4. The clusters usually have a spherical shape.

A detailed description of some of these limitations and different variations proposed to improve the clustering solution can be found in (R. Xu and Wunsch 2005). Similarly, recent work (Z. Zhang, J. Zhang, and H. Xue 2008; Arthur and Vassilvitskii 2007) have also improved the k-means algorithm by proposing new methods to choose the

initial seeds that have resulted in improvements in both clustering accuracy (section 4.2) and instability.

Different version of the *k*-medoids have been proposed such as Partitioning Around Medoids (PAM) by (Kaufman and Rousseeuw 1990), in which the guiding principle is the effect on an objective function by combining the relocation of points between clusters and re-nominating the points as potential medoids. This of course has an effect on the cost of the algorithm since different options must be explored. Similarly, the CLARA (Clustering Large Applications) algorithm (Kaufman and Rousseeuw 1990) and its enhancement to spatial databases, known as CLARANS (Ng and J. Han 1994), are based on the idea of choosing multiple samples to represent a prototype and each is subjected to PAM. The dataset is assigned to the best system of medoids based on the objective function criteria. Other variants that allow to use splitting and merging resulting clusters based on the variance or SSE have also been proposed. Several other enhancements to the *k*-medoids algorithm have been proposed in the literature. A survey of such techniques can be found in (Berkhin 2006).

2.3.3. Other Clustering Methods

Density Based Techniques. This set of algorithms is based on the idea that an open set in the Euclidean space can be divided into a set of its connected components (Berkhin 2006). They consider a similarity graph and try to find partitions of highly connected sub-graphs. Their core idea is based on the notions of density, connectivity and boundary. Ideally, they can find the number *k* of sub-graphs automatically and can find clusters of arbitrary shape and size. Representative algorithms of this category include DBSCAN (Ester et al. 1996), OPTICS (Ankerst et al. 1999), etc. Their running time normally depends on a variety of factors but is usually in the magnitude of $O(m^2)$ where *m* is the number of samples. A limitation of density based clustering algorithms is that they may not be able to separate two dense sub-clusters in a larger cluster and their results are often difficult to interpret. An introduction to density based methods can be found in (J. Han and Kamber 2006) and a survey of some of these methods can be found in (Berkhin 2006; R. Xu and Wunsch 2005) under the heading *Large scale datasets*.

Graph Theory Based Techniques. These algorithms are based on the concepts and properties of graphs theory where each object is represented as a node and similarity between objects is denoted by a weighted edge between objects usually based upon a threshold value. Graph theoretic approaches consist of both hierarchical and partition based approaches. Perhaps the best known partition based graph theoretic clustering algorithm is the Minimum Spanning Tree (MST) (Zahn 1971). It works by constructing a MST on the data and then removing the largest lengths to generate clusters. Hierarchical approaches such as Single Linkage and Complete linkage can also be considered as Graph-based approaches. Clusters generated by using single linkage are sub-graphs of minimum spanning tree of the data (Gower and G. J. S. Ross 1969) and are the connected components in the graph (Gotlieb and S. Kumar 1968). Similarly, complete linkage generated clusters are maximal complete sub-graphs. Other graph theoretic approaches for overlapping clusters have also been developed (Ozawa 1985). A brief survey of graph-theoretical approaches is covered in (A. K. Jain et al. 1999), and a more detailed survey can be found in (R. Xu and Wunsch 2005).

Probability Estimation Based Techniques. This class of algorithms considers the data to be in $\Re^n$ as a sample independently drawn from a mixture of models of several probability distributions of k m-dimensional density functions δ_1, .., δ_k with different parameters. Each sample is then considered to have been derived form a weighted combination of these mixture models with weights $w_1,\ldots w_k$ and $\Sigma w_{i=1..k}=1$. The objective of the algorithm in this category is thus to estimate the set of parameters for each density function δ_i, because each cluster is thought to be generated by such a function. The probability that a sample was generated by such a function is then computed based on density estimates and the number of data points associated with such a cluster. A representative algorithm of this category is the Expectation-Maximization (EM) methods. A survey of methods in this category can be found in (R. Xu and Wunsch 2005; A. K. Jain et al. 1999; Achlioptas and McSherry 2005).

Several other methods have been proposed in the clustering literature such as Grid based methods, Fuzzy clustering methods, Evolutionary algorithms, Search based methods etc. An excellent survey of clustering data mining techniques can be found in (A. K. Jain et al. 1999; Berkhin 2006; M. W Berry 2007; M. S Yang 1993). A more recent survey of clustering algorithms and their applications can be found in (R. Xu and Wunsch 2005). In the next subsection, we consider a limitation of these algorithms when applied to high-dimensional dataset known as the curse of dimensionality.

2.3.4. Curse of Dimensionality

As datasets become larger and more varied, adaptations of existing algorithms are required to maintain the quality of cluster as well as efficiency. However, high-dimensional data poses some problems for traditional clustering algorithms. Berkhin (Berkhin 2006) identifies two major problems for traditional clustering algorithms – the presence of *irrelevant attributes* and the *curse of dimensionality*.

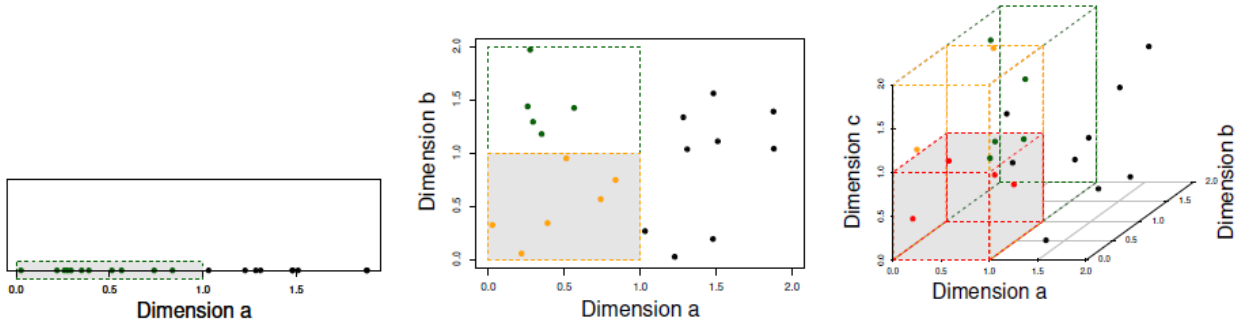
The problem of *irrelevant features* is as follows: data groups (clusters) are typically characterized by a small group of attributes. The other features are considered as *irrelevant* attributes since they do not necessarily help in the clustering process. Moreover, such attributes could confuse clustering algorithms by hiding the real clusters by heavily influencing similarity measures. It is common in high dimensional data that any pair of instances share some features and clustering algorithms tend to get lost since "searching for clusters where there are no clusters is a hopeless enterprise" (Berkhin 2006). While irrelevant features may also occur in low dimensional data, their likelihood and strength increases substantially with increase in dimensionality.

The *curse of dimensionality* is a well known problem in high dimension data. Originally coined by Bellman (Bellman 1961), the term is used to refer to the exponential growth of hyper-volume with a linear increase in dimensionality. There are two phenomenons

1. the density of points decrease exponentially with the increase in dimensions, and
2. the distance between two given points, chosen randomly, tends to be increasingly similar

Mathematically, the concept of *nearness* increasingly becomes "meaningless" (Beyer et al. 1999). In particular, for a given object, the distance between the farthest and nearest data point tends to decrease with increase in dimensionality (Beyer et al. 1999). Traditional distance measures such as *Euclidean* or *Cosine* do not always make

much sense in this case. This is further illustrated in FIGURE 2.4 below. The dataset consists of 20 points randomly placed between 0 and 2 in each of three dimensions. FIGURE 2.4 (a) shows the data projected onto one axis. The points are close together with about half of them in a one unit sized bin. By adding additional dimension, the data points are further pulled apart (FIGURE 2.4(b) and (c)). According to Berkhin (Berkhin 2006), this effect tends to influence similarity for dimensions greater than 15. In the case of document clustering, the curse of dimensionality can be explained by considering two documents x_1 and x_2. Clearly, even if documents x_1 and x_2 do not belong to the same topic, they will typically share some common words. The probability of finding more such random words will increase with the increase in dimensionality of the given corpus.



(a) 11 Objects in One Unit Bin (b) 6 Objects in One Unit Bin (c) 4 Objects in One Unit Bin

FIGURE 2.4 The curse of dimensionality. Data in only one dimension is relatively tightly packed. Adding a dimension stretches the points across that dimension, pushing them further apart. Additional dimensions spread the data even further making high dimensional data extremely sparse **(Parsons, Haque, and H. Liu 2004)**

Several pre-processing steps (some of which will be discussed in section 4.4) are a way of reducing dimensions but they act on a global scale i.e. reduce the global feature set. As a result two broad-based approaches have been proposed in the literature to deal with problems with high dimensionality. The first one tries to capture the semantic relationship in the feature space hence providing the clustering algorithm with more information about the feature relationships. Typically feature transformation techniques such as a low rank approximation using Singular Valued Decomposition (SVD) (section 2.4.1) are used to transform the high dimensional data onto a lower dimensional space. The second approach, known as *biclustering* or *co-clustering*, divides the feature space into *l* clusters and tries to find clusters in the sub-spaces. It involves the simultaneous clustering of rows and columns and exploiting the duality between the two. Moreover, it provides us with a clustering of the features in addition to the instances. Various approaches based on matrix decomposition, information theory, etc have been proposed in the literature. In the next section we explore some techniques adapted to clustering high dimensional data while various co-clustering algorithms are explored in section 2.5 below.

2.4. Using Information about Word Semantics

To overcome the issue of high dimensionality, various approaches have been suggested that take into account the

semantic relationship occurring within the dataset in order to better perform the clustering task. We will refer to these algorithms as semantic based or structure based algorithms. In this section, we examine some of the popular semantic based algorithms proposed in the literature, particularly for text analysis.

2.4.1. Latent Semantic Analysis

Latent Semantic Analysis (LSA) is a well known technique that has been applied to a wide variety of learning tasks such as Information Retrieval (Deerwester et al. 1990), document classification (Chakraborti et al. 2006; Chakraborti, Mukras, et al. 2007; Zelikovitz and Hirsh 2001), and filtering (Dumais 1994). The principle behind Latent Semantic Analysis (LSA) is to determine the significance of words and their similarities in a large document corpus. Such significance depends on the occurrences of the words and the context of their occurrences based on the hypothesis that words that occur in similar contexts are similar in nature.

Mathematical Principle Latent Semantic Analysis was proposed by (Deerwester et al. 1990) as a least square projection method based on the mathematical technique termed as Singular Value Decomposition (SVD). The principle of LSA can be explained by taking the following example (D. I. Martin and M.W. Berry 2007) as shown in Table 2-2 below. The documents C1-C5 are Human-Computer interaction related and the documents M1-M4 are related to graphs. The keywords used in the example are in italics.

Table 2-2 Titles for Topics on Music and Baking

Label	**Titles**
C1	*Human* machine *interface* for Lab ABC *computer* applications
C2	A *survey* of *user* opinion of *computer system response time*
C3	The *EPS user interface* management *system*
C4	*System* and *human system* engineering testing of *EPS*
C5	Relation of *user*-perceived *response time* to error measurement
M1	The generation of random, binary, unordered *trees*
M2	The intersection *graph* of paths in *trees*
M3	*Graph minors* IV: Widths of *trees* and well-quasi-ordering
M4	*Graph minors*: A *survey*

The LSA based technique takes as input data corpus a document by word matrix **A**, whose rows represents keywords and whose columns represent the labels/topics. Each elements A_{ij} represent the number of occurrence of the keyword *i* in the topic *j*. This is shown in Table 2-3. However, using raw frequency may not yield the best results and a transformation of the data is needed (Landauer et al. 2007). Various such schemes have been used such as a sub-linear transformation given by *log* ($freq_i$+1) followed by IDF or a simple TF-IDF weighting as described

previously. Such transformations are needed to take into account the distribution of a word in the given corpus. For example, a word which occurs 25 times in a corpus of, say, 100 documents could be evenly distributed in 25 documents or occur multiple times in fewer documents.

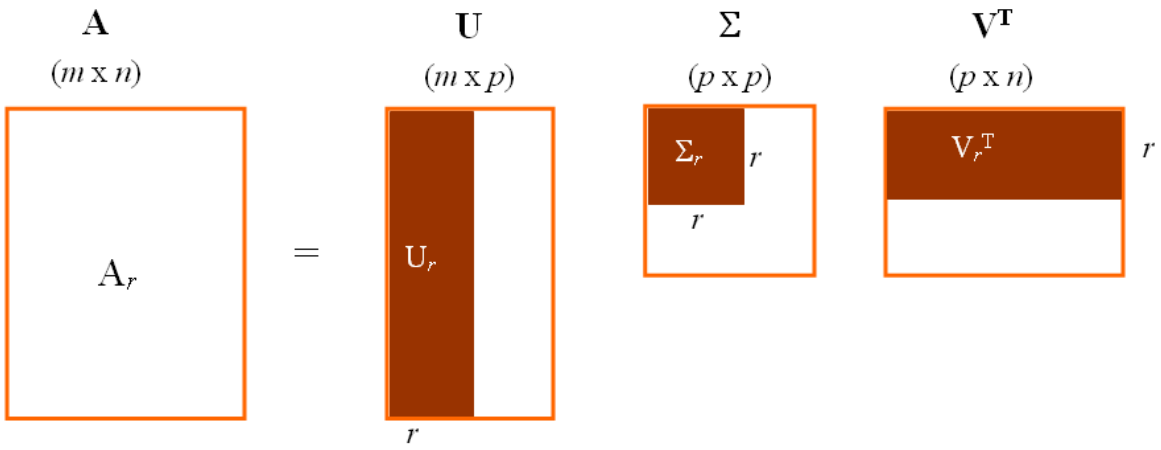


FIGURE 2.5 Diagram for the truncated SVD

The next and most essential step of using LSA is the reduced-rank singular valued decomposition, performed on the transformed matrix, in which the r largest singular values are retained and the remainder set to zero. Mathematically speaking, the matrix **A** is decomposed as a multiplication of 3 matrices given by

(2.11) $$\mathbf{A} = \mathbf{U}\sum\mathbf{V}^{\mathrm{T}}$$

where **U** and **V** are the left and right orthogonal matrices and **Σ** is the singular values matrix. The original matrix **A** is an m by n matrix. The matrix **U** corresponds to the rows by dimension matrix and the matrix **V** to the dimension by columns of the original matrix **A**. The diagram for the truncated matrix $\mathbf{A}_r$ is shown in FIGURE 2.5. A "*compression*" of information $\mathbf{A}_r$ ($\mathbf{A}_r = \mathbf{U}_r \sum_r \mathbf{V}^{\mathrm{T}}{}_r$), is obtained by selecting the top r singular values in the matrix **Σ** by setting the smallest singular values from (r+1…p) to zero. The resulting matrix $\mathbf{A}_r$ is the best (minimum distance) rank r approximation to the original matrix **A**. The first r columns of **U** and **V** are orthogonal but the rows of **U** and **V** are not orthogonal.

For our example of Table 2-3 and using r=2 which corresponds to keeping the highest 2 singular values of **Σ** and only the first 2 columns of **U** and **V**, the resulting term similarity matrix is shown in Table 2-5 (the term-term similarity matrix corresponds to $\mathbf{U}_r\mathbf{\Sigma}_r$ $(\mathbf{U}_r\mathbf{\Sigma}_r)^{\mathrm{T}}$). As can be seen from Table 2-5, the terms '*user'* and '*human'* now have a (relatively) strong similarity value of 0.94 even though the two terms never occur together in any document. On the other hand, the terms '*trees'* and '*computer'* have a similarity value of 0.15, which albeit being (relatively) small is still non-zero.

Table 2-3 Term by document matrix (Kontostathis and Pottenger 2006)

	c1	c2	c3	c4	c5	m1	m2	m3	m4
human	1	0	0	1	0	0	0	0	0
interface	1	0	1	0	0	0	0	0	0
computer	1	1	0	0	0	0	0	0	0
user	0	1	1	0	1	0	0	0	0
system	0	1	1	2	0	0	0	0	0
response	0	1	0	0	1	0	0	0	0
time	0	1	0	0	1	0	0	0	0
EPS	0	0	1	1	0	0	0	0	0
survey	0	1	0	0	0	0	0	0	1
trees	0	0	0	0	0	1	1	1	0
graph	0	0	0	0	0	0	1	1	1
minors	0	0	0	0	0	0	0	1	1

Table 2-4 Deerwester Term-to-Term Matrix, truncated to 2 dimensions (Kontostathis and Pottenger 2006)

	t1	t2	t3	t4	t5	t6	t7	t8	t9	t10	t11	t12
human (t1)	x	0.54	0.56	0.94	1.69	0.58	0.58	0.84	0.32	-0.32	-0.34	-0.25
interface (t2)	0.54	x	0.52	0.87	1.50	0.55	0.55	0.73	0.35	-0.20	-0.19	-0.14
computer (t3)	0.56	0.52	x	1.09	1.67	0.75	0.75	0.77	0.63	0.15	0.27	0.20
user (t4)	0.94	0.87	1.09	x	2.79	1.25	1.25	1.28	1.04	0.23	0.42	0.31
system (t5)	1.69	1.50	1.67	2.79	x	1.81	1.81	2.30	1.20	-0.47	-0.39	-0.28
response (t6)	0.58	0.55	0.75	1.25	1.81	x	0.89	0.80	0.82	0.38	0.56	0.41
time (t7)	0.58	0.55	0.75	1.25	1.81	0.89	x	0.80	0.82	0.38	0.56	0.41
EPS (t8)	0.84	0.73	0.77	1.28	2.30	0.80	0.80	x	0.46	-0.41	-0.43	-0.31
survey (t9)	0.32	0.35	0.63	1.04	1.20	0.82	0.82	0.46	x	0.88	1.17	0.85
trees (t10)	-0.32	-0.20	0.15	0.23	-0.47	0.38	0.38	-0.41	0.88	x	1.96	1.43
graph (t11)	-0.34	-0.19	0.27	0.42	-0.39	0.56	0.56	-0.43	1.17	1.96	x	1.81
minors (t12)	-0.25	-0.14	0.20	0.31	-0.28	0.41	0.41	-0.31	0.85	1.43	1.81	x

Table 2-5 Term-to-Term Matrix on a modified input matrix, truncated to 2 dimensions (Kontostathis and Pottenger 2006)

	t1	t2	t3	t4	t5	t6	t7	t8	t9	t10	t11	t12
human (t1)	x	0.50	0.60	1.01	1.62	0.66	0.66	0.76	0.45	–	–	–
interface (t2)	0.50	x	0.53	0.90	1.45	0.59	0.59	0.68	0.40	–	–	–
computer (t3)	0.60	0.53	x	1.08	1.74	0.71	0.71	0.81	0.48	–	–	–
user (t4)	1.01	0.90	1.08	x	2.92	1.19	1.19	1.37	0.81	–	–	–
system (t5)	1.62	1.45	1.74	2.92	x	1.91	1.91	2.20	1.30	–	–	–
response (t6)	0.66	0.59	0.71	1.19	1.91	x	0.78	0.90	0.53	–	–	–
time (t7)	0.66	0.59	0.71	1.19	1.91	0.78	x	0.90	0.53	–	–	–
EPS (t8)	0.76	0.68	0.81	1.37	2.20	0.90	0.90	x	0.61	–	–	–
survey (t9)	0.45	0.40	0.48	0.81	1.30	0.53	0.53	0.61	x	–	–	–
trees (t10)	–	–	–	–	–	–	–	–	–	x	2.37	1.65
graph (t11)	–	–	–	–	–	–	–	–	–	2.37	x	1.91
minors (t12)	–	–	–	–	–	–	–	–	–	1.65	1.91	x

The purpose of reducing the dimension is to capture the semantic relationships in the data. Words that have similar meanings are near each other and documents that are similar in meaning are near each other in the reduced *r* dimensional space (Homayouni et al. 2005). Konstosthatis and Pottenger (Kontostathis and Pottenger 2006) provide a more concise framework for understanding the working of LSA and show a clear relationship between the similarity values of term pairs and the average number and length of paths between them in their corresponding bi-partite graph. For example, the words *user* and *human* as seen above do not directly co-occur in any document, however the term *user* occurs with the term *interface* which also co-occurs with the term *human*. The term *interface* is thus a transitive relation between the terms *user* and *human* and represents an example of a second-order co-occurrence.

It can be mathematically proven (as done in (Kontostathis and Pottenger 2006)) that the LSA algorithm encapsulates the term co-occurrence information. More precisely, they show that for every non-zero element in the resulting low rank approximation matrix, there exists a connectivity path in the original matrix. In other words, a word or group of words not connected to other words would result in a zero similarity value in the term-term similarity matrix. Taking back the example of Table 2-2, it is evident that the topics can be divided into two subsets C1-C5 and M1-M4. Note that only the word '*survey'* provides a transition between these two sub-sets, and by removing it (setting the corresponding value to 0) the two subsets becomes disjoint. The corresponding term-term matrix generated by removing the word survey is given in Table 2-5. Since there is now no transitive word between '*user'* and '*human'*, the corresponding value is zero.

The above phenomenon is very important in the task of document clustering and information retrieval. It can be argued that the terms *user* and *human* can be used interchangeably to specify someone who utilizes, say a software system. Using a word matching measure (such as Cosine, Euclidean, etc) on the document vectors, the similarity between the words and the corresponding documents containing these terms would be zero. Using a low rank approximation on the other hand, can generate non-zero similarities if the two terms co-occur with other terms which in turn would generate a similarity value between the documents containing these two terms.

A limitation of the LSA, however, is that the results can not be easily interpreted. While a non-zero value in a low rank matrix indicate transitivity, it is difficult to interpret the value obtained. For example, it is not clear how the value of 0.94 was obtained between the terms *user* and *human*. Moreover, the approximated low-rank matrix may also contain negative values whose interpretation is also non-trivial. As will be seen in the next chapter, we will use the concept of transitivity and higher-order co-occurrences to define a new algorithm that explicitly takes the (weighted) higher order paths between words and documents and has a clear interpretation of the results.

The choice of dimensions *r* plays a critical role in the performance of LSA and the best choice of rank remains an open question. Several authors (D. I. Martin and M.W. Berry 2007; Lizza and Sartoretto 2001; Jessup and J. H. Martin 2001) have performed empirical evaluation on large datasets and suggest a range of between 100 and 300 for the number of dimensions to keep. Keeping only a small number of dimensions fail to exploit the latent relationship present in the data and keeping too many dimensions amounts to word matching. In practice, the optimal value of *r* depends on the corpus and the value is usually empirically determined. (Landauer et al. 2007) found the optimal number to be 90 for the task of information retrieval on a collection of medical texts but showed a fairly large

plateau for the value of r. Schütze and Silverstein (Schütze and Silverstein 1997) observed that accurate performance for the document clustering task was achieved using only a small number of dimensions (20-100).

2.4.2. Other Approaches

Several other alternate clustering approaches particularly for document clustering like document clustering using random walk, clustering by friends, higher-order co-occurrences, etc have been proposed.

Clustering by Friends Dubnov et al (Dubnov et al. 2002) proposed a technique labeled "clustering by friends, which uses a pair-wise similarity matrix to extract the two most prominent clusters in the data. The algorithm is non-parametric and iteratively employs a two-step transformation of the proximity matrix.

- Normalization Step: Given a proximity matrix $\mathbf{S}^{(0)}$, let $\mathbf{s}_i=(S_{i1},S_{i2},\ldots,S_{im})$ denote the similarity between a document i and all other documents. The vector $\mathbf{s}_i$ is normalized by dividing each of its components by $||\mathbf{s}_i||$. The resulting transformed matrix is denoted by $\mathbf{S}'$.
- Re-estimation step: A new proximity matrix is calculated from the normalized proximity matrix $\mathbf{S}'$ denoted by $\mathbf{S}'^{(t)}$ for t=1,2,…

Thus, documents are represented by their proximity to other documents. In this representation, documents that are close together (belong to the same cluster) have common "friends" or "non-friends". At each iteration, documents with similar friends are brought closer together and after a few iterations a two-value matrix is observed that corresponds to the two principal clusters. The process may be repeated for finding more clusters. Dubnov et al. used the L_1 normalization and KL-divergence for the re-estimation step but other similarity values and normalizations can be used. This method is intuitive since it looks at the neighborhood of a document to determine its similarity with other documents. Thus, even if two documents do not share a common set of vocabulary, they might still be clustered. The algorithm is also reasonably resistant to noise since a few odd documents will not necessarily affect its most similar friends or non-friends. It must be noted that while the algorithm is advertized as non-parametric, the output is a hierarchical tree and to obtain the actual clusters, the tree must be cut at some point. Also the algorithm is limited to finding two clusters at a time.

Document Clustering Using Random Walk Güneş Erkan (Erkan 2006) proposed a new document vector representation where a document is represented as an m-dimensional vector, where m is the number of documents. Instead of the usual frequency of occurrence like in the *bag-of-words* model, they define $\mathbf{A}$ as a proximity matrix (hence n=m) in their technique whose entries represent a measure of the generation probability of document j from the language model of document i. The concept is similar to building a proximity matrix and then iteratively improving the similarity values. A new directed graph is now generated where the nodes are the documents and the edges are the probabilities of generating one document based on the language model of the other document. The algorithm then uses a restricted random walk to reinforce these generation probabilities and find "hidden patterns" in this graph. These weighted links are incremented by calculating probabilities of starting from d_i and end up at d_j in a t-step walk. The value of t is kept low since

- an infinite size walk ending up at d_j will be roughly the same irrespective of the starting point.
- the aim is to discover local links "inside a cluster" that separates the cluster with the rest of the graph and not a global semantic structure of the whole graph.
- the generation similarities lose their significance because they are multiplied at each step.

The clustering is performed using a *k*-means approach on the final proximity matrix.

Co-training of Data Maps and Feature Maps (CoFD) This algorithm proposed by (S. Zhu, T. Li, and Ogihara 2002) is based on the concept of representing data using two maps (clustering) – a sample map $C_X : X \rightarrow \{1,2,\ldots k\}$ for the document set, and a feature map $C_Y : Y \rightarrow \{1,2,\ldots k\}$ for the feature set. The algorithm is based on the Maximum Expectation Principle[5] to measure the "concept" of each map and the resulting model.

The informal idea is to find the sample map C_X and feature map C_Y from which the original data **A** was most likely generated. Given the number of clusters *k*, we define C_X and C_Y as clustering of the data and measure the likelihood of its generation. If we consider the *m* by *k* matrix, **B**, such that B_{ij}=1 if $C_X(i)=C_Y(j)$ and 0 otherwise., then we could measure the likelihood of generation of the data from the maps C_X and C_Y by considering $P(A_{ij}=b|B_{ij}(C_X,C_Y)=c)$ where $P()$ represents the probability, and *b* and *c* belong to $\{0,1\}$. This is interpreted as the j^{th} feature active in the i^{th} sample in the real data conditioned on the j^{th} feature being active in the i^{th} sample given by the model C_X and C_Y. The assumption here is that the conditional probability is only dependent on the values of *b* and *c*. One can now estimate the likelihood of the model using

$$\log L(C_X, C_Y) = \log \prod_{i,j} P\left(A_{ij} \mid B_{ij}(C_X, C_Y)\right) \tag{2.12}$$

where log *L* is the log likelihood. Our goal here is to find $\arg\max_{C_X,C_Y} \log L(C_X,C_Y)$. For this, the authors use a hill-climbing algorithm based on alternately estimating C_X and C_Y. They use an approximate (and greedy) approach to estimate C_Y given C_X, for instance, by optimizing each feature $C_Y(j)$ (i.e. the cluster label of feature *j*) by minimizing the conditional entropy $H(A_{*j}| B_{*j}(C_X,C_Y))$ where the '*' in the subscript means over all rows. $C_Y(j)$ is assigned to the class resulting in the minimum entropy. Optimizing C_X given C_Y is similar. The outline of the algorithm is as follows:

1. Randomly assign the data points to a sample map $C_X^{(0)}$. Set t=1 (t is the iteration)
2. Compute feature map $C_Y^{(t)}$ from $C_X^{(t-1)}$ to increase the likelihood
3. Compute sample map $C_X^{(t)}$ from $C_Y^{(t-1)}$
4. $t=t+1$. Repeat steps 2 and 3 until no change.

The algorithm starts with an initial clustering then iteratively improves the clustering by fixing C_X and improves C_Y and vice versa. This algorithm can be compared to co-clustering algorithms (see 2.5) where $k=l$ and a mixture of

[5] Maximum Expectation Principle states that the best model is one which has the highest likelihood of generating the data.

models based algorithm is iteratively applied on the sample and feature set.

Several other works have also been proposed that takes advantage of the structural relationship in the data to estimate similarity measures in high dimensional data, such as the shared nearest neighbor approach by (Jarvis and Patrick 1973) and its extension (Ertoz, Steinbach, and V. Kumar 2002). Similarly, the point wise mutual information (PMI) based Second-Order Co-occurrence PMI (SOC-PMI) proposed by (Islam and Inkpen 2006) exploits words that co-occur with other words. The SOC-PMI method maintains a sorted list of the *closest neighbors* of a given word using PMI. It should be noted that while these methods have been proposed to compute the proximity between words, they can also be used to calculate the proximity between documents. In the next section, we will see a further development of the concept of mutual information that performs a simultaneous clustering of both the documents and words by maximization of mutual information.

A common theme in these alternative approaches to clustering high dimensional data is that they try to implicitly find relationship between (the features) in the data. As pointed out in (Hastie et al. 2005), all methods that overcome the dimensionality problems have an adaptive metric for measuring neighborhoods. Using this additional information helps to reduce the adverse effect of the curse of dimensionality. This idea will also form the basis of our novel co-similarity based co-clustering technique that forms the basis of this thesis (Chapter 3). For the moment, however, we proceed to discuss a different approach to dealing with high dimensional data which is to simultaneously cluster the feature set, thereby explicitly reducing the n-dimensional feature space and then find clustering that are defined in the subspaces. This is known as *biclustering* or *co-clustering* which we discuss in the next section.

2.5. Co-Clustering

As mentioned in Chapter 1, co-clustering or simultaneous clustering of rows and columns of two-dimensional data matrices is a data mining technique with various applications such as text clustering and microarray analysis. Most of the proposed co-clustering algorithms such as (Deodhar et al. 2007; Long, Z. M Zhang, and P. S Yu 2005; Dhillon, Mallela, and Modha 2003), among many others, work on the data matrices with a special assumption of the existence of a number of mutually exclusive row and column clusters. Several co-clustering algorithms have also been proposed that view the problem of co-clustering as that of a bi-partite graph partitioning by finding sub-graphs such that the weight of the edges between the sub-graphs are minimized (Abdullah and A. Hussain 2006; Long et al. 2006; Dhillon 2001; Madeira and Oliveira 2004).

Co-clustering can be applied in situations where a data matrix **A** is given in which its elements A_{ij} represent the relation between its rows i and its columns j, and we are looking for subsets of rows with certain coherence properties in a subset of the columns. As opposed to independently clustering rows and columns of a given data matrix, co-clustering is defined as the simultaneously partitioning of the rows and the columns such that a partitioning (cluster) of the rows show some statistical relevance in a partition (cluster) of the columns. This is shown in FIGURE 2.6.

In recent years, co-clustering has been successfully applied to a number of application domains such as:

- Bioinformatics: co-cluster genes and conditions (Kluger et al. 2003; Cho et al. 2004; Y. Cheng and Church 2000; Madeira and Oliveira 2004; Barkow et al. 2006)
- Text Mining: co-cluster terms and documents (and categories) (Dhillon et al. 2003; Deodhar et al. 2007; Long et al. 2005; B. Gao et al. 2005; Takamura and Matsumoto 2002; Dhillon 2001)
- Natural Language Processing: co-cluster terms & their contexts for Named Entity Recognition (Rohwer and Freitag 2004)
- Image Analysis: co-cluster images and features (Qiu 2004; J. Guan, Qiu, and X. Y Xue 2005)
- Video Content Analysis: co-cluster video segments & prototype images, co-cluster auditory scenes & key audio effects for scene categorization (Zhong, Shi, and Visontai 2004; R. Cai, Lu, and L. H Cai 2005)
- Miscellaneous: co-cluster advertisers and keywords (Carrasco et al. 2003)

In this thesis, we are more concerned with co-clustering for text mining and bioinformatics and will focus on algorithms that have been proposed in these domains. We attempt to find connections among the several popular algorithms and, as such, we have identified several major families of these algorithms which we explore in the following sub-sections. It should be noted that soft co-clustering algorithms have also been proposed in the literature (Shafiei and Milios 2006) but these algorithms are beyond the scope of this thesis and we limit ourselves to the hard clustering problem.

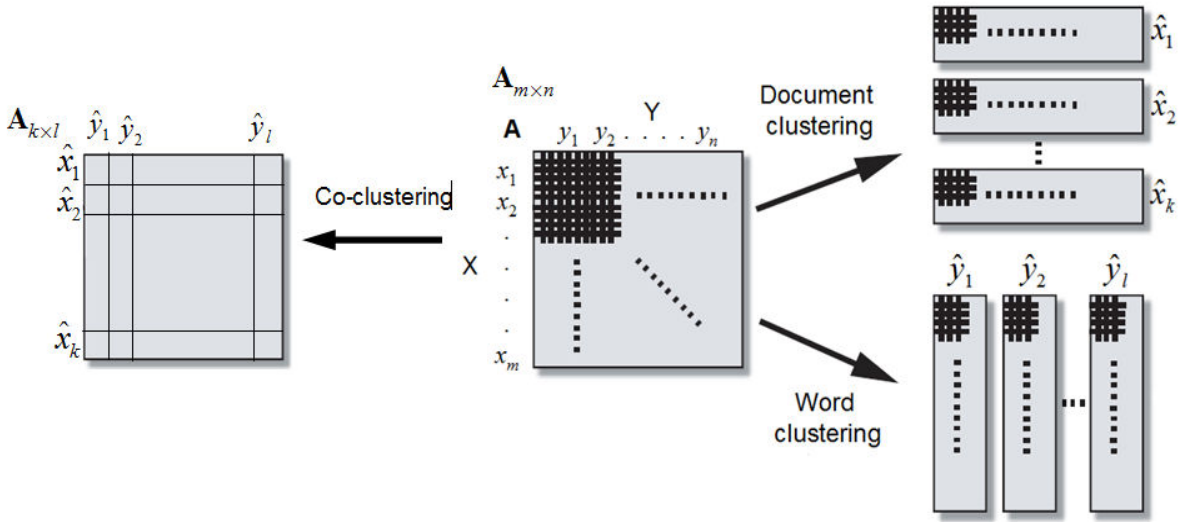


FIGURE 2.6 The Two-Way clustering and Co-clustering Frameworks

2.5.1. Matrix Decomposition Approaches

Given a matrix $\mathbf{A} \in \Re^{m \times n}$, we are interested in finding an approximate matrix **Z** to **A** such that,

(2.13) $$\|\mathbf{A}\text{-}\mathbf{Z}\|_F^2$$

is minimized, where $\|.\|_F$ is the Frobenius Norm[6]. The interest is in finding a matrix **Z** that can usually be defined as matrix decomposition or a low-rank approximation to the original matrix **A**. This decomposition of **Z** allows us to "capture" the hidden block structure of the original matrix **A**. Depending on the approach used, several conditions can be placed on the matrix **Z**. For instance, in the case of LSA, **Z** can be a low-rank approximation to the matrix **A**.

LSA as a Co-clustering Algorithm

As seen previously, the Latent Semantic Analysis is a remarkable matrix factorization approach based on SVD typically used for dimensionality reduction. However, one may relate the SVD to a co-clustering task by considering an *idealized* or *perfect* co-cluster matrix with a block diagonal structure. Define the matrix $\mathbf{A} = [\hat{\mathbf{A}}_1, \hat{\mathbf{A}}_{2,\ldots,} \hat{\mathbf{A}}_k]$ (k is the number of co-clusters) where each $\{ \hat{\mathrm{A}}_i\}$, i=1..k are arbitrary matrices corresponding to the row cluster $\hat{x}_i$ and column cluster $\hat{y}_i$. All other values in the matrix **A** are assumed to be zero as shown in FIGURE 2.7 below. It is intuitive, based on our earlier discussion of transitivity, that each pair of singular vectors will quantify one bicluster from the matrix A. For each $\{\hat{\mathbf{A}}_i\}$, there will be a singular vector pair ($\mathbf{u}_i$, $\mathbf{v}_i$) such that non-zero elements of $\boldsymbol{u}_i$ correspond to rows occupied by $\hat{\mathbf{A}}_i$ and non-zero components of $\boldsymbol{v}_i$ correspond to columns occupied by $\hat{\mathbf{A}}_i$. However, such *ideal* or *perfect* biclusters are rare and in most practical cases, elements outside the diagonal block might be non-zero. Even in such a case, if the block diagonal elements are dominant in the matrix **A**, *"the SVD is able to reveal the co-clusters too as dominating components in the singular vector pairs"* (Busygin, Prokopyev, and Pardalos 2008).

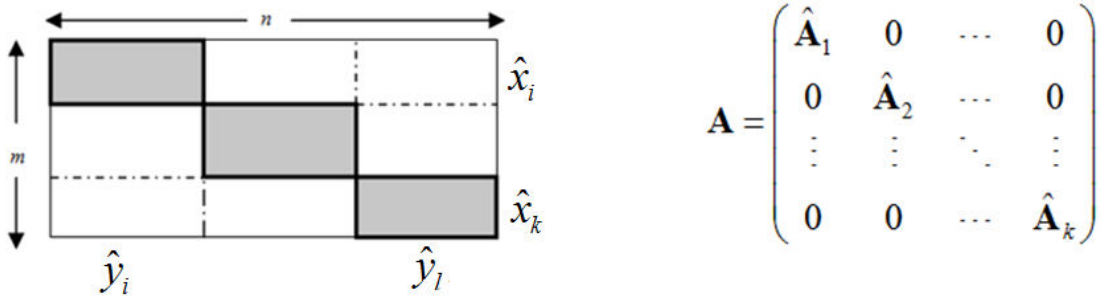


FIGURE 2.7 An ideal matrix with block diagonal co-clusters

To re-iterate the connection between LSA and co-clustering, we may look as the working of LSA from a

[6] The Frobenius Norm, also known as a Euclidean Norm, is defined as the square root of the sum of the square of the elements of a matrix, $\|\mathbf{A}\|_F^2 = \sqrt{\sum_{i=1..m} \sum_{j=1..n} (A_{ij})^2}$

geometrical perspective. Consider the geometrical representation of documents in terms of their words as showing in FIGURE 2.8 below. In a traditional (vector space model) representation (FIGURE 2.8 (a)), the words form a natural axes of the space and two terms (such as Term 1 and Term 2) are orthogonal because there is no similarity between them. Documents are represented as vectors in this terms space and the number and nature of words in a document determine its length and direction respectively. As a result, if we compare two documents, say Doc 3 and Doc 4, which do not share any common term, then the resulting similarity between them, will be zero.

FIGURE 2.8 (b) shows a geometric representation of documents and terms, the axes being derived from the SVD. Both terms and documents are represented in this reduced *r*-dimensional space after reducing the rank of the matrix. In this representation, the derived LSA axes are orthogonal. The Cosine value between Doc 3 and Doc 4 will be non-zero. As can be seen, in the LSA representation, the geometrical analogy corresponds to both the words and documents being represented in the same space and any clustering of either on the low-rank approximation matrix are dependent on the other. Note that when $\mathbf{Z} = \mathbf{U}_r\sum_r\mathbf{V}_r^T$, then **Z** represents the *r*-rank approximation of **A** such that the value in equation (2.13) is minimized over all rank *r* matrices. Geometrically, this amounts to finding the "best-fit" subspace for the points of **A** (Landauer et al. 2007).

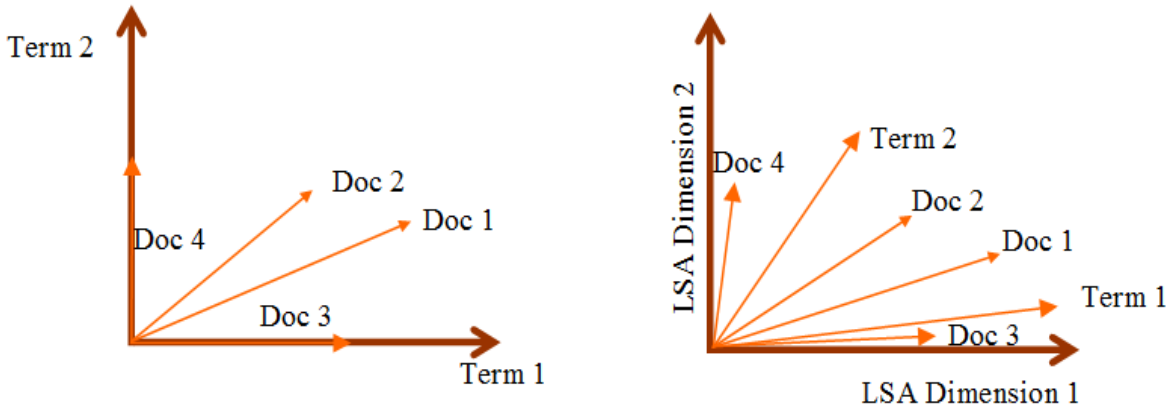


FIGURE 2.8 Comparison (a) of vector space, and (b) LSA representation **(**adapted from (Landauer et al. 2007)**)**

Block Valued Decomposition

The Block Value Decomposition (BVD) proposed by (Long et al. 2005) can be considered as a general framework for co-clustering. It is a partitioning-based co-clustering algorithm that seeks to find a triple decomposition of a *dyadic*[7] data matrix. The matrix is factorized into 3 components — the row coefficient matrix **R**, the block value matrix **B** and the column coefficient matrix **C**. These coefficient matrices denote the degree to which the rows and columns are associated with their clusters, whereas the block value is an explicit and compact

[7] A dyadic data refer to a domain of finite sets of objects in which the observations are made for dyads i.e., pairs with one element in each set.

representation of the hidden co-cluster structure in the original matrix, **A**.

We wish to partition the matrix A into *k* row clusters and *l* column clusters. This partitioning or compact *k* by *l* representation of **A** is contained in the Block value matrix, **B**. For a document by word matrix **A**, each value corresponding to a row in **R** gives the association of that document to each of *k* possible clusters of documents and each column in **C** contains the degree of association of each word in **A** to each of the *l* possible partitioning of the words. More formally, the block value decomposition of a data matrix $\mathbf{A} \in \Re^{m \times n}$ is given by the minimization of

$$f(\mathbf{R}, \mathbf{B}, \mathbf{C}) = \left\| \mathbf{A} - \mathbf{RBC} \right\|^2 \tag{2.14}$$

where $\mathbf{R} \in \Re^{m \times k}$, $\mathbf{C} \in \Re^{l \times n}$, and $\mathbf{B} \in \Re^{k \times l}$ and subject to the constraint that $\forall ij$: $R_{ij} \geq 0$ and $C_{ij} \geq 0$. We seek to approximate the original data matrix **A** by the reconstruction matrix, **RBC**.

The objective of the algorithm is to find the matrices **R**, **B** and **C** such that equation (2.14) is minimized. The matrices **R**, **B** and **C** are randomly initialized and then iteratively updated to converge to a local optimum. The authors in (Long et al. 2005) show that the following updating rules for these matrices are monotonically non-increasing, thus ensuring that we converge on local minima,

$$R_{ij} \leftarrow R_{ij} \frac{(\mathbf{AC}^{\mathrm{T}}\mathbf{B}^{\mathrm{T}})_{ij}}{(\mathbf{RBCC}^{\mathrm{T}}\mathbf{B}^{\mathrm{T}})_{ij}} \tag{2.15}$$

$$B_{ij} \leftarrow B_{ij} \frac{(\mathbf{R}^{\mathrm{T}}\mathbf{AC}^{\mathrm{T}})_{ij}}{(\mathbf{R}^{\mathrm{T}}\mathbf{RBCC}^{\mathrm{T}})_{ij}} \tag{2.16}$$

$$C_{ij} \leftarrow C_{ij} \frac{(\mathbf{B}^{\mathrm{T}}\mathbf{R}^{\mathrm{T}}\mathbf{A})_{ij}}{(\mathbf{B}^{\mathrm{T}}\mathbf{R}^{\mathrm{T}}\mathbf{RBC})_{ij}} \tag{2.17}$$

As compared to the SVD approach, BVD has a more intuitive interpretation. Firstly, each row and column of the data matrix can be seen as a additive combination of block values since BVD doesn't allow negative values. The product **RB** is a matrix containing the basis of the column space of **A** and the product **BC** contains the rows space of **A**. Each column of the *m*-by-*l* matrix **RB** captures the base topic of a word cluster and each row of the *k*-by-*n* matrix **BC** captures the base topic of a particular document cluster.

As opposed to SVD, BVD is an explicit co-clustering approach characterized by the *k*-by-*l* block structure matrix **B**. The updating of **R**, **B** and **C** are intertwined and the strength of each row coefficient and column coefficient association in **R** and **C** respectively depends on the other and the block structure.

Other Approaches

Several low-rank matrix approximation approaches such as the Independent Component Analysis (ICA) (Oja, Hyvarinen, and Karhunen 2001; Comon and others 1994), Principle Component Analysis (PCA) (Ringnér 2008;

Jolliffe 2002; Hotelling 1933), Random projection (Bingham 2003), etc have been proposed in the literature. However, these cannot easily be labeled as co-clustering approaches and, hence, are not discussed further. Interested readers can find a discussion on such methods in (Bingham 2003), for example. Here, we briefly examine the Non-negative Matrix Factorization (NMF) proposed by (D. D Lee and Seung 1999) and used for document clustering by (W. Xu, X. Liu, and Gong 2003).

The Non-negative Matrix Factorization algorithm proposed by (D. D Lee and Seung 1999) is a matrix factorization technique. In the semantic space derived by the non-negative matrix factorization (NMF), each axis captures the base topic of a particular document cluster, and each document is represented as an additive combination of the base topics. The cluster membership of each document can be easily determined by finding the base topic (the axis) with which the document has the largest projection value.

Mathematically, given a non-negative data matrix **A**, NMF tries to find an approximate matrix **A≈WH** where **W** and **H** have non-negative components. As compared to the SVD based decomposition, NMF has two basic differences

1. The latent semantic space derived by NMF need not be orthogonal, and
2. Each document is guaranteed to take a non-negative value in each direction.

This can be interpreted as each axis in the derived space has a straightforward relation with a document cluster. The NMF can be seen as a special case of BVD by considering the matrix decomposition as **A≈WIH**, where **I** is the identity matrix. Thus, NMF can be considered as a biclustering algorithm with the additional constraint that the number of word clusters is the same as the number of document clusters and that each document cluster must be associated with a word cluster.

2.5.2. Bipartite Graph Partitioning Approaches

The idea of using graph-theoretic techniques have been considered for clustering and many earlier hierarchical agglomerative clustering algorithms (Strehl, Ghosh, and Mooney 2000; Duda et al. 2001) among others. The idea behind graph partitioning approaches is to model the similarity between objects (e.g. documents) by a graph whose vertices correspond to objects and weighted edges give the similarity between the objects. The objective is then to find a cut in the graph such that similarities between vertices (as denoted by edges) are maximized within sub-graphs and similarity between sub-graphs is minimized. In this sub-section, we briefly introduce some concepts related to graph theory and linear algebra and use it to present some standard spectral clustering approaches proposed in the literature.

Definition 1 Let G = {V,E} be a graph with a set of vertices V={1,2,… ,$|V|$} and set of edges E. The adjacency matrix is defined as

(2.18) $$A_{ij} = \begin{cases} E_{ij}, & if\ there\ is\ a\ link\ between\ i\ and\ j \\ 0, & otherwise \end{cases}$$

where E_{ij} corresponds to some weighting scheme on the edge between *i* and *j* (section 2.1)

Definition 2 The degree of a vertex v_i denoted by d_i is given by

(2.19) $$d_i = \sum_j A_{ij}$$

and the degree matrix of the graph is a diagonal matrix defined as

(2.20) $$D_{ij} = \begin{cases} d_i, & if\ i = j \\ 0, & otherwise \end{cases}$$

Note that the degree matrix **D** is a diagonal matrix with degree d_i in the diagonal. For a subset $S \subset V$, we denote the complement V\S as $\overline{S}$.

Given a partitioning of the vertex set V into k subsets V_1, V_2, …, V_k, the most common objective functions measure the quality of the partitioning is given by a measure of the cut, which we would like to minimize. Mathematically speaking,

(2.21) $$cut(V_1, V_2, \ldots, V_k) = \sum_{a<b} \sum_{i \in V_a, j \in V_b} A_{ij}$$

The problem with the above objective function is that in many cases, the clustering of the graph results in separating just one vertex from the rest of the graph. Clearly, this is not a desired clustering since a cluster should be reasonably large. As a result, two popular measures have been proposed in the literature

- *RatioCut* (Hagen and Kahng 1992)

  Hagen and Kahng proposed a simple extension of the *cut* function to take into account the number of vertices

(2.22) $$RatioCut(V_1, V_2, \ldots, V_k) = \sum_{i=1}^{k} \frac{cut(V_i, \overline{V})}{|V_i|}$$

- NormalizedCut (Shi and Malik 2000)

  Shi and Malik proposed including the weights of the edges vol(V) as a normalizing factor and hence,

(2.23) $$NCut(V_1, V_2, \ldots, V_k) = \sum_{i=1}^{k} \frac{cut(V_i, \overline{V})}{vol(V_i)} \quad where \quad vol(V_i) = \sum_{j \in V_i} d_j$$

Note that both objective functions try to achieve clusters that are "balanced" since both will take small values if the cluster V_i is large. However, adding these conditions into the simple cut function results in a *mincut* problem that is NP hard, see for example (D. Wagner and F. Wagner 1993). The spectral clustering algorithm described below is a popular method used to solve this *mincut* problem by relaxing some of the conditions.

Spectral Co-clustering

Spectral clustering has gained popularity over the years in different domains such as document clustering ((Dhillon 2001) , image processing (Shi and Malik 2000), bioinformatics (Kluger et al. 2003), etc. It is simple to implement and can be solved using standard linear algebra based software. For a self-contained detailed tutorial on spectral clustering including a brief introduction to graph theory, graph Laplacians and the interpretation of spectral clustering, we refer the reader to (von Luxburg 2007). Similarly, a survey on various spectral clustering algorithms can be found in (Filippone et al. 2008a).

Perhaps the most important tool in spectral clustering is the graph Laplacian matrix. Spectral approach to clustering is strongly connected to *Laplacian Eigen maps*. As described in the previous section, the dimensionality reduction problem aims to find a low dimensional representation of the original data on which clustering is performed. Here we consider a popular spectral co-clustering method proposed by (Dhillon 2001) and describe its relation to the SVD problem considered earlier.

Definition 3 The graph Laplacian **L** of a graph is a |V| by |V| symmetric matrix with one row and column for each vertex such that

(2.24) $$L_{ij} = \begin{cases} d_i, & if\ i = j \\ -A_{ij}, & otherwise \end{cases}$$

The graph Laplacian matrix **L** has many important properties such as

- **L** is a symmetric and semi-definite positive matrix that has *n* eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$.
- The smallest Eigenvalue of **L** is 0 and the corresponding eigenvector is a constant one vector.
- Let G be an undirected graph with non-negative weights, the multiplicity *k* of the 0 eigenvalue of **L** equals the number of connected components $\mathbf{A}_1,\ldots,\mathbf{A}_k$
- The matrices **L**,**D** and **A** are related by the equation **L=D-A**

As described above, we are looking for partitioning of the set of vertices into *k* partitions, $V=V_1 \cup V_2 \cup \ldots V_k$. For a co-clustering of words and documents, the graph partitioning is shown in FIGURE 2.9 below. Let G={X,Y,E} be a graph with two sets of vertices X and Y, and a set of graph edges E connecting vertices of X with vertices of Y. Then the adjacency matrix of the bipartite graph can be expressed as

(2.25) $$\mathbf{M}=\begin{bmatrix} 0 & \mathbf{A} \\ \mathbf{A}^T & 0 \end{bmatrix}$$

Where the first *m* vertices are those of X and the last *n* are those of Y i.e. |V| = |X|+|Y|.

Let a partition $V=V_1 \cup V_2$ be defined as a vector **p** such that

(2.26) $$p_i = \begin{cases} +1, & i \in V_1 \\ -1, & i \in V_2 \end{cases}$$

Theorem 1 Given the Laplacian matrix **L** of **G** and a partition vector **p**, the following condition holds (Dhillon 2001)

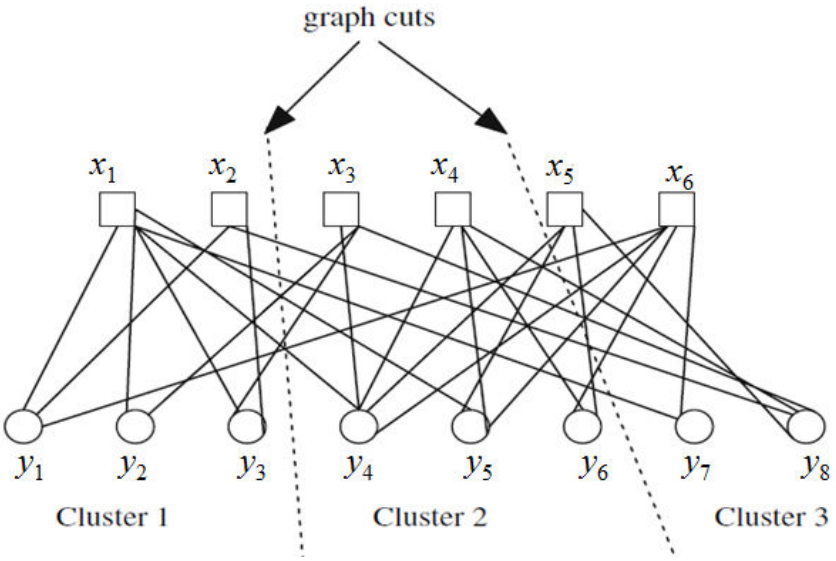


FIGURE 2.9 The square and circular vertices (*x* and *y* respectively) denote the documents and words in the co-clustering problem that is represented as a bi-partite graph. Partitioning of this graph leads to co-clustering of the two data types. **(Rege, Dong, and Fotouhi 2008)**

(2.27) $$\frac{p^T \mathbf{L} p}{p^T p} = \frac{1}{|V|}.4\, cut(V_1, V_2)$$

From the above theorem, it is clear that the cut is minimized by the trivial solutions when all p_i are either +1 or -1. As a result, a different objective function that favors balanced clusters has been proposed. Let each vertex be associated with a weight w_i and **W** denote a diagonal matrix of such weights, and let weight(V_i) be the sum of weights of vertices in the partition V_i, then the Rayleigh Quotient is given by

(2.28) $$Q(V_1, V_2) = \frac{cut(V_1, V_2)}{weight(V_1)} + \frac{cut(V_1, V_2)}{weight(V_2)}$$

It can be shown (Dhillon 2001) that the generalized partition vector **q** equals the above objective function where the generalized partition vector **q** with elements

(2.29)
$$q_i = \begin{cases} +\sqrt{\frac{v_2}{v_1}}, & i \in V_1 \\ -\sqrt{\frac{v_1}{v_2}}, & i \in V_2 \end{cases}$$

satisfies $\mathbf{q}^T\mathbf{We}=0$ and $\mathbf{q}^T\mathbf{Wq}$=weight(V) where $\mathbf{e}=[1,\ldots,1]^T$ is a constant vector.

Theorem 2. Using the definition in Eq. (2.29) above, it can be shown that

(2.30)
$$\frac{\mathbf{q}^T\mathbf{Lq}}{\mathbf{q}^T\mathbf{Wq}} = \frac{cut(V_1,V_2)}{weight(V_1)} + \frac{cut(V_1,V_2)}{weight(V_2)}.$$

Theorem 2 above is the generalization of theorem 1. We would like to find the global minima from equation (2.30) above. This, however, is a NP-complete problem and hence we look to find a relaxation to the optimal generalized partition vector.

Theorem 3. The problem

(2.31)
$$\min_{\mathbf{q}\neq 0} \frac{\mathbf{q}^T\mathbf{Lq}}{\mathbf{q}^T\mathbf{Wq}}, \text{ subject to } \mathbf{q}^T\mathbf{We} = 0$$

is solved when $\mathbf{q}$ is the eigenvector corresponding to the 2nd smallest eigenvalue λ_2 of the generalized eigenvalue problem,

(2.32) $\mathbf{L}z=\lambda\mathbf{W}z$

The solution to the above problem can be obtained via the SVD. Note that for the bipartite case, the Laplacian and Degree matrices are given by

(2.33)
$$\mathbf{L} = \begin{bmatrix} \mathbf{D}_1 & -\mathbf{A} \\ -\mathbf{A}^T & \mathbf{D}_2 \end{bmatrix}, and\ \mathbf{D} = \begin{bmatrix} \mathbf{D}_1 & 0 \\ 0 & \mathbf{D}_2 \end{bmatrix}$$

Where $\mathbf{D}_1$ and $\mathbf{D}_2$ are degree matrices corresponding to X and Y respectively. Then $\mathbf{L}z=\lambda\mathbf{W}z$ can be written as

(2.34)
$$\begin{bmatrix} \mathbf{D}_1 & -\mathbf{A} \\ -\mathbf{A}^T & \mathbf{D}_2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \lambda \begin{bmatrix} \mathbf{D}_1 & 0 \\ 0 & \mathbf{D}_2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

If we denote $u = \mathbf{D}_1^{1/2}x$ and $v = \mathbf{D}_2^{1/2}y$, then

(2.35)
$$\begin{aligned} \mathbf{D}_1^{-1/2}\mathbf{A}\mathbf{D}_2^{-1/2}v &= (1-\lambda)u, \\ \mathbf{D}_2^{-1/2}\mathbf{A}^T\mathbf{D}_1^{-1/2}u &= (1-\lambda)v \end{aligned}$$

which are the precise equations that define the singular value decomposition of the normalized matrix $\mathbf{A}_{norm} = \mathbf{D}_1^{1/2}\mathbf{A}\mathbf{D}_2^{1/2}$. Note that the minimum cut problem can now be solved by finding the singular values since

the singular vectors u_2 and v_2 of $\mathbf{A}_{norm}$ give a real approximation to the discrete optimization problem of minimizing the normalized cut. The co-clustering is performed on the eigenvector pair corresponding to the second smallest eigenvalues by using k-means. For a generalized case of *k*-modal information from $s=[\log_2 k]$ singular vectors corresponding to $u_2,u_3,\ldots u_{s+1}$ and to $v_2,v_3,\ldots v_{s+1,}$ we can formulate an *s*-dimensional matrix **Z** given by

$$\mathbf{Z} = \begin{bmatrix} \mathbf{D}_1^{-1/2}\mathbf{U} \\ \mathbf{D}_2^{-1/2}\mathbf{V} \end{bmatrix} \text{ where } \mathbf{U}= (u_2,u_3,\ldots u_{s+1}) \text{ and } \mathbf{V}=(v_2,v_3,\ldots v_{s+1}) \tag{2.36}$$

Note that the rows of **Z** represent both the object and features of the original dataset **A**.

Spectral clustering has been widely used in the literature. Kluger et al. (Kluger et al. 2003) used spectral clustering to find biclusters in gene expression data. They explore a variant where the dataset is normalized to have constant row and column sum before normalization (known as *Bistochatization* or *Binormalization*). Shi and Malik (Shi and Malik 2000) used spectral clustering for image segmentation. Zha and Ji (Zha and Ji 2002) used spectral clustering to cluster documents from two languages where documents from each type formed vertices of the bi-partite graph. Wu et al. used spectral clustering on stories and features extracted from video key frames that each represent one type of vertices.

Given the popular use of spectral algorithms, it is important to mention the theoretical and intuitive meaning behind the approach. At least two main explanations have been provided for the working of spectral approach (von Luxburg 2007). We consider the random walk model here. Using a random walk model, spectral clustering has an intuitive interpretation. Using the *mincut* objective function, a partitioning with a low cut denotes low probability of jumping between different partitions. Thus, spectral clustering can be interpreted as trying to find a partition such that random walks mostly stays long within the same partition and seldom jumps between partitions.

Other Approaches

Ding (Ding 2003) performed document-word co-clustering by extending the Hopfield networks to partition bipartite graphs and show that the solution is the Principal Component Analysis. Rege et al (Rege et al. 2008) proposed an isoperimetric graph partitioning algorithm for co-clustering known as the Isoperimetric Co-clustering Algorithm (ICA). Instead of the spectral clustering approach which defines *mincut* and tries to find eigenvalues based on SVD, the authors propose a methodology that heuristically minimizes the ratio of the perimeter of the bipartite graph partition and the area of the partition. Their algorithm is based on a solution to a non-singular sparse system of linear equations.

Bipartite graph partition based on crossing minimization has also been proposed in the literature (Abdullah and A. Hussain 2006; Erten and Sozdinler 2009; Ahmad and Khokhar 2007). The basic approach behind these methods is a fast but approximate biclustering by minimizing the number of edge crossings in a bi-partite graph model. To achieve this, several crossing minimization heuristics such as the Barycenter approach proposed by Sugiyama(Sugiyama, Tagawa, and Toda 1981), the Median Heuristic proposed by Eades and Wormald (Eades and Wormald 1986), Genetic algorithms (Mäkinen and Sieranta 1994), simulated annealing (May and Szkatula 1988), etc have been used in the literature.

2.5.3. Information Theoretic Approaches

Information theory has been widely used in the area of machine learning. It has been used for clustering data in both one sided methods (clustering) such as the Information Bottleneck method proposed by Tishby et al. (N. Tishby, F. C Pereira, and Bialek 2000) and two sided (co-clustering) such as the Information Theoretic Co-clustering (ITCC) framework used by Dhillon et al. (Dhillon et al. 2003).

We denote discrete random variables as before by capitals (X,Y,..) and their realizations (values taken by the random variable) by lower case (x, y,…). The probability distribution function of X is denoted by $p(x) = p(X=x)$, and the conditional probability distribution as $p(x|y) = p(X=x|Y=y)$. A brief summary of Information theory and different concepts is given in Appendix II.

The Information Bottleneck (IB) method introduced by Tishby et al (N. Tishby et al. 2000) is an information theoretic approach to clustering. While most of information theory has focused on the problem of transmitting information rather than its meaning, the IB framework sees information theory, particularly source compression, as a natural quantification approach to the meaning of the data. In the context of document clustering, a natural measure of similarity of two documents is the similarity between their word conditional distributions. For every document we can define

$$p(y \mid x) = \frac{n(y \mid x)}{\sum_{y \in Y} n(y \mid x)} \tag{2.37}$$

where $n(y|x)$ is the number of occurrences of the word y in document x. We would like documents with similar conditional word distributions to belong to the same cluster. This formulation of finding a cluster hierarchy of the members of one set (e.g. documents), based on the similarity of their conditional distributions with respect to the members of another set (e.g. words), was first introduced in (Fernando Pereira, Naftali Tishby, and Lillian Lee 1993) and called "distributional clustering". IB formalizes the problem as that of finding a short code for X that preserves the maximum information about Y. It squeezes the information that X provides about Y through a bottleneck formed by a limited set of clusters, $\hat{\mathrm{X}}$. The intuitive idea behind this information theoretic approach is to "find clusters of the members of the set X, denoted here as $\hat{\mathrm{X}}$, such that the mutual information $I(\hat{\mathrm{X}};\mathrm{Y})$ is maximized under a constraint extracted from X, $I(\hat{\mathrm{X}};\mathrm{X})$ " (N. Slonim and N. Tishby 2000). Here the compactness is determined by the mutual information between the random variable X and clusters $\hat{\mathrm{X}}$ given by $I(\hat{\mathrm{X}};\mathrm{X})$, while the quality of the clustering is given by the fraction of the information captured about Y given by $I(\hat{\mathrm{X}};\mathrm{Y}) / I(\mathrm{X};\mathrm{Y})$.

Co-clustering can be achieved by applying the IB algorithm to both the words and documents alternately. In this regard, two such approaches have been proposed to extend the algorithm for two-way clustering. Slonim and Tishby (N. Slonim and N. Tishby 2000) proposed a Double Clustering (DC) algorithm consisting of two steps: first cluster the words using IB and then define a new conditional probability distribution using these word clusters instead of

the individual words. Each word y is represented by its conditional word cluster over the set of documents, $p(x|y)$. From this, we can use distributional clustering algorithm to obtain the word clusters $\hat{Y}$. As a second step for document clustering, we use these word clusters to replace the original representation of documents i.e. instead of using the conditional distribution of words, $p(y|x)$ to represent a document we now use its conditional distribution over the word clusters $p(\hat{y}|x)$ defined by

$$p(\hat{y} \mid x) = \frac{n(\hat{y} \mid x)}{\sum_{\hat{y} \in \hat{Y}} n(\hat{y} \mid x)} = \frac{\sum_{\hat{y} \in \hat{Y}} n(y \mid x)}{\sum_{\hat{y} \in \hat{Y}} \sum_{y \in \hat{y}} n(y \mid x)} \tag{2.38}$$

The algorithm generates a *coarse* feature set in the form of feature clusters and the corresponding conditional distribution can help reduce the effect of noise and sparseness that might exist in the original dataset.

El-Yaniv and Souroujon (El-Yaniv and Souroujon 2001) extended the above DC algorithm to iterative over word clusters and document clusters several time. Thus, words are grouped into word clusters and used as input to define the conditional probability distribution when clustering for documents using the IB framework. The resulting document clusters are in turn used as input to define the conditional probability when clustering for words and the process is repeated several times. The resulting algorithm, called the Iterative Double Clustering (IDC) yields a significant improvement of the accuracy (section 4.2) of the document clustering over the DC algorithm. However, no convergence properties or theoretical explanation of the iterations have been proposed and the optimal number of iterations is empirically determined.

Information Theoretic Co-clustering

The Information Theoretic Co-clustering (ITCC) algorithm, proposed by Dhillon et al. (Dhillon et al. 2003), is similar to the IB framework but generalizes to simultaneously clustering the two dimensions. The algorithm treats the non-negative contingency table as a joint probability distribution between two random variables X and Y that take values over the rows and columns respectively. Given a contingency matrix, the ITCC algorithm tries to minimize the information loss between the contingency matrix and its approximation given by the co-clustered matrix.

More formally, given a contingency matrix, $p(x,y)$, describing a joint probability distribution, and a target matrix of size $k \times l$ that is an optimal co-clustering, with an assignment of each row to one of the k row-clusters, and each column to one of the l column-clusters, that minimizes the loss of information between the original matrix $p(x,y)$ and the clustered matrix $p(\hat{x},\hat{y}) = \sum_{x \in \hat{x}, y \in \hat{y}} p(x,y)$ as follows:

$$\min_{\hat{X},\hat{Y}} I(X;Y) - I(\hat{X};\hat{Y}) \tag{2.39}$$

This loss can be written in the form of a Kullback Leibler divergence between $p(x,y)$ and $q(x,y)$ where $q(x,y) = p(\hat{x},\hat{y})p(x \mid \hat{x})p(y \mid \hat{y})$ given by,

(2.40) $$D_{KL}(p(\mathrm{X};\mathrm{Y}) \,||\, q(\mathrm{X},\mathrm{Y}))$$

Thus, finding an optimal co-clustering is equivalent to finding a distribution *q*, which is close to p in terms of KL divergence. *q* has the same marginal distributions of *p*, and by working on *q* we can reach our goal. We illustrate this relationship between *p* and *q* with a small example. Let *p*(X,Y) be given as follows (Dhillon et al. 2003):

(2.41) $$p(\mathrm{X};\mathrm{Y}) = \begin{bmatrix} .05 & .05 & .05 & 0 & 0 & 0 \\ .05 & .05 & .05 & 0 & 0 & 0 \\ 0 & 0 & 0 & .05 & .05 & .05 \\ 0 & 0 & 0 & .05 & .05 & .05 \\ .04 & .04 & 0 & .04 & .04 & .04 \\ .04 & .04 & .04 & 0 & .04 & .04 \end{bmatrix} \quad p(\hat{\mathrm{X}};\hat{\mathrm{Y}}) = \begin{bmatrix} .3 & 0 \\ 0 & .3 \\ .2 & .2 \end{bmatrix}$$

The joint probability distribution $p\left(\hat{X},\hat{Y}\right)$ can be obtained when we consider the natural distribution of the rows into 3 row clusters as $\hat{x}_1 = \{x_1, x_2\}, \hat{x}_2 = \{x_3, x_4\}$ and $\hat{x}_3 = \{x_5, x_6\}$. Similarly, the columns form two clusters as $\hat{y}_1 = \{y_1, y_2, y_3\}$ $\hat{y}_2 = \{y_4, y_5, y_6\}$. It can be verified using mutual information (equation(2.39)) that the loss in information as a result of this clustering is only 0.957 and that any other clustering of *p* would result in a higher loss in mutual information. Now using the definition of *q* above, we can easily verify that the corresponding approximation *q*(X,Y) is defined as,

(2.42) $$q(\mathrm{X};\mathrm{Y}) = \begin{bmatrix} .054 & .054 & .042 & 0 & 0 & 0 \\ .054 & .054 & .042 & 0 & 0 & 0 \\ 0 & 0 & 0 & .042 & .054 & .054 \\ 0 & 0 & 0 & .042 & .054 & .054 \\ .036 & .036 & 0.28 & .028 & .036 & .036 \\ .036 & .036 & .028 & 0.28 & .036 & .036 \end{bmatrix}$$

Interestingly, the Kullback Leibler divergence between *p* and *q*, $D_{KL}(p||q)$ = 0.0957 is the same as the loss in mutual information between *p*(X,Y) and $p\left(\hat{\mathrm{X}},\hat{\mathrm{Y}}\right)$.

In the context of co-clustering, it is more intuitive to write the probabilities as,

(2.43) $$p\left(\mathrm{x},\mathrm{y},\hat{\mathrm{x}},\hat{\mathrm{y}}\right) = p(\hat{x},\hat{y})\,p(x,y \mid \hat{x},\hat{y})$$

(2.44) $$q\left(\mathrm{x},\mathrm{y},\hat{\mathrm{x}},\hat{\mathrm{y}}\right) = p(\hat{x},\hat{y})\,p(x \mid \hat{x})\,p(y \mid \hat{y})$$

Comparing the distributions *p* and *q* above, it can be shown (Dhillon et al. 2003) that the distribution *q* tries to approximate the conditional joint distribution $p(x,y \mid \hat{x},\hat{y})$ as a product of $p(x \mid \hat{x})$ and $p(y \mid \hat{y})$, i.e. the co-clustering tries to capture the joint distribution X and Y through the individual row and column cluster distributions.

The following lemma (Dhillon et al. 2003) forms the basis of their co-clustering algorithm.

Lemma 1.

Given an assignment C_X and C_Y of a row to a row cluster and a column to a column cluster, where C_X and C_Y are given by,

(2.45) $$C_X : \{x_1, x_2, ..., x_m\} \longrightarrow \{\hat{x}_1, \hat{x}_2, ..., \hat{x}_k\}$$

(2.46) $$C_Y : \{y_1, y_2, ..., y_n\} \longrightarrow \{\hat{y}_1, \hat{y}_2, ..., \hat{y}_l\}$$

The loss in mutual information can be expressed as

(a) a weighted sum of the relative entropies between rows distribution $p(Y|x)$ and "row-lumped" distributions $q(Y|\hat{x})$,

(2.47) $$D_{KL}\left(p(X,Y,\hat{X},\hat{Y}) \middle\| q(X,Y,\hat{X},\hat{Y})\right) = \sum_{\hat{x}} \sum_{x:C_X(x)=\hat{x}} p(x) D_{KL}\left(p(Y|x) \middle\| q(Y|\hat{x})\right)$$

(b) a weighted sum of the relative entropies between rows distribution $p(X|y)$ and "column-lumped" distributions $q(X|\hat{y})$,

(2.48) $$D_{KL}\left(p(X,Y,\hat{X},\hat{Y}) \middle\| q(X,Y,\hat{X},\hat{Y})\right) = \sum_{\hat{y}} \sum_{y:C_Y(y)=\hat{y}} p(y) D_{KL}\left(p(X|y) \middle\| q(X|\hat{y})\right)$$

Using Lemma 1, the co-clustering can be found by assigning a row x to the new row-cluster $C_X^{(t+1)}(x)$ and assigning column y to a new column-cluster $C_Y^{(t+1)}(y)$ as follows:

(2.49) $$C_X^{(t+1)}(x) = \underset{\hat{x}}{\operatorname{argmin}}\, D\left(p(Y|x) \,\|\, q^{(t)}(Y|\hat{x})\right)$$

(2.50) $$C_Y^{(t+1)}(y) = \underset{\hat{y}}{\operatorname{argmin}}\, D\left(p(X|y) \,\|\, q^{(t)}(X|\hat{y})\right)$$

The co-clustering algorithm in (Dhillon et al. 2003) works by assigning each row and each column to the cluster that minimizes the KL distance for rows and columns assignments. The algorithm starts with some (usually random) initial co-clustering $(C_X^{(0)}, C_Y^{(0)})$ and some $q^{(0)}$ and then iterates these assignments, monotonically decreasing the value of the objective function. It stops when the change in the objective is smaller than a given threshold. Unfortunately, the algorithm only finds a local minimum, which is influenced by the starting condition. Also, it may happen that one of the row (or column) cluster becomes empty, at some iteration of the algorithm. In this case, the algorithm is not able to fill it up anymore, and the cluster is lost: the result will have $(k - 1)$ clusters.

As opposed to the DC (N. Slonim and N. Tishby 2000) and IDC clustering algorithms (El-Yaniv and Souroujon 2001) discussed in the previous sub-section, the ITCC algorithm combines the clustering by intertwining word and document clusterings at all stages and continuously improves both until a local minimum is found. This represents a

true co-clustering approach as opposed to a two-way clustering approach of first finding word clusters followed by document clusters. The ITCC algorithm, however, is prone to getting stuck in a local minima and the end co-clustering result is dependent on the initialization step. As a result, successive runs of the algorithm can yield significantly varying results.

Other Information Theory Based Approaches

A more generalized co-clustering framework is presented in (Banerjee et al. 2004) wherein any Bregman divergences can be used in the objective function, and various conditional expectation based constraints can be incorporated into the framework. The Bregman co-clustering algorithm (Banerjee et al. 2004) associates a co-clustering task with a matrix approximation task whose quality is evaluated by the approximation error.

Bekkerman et al. (Bekkerman, El-Yaniv, and McCallum 2005) proposes an extension of the ITCC algorithm (Dhillon et al. 2003) that simultaneously clusters variables of several types (e.g. documents, words and authors) based on pair-wise interactions between the multiple data matrices. The goal of the algorithm is to aim at maximizing an objective function that measures multiple pair-wise mutual information between cluster variables. We generalize the notation used previously such that $\mathbf{X} = \{\mathrm{X}_i | i = 1, 2, ..., u\}$ be the variables to be clustered and $\hat{\mathbf{X}} = \{\hat{\mathrm{X}}_i | i = 1, 2, ..., u\}$ be their respective clustering, where u is the number of interactions (for co-clustering, u=2). Let **G** = (V,E) be an undirected graph with the vertices V represent the clustering and an edge E_{ij} appears in E if we are interested in maximizing the mutual information between the two clustering. The objective function is then given by

$$\max_{\{\hat{\mathrm{X}}_i\}} \sum_{e_{ij} \in \mathrm{E}} I(\mathrm{X}_i; \mathrm{X}_j) \tag{2.51}$$

The actual clustering algorithm is a mixture among different clustering directions, constructed to locally optimize the objective function. It blends *agglomerative* procedures for some variables and *top-down* for others. Both the routines are based on greedy approaches with end corrective measures, much in the same way as used in the sequential Information Bottleneck (sIB) algorithm (N. Slonim, N. Friedman, and N. Tishby 2002).

Liu and Shah (J. Liu and Shah 2007) have used an information theoretic algorithm for scene modeling to detect intermediate semantic concepts, using a Maximization of Mutual Information (MMI) approach. Information theoretic approaches have also been used in other works closely relating to co-clustering such as Self Taught Clustering by Dai et al. (Dai et al. 2008) who uses *auxiliary data* to co-cluster the target data. The idea behind using this concept is that when the target dataset is small, they are not sufficient to allow effective learning of a high quality feature representation and using auxiliary unlabelled data can increase the clustering results when the target and auxiliary data are simultaneously clustered. Similarly, co-clustering based on information theory has also been proposed, by the same authors (Dai et al. 2007), for the classification of out-of-domain documents as a supervised approach.

Long et al. (Long et al. 2006) have proposed *k*-partite graph learning, which is an extension of a binary

clustering into graphs that can be partitioned into *k* sets. The algorithm is based on Relational Summary Network (RSN), which can be seen as a principled generalized framework for learning on a *k*-partite graph to find hidden structures. The proposed algorithm tries to identify the hidden structure of a *k*-partite graph by constructing a relation summary network to approximate the original *k*-partite graph under a broad range of distortion measures. Similar to the Bregman co-clustering algorithm, it uses Bregman divergences to iteratively find a local optimal RSN which tried to retain the many-to-many relationships across the datasets at the levels of clusters. The many-to-many relationships between the clusters summarize the many-to-many relations between the data samples assigned to them and this is captured by the edge weights between the cluster vertices in the cluster graph.

2.5.4. Matrix Density Based Methods

This group of algorithms tries to find patterns in matrices that satisfy some objective function such as the mean square residue in the sub-matrix. The idea is that meaningful co-clusters will have homogenous data and hence the residue (or any other function) would be minimized within the sub-matrix. First, we introduce some basic concepts used in most algorithms of this category. Given an element A_{ij} in the matrix **A**, we define the following

1. The row average $\mu_{i\hat{y}}$ corresponding to row *i* in the bicluster is given by $\mu_{i\hat{y}} = \frac{1}{|\hat{y}|}\sum_{j\in\hat{y}} A_{ij}$,
2. The column average $\mu_{\hat{x}j}$ corresponding to column *j* in the bicluster is given by $\mu_{\hat{x}j} = \frac{1}{|\hat{x}|}\sum_{i\in\hat{x}} A_{ij}$,
3. The biclustering average $\mu_{\hat{x}\hat{y}}$ corresponding to row *cluster* $\hat{x}$ and column cluster $\hat{y}$ is given by $\mu_{\hat{x}\hat{y}} = \frac{1}{|\hat{x}\hat{y}|}\sum_{i\in\hat{x}, j\in\hat{y}} A_{ij}$.

Direct Clustering of a Data Matrix

The seminal work on co-clustering was based on a density based technique proposed by Hartigan (Hartigan 1972) which he referred to as block clustering. The block clustering algorithm is based on statistical analysis of the sub-matrices that form the biclusters. The block clustering algorithm splits the original data matrix into a set of sub-matrices whose quality is assessed by the measure of variance given by

(2.52) $$VAR(\hat{x},\hat{y}) = \sum_{i\in\hat{x}, j\in\hat{y}} \left|A_{ij} - \mu_{\hat{x}\hat{y}}\right|^2$$

Hartigan defined the "best" bicluster to have zero variance. According to this criterion, the best bicluster will always consist of a single element or have constant values. To overcome this, a minimum number of elements per cluster are fixed and the algorithm tries to find a pre-determined number *k* of biclusters in the data. Perfect biclusters are rare in the real world, but biclusters with lower variance are considered to be better biclusters than those with higher variance. The quality of the computed biclusters is computed using the overall variance of all the *k* clusters

(2.53) $$VAR(\hat{X},\hat{Y}) = \sum_{r=1..k} \sum_{i\in\hat{x}_r, j\in\hat{y}_r} \left| A_{ij} - \mu_{\hat{x}_r\hat{y}_r} \right|^2$$

The block clustering algorithm is capable of finding constant biclusters. Hartigan also mentions that other objective functions maybe used to find difference kind of biclusters with other desirable properties.

Cheng and Church Algorithm

Cheng and Church (C&C) (Y. Cheng and Church 2000) were the first to use biclustering for gene expression data. Cheng and Church's algorithm is based on the concept of mean residue score, denoted here by *H* as a measure of coherence of the rows and columns in the bicluster. A δ-bicluster is defined as $H(\hat{x}_i,\hat{y}_i) < \delta$ for a given bicluster *i* and some δ>0. Thus, the Cheng and Church's algorithm aims to find δ-biclusters whose mean residue score is greater than a predefined value, δ. The measure *H* is given by the following

(2.54) $$H(\hat{x},\hat{y}) = \sum_{i\in\hat{x}, j\in\hat{y}} \frac{(A_{ij} - \mu_{\hat{x}j} - \mu_{i\hat{y}} + \mu_{\hat{x}\hat{y}})^2}{|\hat{x}||\hat{y}|}$$

The measure *H* is known as the mean squared residue.

Using their technique, biclustering is performed by greedily removing rows and columns from the data matrix so as to reduce the overall value of *H*, and this continues till the given value of δ is reached. In the second phase rows and columns are added using the same scoring scheme. This continues as long as the matrix size grows without crossing the threshold. After a bicluster is extracted, the values of the bicluster are replaced by random values and the process is repeated. There are a number of problematic issues associated with their approach including

1. how to ascertain the right value of δ,
2. the possibility of an exponential growth in the number of sub-matrices,
3. the approach of deleting rows and columns from the data matrix (in order to improve δ) can land into a local minima, and
4. the random values replaced in an extracted bicluster could influence the rest of the clustering process.

The Cheng and Church algorithm produces one cluster at a time. Several enhancements to this algorithm have been proposed in the literature. Yang et al. (Y. H Yang et al. 2002), for example, have criticized the Cheng and Church approach because the replaced random numbers can influence further co-clusters and propose generalized the definition of δ-bicluster to cope with missing values and avoid being affected cause by replacing the values of the extracted biclusters by random values. They provide a new algorithm called FLOC (Flexible Overlapped biclustering) (J. Yang et al. 2003) that simultaneously produce the *k* co-clusters allowing for overlaps by introducing the concept of an "occupancy threshold" for rows and columns. Similarly, (Tibshirani et al. 1999) enhanced the block clustering algorithm by Hartigan by adding a backward pruning method.

2.5.5. Other Approaches

Several other approaches have been proposed in the literature for the simultaneous clustering of objects and their features such as those based on a Bayesian framework (Sheng, Moreau, and De Moor 2003), general probabilistic models (Segal, Battle, and Koller 2002; Segal et al. 2001), plaid model (Lazzeroni and Owen 2002), Order preserving sub matrices (Ben-Dor et al. 2003; J. Liu and W. Wang 2003), self organizing maps (Busygin et al. 2002), etc. An excellent survey of commonly used biclustering techniques, with particular emphasis on gene expression analysis, according to the structure of the biclusters extracted can be found in (Madeira and Oliveira 2004) and (Tanay, Sharan, and Shamir 2005). A more recent survey of co-clustering algorithms in general can be found in Busygin et al (Busygin et al. 2008).

Several other techniques have been proposed that exploits the structural relation in a graph to implicitly perform co-clustering. Instead of starting with a simultaneous clustering of both samples (rows) and features (columns), known as hard clustering, these techniques uses relationships defined in terms of similarity score between elements of one dimension to influence the clustering of the other. We shall discuss some of these techniques in the next chapter (section 3.6) where we also provide a comparison between these techniques and our proposed co-similarity measure.

2.6. Conclusion of the Chapter

In the first part of this chapter, we saw the basic concept of clustering as the grouping of "similar" objects. We introduced the vocabulary and the notations utilized in the rest of the chapter. We further discussed the different types of data representation usually found in the clustering literature. The first kind of data representation observed is the popular Vector Space Model where elements are represented as row vectors and features form column vectors. A second approach to represent such data is based upon using a bi-partite representation. We consider a collection of texts as a bi-partite graph where one set of nodes represent the documents in a text corpus and the other set of nodes represent the words that occur in those documents. A weighted value may be assigned as a link from a document to a word indicating the presence of the word in that document. Different weighting schemes, such as the TF-IDF, may be incorporated to better represent the importance of words in the corpus.

Classical similarity based clustering algorithms use these n-dimensional vector representation to find similarity (or distance) between documents. This approach is simple and compares the vocabulary set of one document relative to the other using some matching criteria. Usually, geometric and probabilistic criteria are used to judge the similarity between objects. The downside of this, however, is the sparseness of the data and its high dimensional nature, which can result in the curse of dimensionality. Two alternative approaches have been discussed in this chapter that tries to utilize additional information in the data to minimize the effect of high dimensionality. The first approach exploits the semantic and structural relation in the data while the second approach explicitly clusters the feature set to form sub-spaces that might be more relevant to sub-sets of the objects.

The structural based approach brings features close together with other features with which they share semantic relationship in relative terms. This can be seen as a soft-approach to defining subspaces in the global feature space.

Algorithms exploiting semantic relationships in the data form complex hidden relationships in the semantic space and exploit the additional information obtained to help in the clustering task. Dimensions are not explicitly reduced but by having relationships between the feature set, it is possible to avoid the curse of dimensionality. The advantage of using this kind of approach is that, unlike the co-clustering based algorithms, it is not necessary to know (and provide to the algorithm) the number of column clusters which is usually not so evident. These classes of algorithms, however, are usually harder to interpret in terms of the results since a clear picture of the underlying process is not evident. Semantic based algorithms discussed in this chapter also tend to take a global approach when exploiting these structural relationships in the data i.e. the structure is examined for the full feature set and does not explicitly take localized relation into account, as is the case of co-clustering.

Co-clustering algorithms form a hard partitioning of both the feature space and the samples space and simultaneously cluster both the dimensions thus explicitly reducing the dimension space and search for clustering in the subspaces. By doing so, co-clustering algorithms are able to take advantage of the clustering of one space to cluster the other. Co-clustering approaches are desirable when we need to identify a set of samples related to a set of features like in gene expression data where we need to identify a set of genes that show similar behavior under a set of experiments. Even when a feature clustering is not required, using co-clustering algorithms can improve the clustering of the objects, like documents. Clearly, identifying feature sub-spaces can help classify objects for example finding topics can help in the clustering of documents by associating each document with a topic. This also makes the interpretation of results easier since we can intuitively match documents with topics. Specifying the number of feature clusters, however, is usually not a trivial task. Unlike the semantic based algorithms, co-clustering usually tries to optimize some given objective function but doesn't explicitly take into account finer semantic relationships.

In the next chapter, we present a novel algorithm that generates similarities between elements of one dimension but by taking similarities of the other dimension into consideration. Thus, the proposed algorithm not only takes advantage of the structural relationship that exists in the data, but by embedding the similarities of one dimension when calculating the other, it also takes into accounts a more "localized" concept to these structural relationships.

Chapter 3
Co-similarity Based Co-clustering

In this chapter, we propose a new similarity measure, known as χ-Sim, which exploits the duality between objects and features. Our aim is to propose a method for the comparison of objects that incorporates the basic ideas of semantic similarity, which can be used with any popular clustering algorithm that uses a proximity matrix as an input. As opposed to traditional similarity measures, the method we propose takes into account the structure of both sets of nodes in a bipartite graph, thus, incorporating information similar to co-clustering while calculating similarity measures. We start by presenting some basic definitions that also serve as a motivation for our work followed by the presentation of our proposed algorithm. As a second contribution, a theoretical study is also performed to get an intuition into the inner workings of the algorithm. We believe this explanation and the related discussion will allow us to better understand the behavior of the algorithm and provide possible directions for future exploration as well as proposing modifications or variants of the algorithm. In the last section of the chapter, we extend the proposed algorithm to perform supervised classification by exploiting category labels from a training set by providing different ways to benefit from such information within the scope of the algorithm.

3.1. Introduction

As discussed in Chapter 2, most traditional similarity measures do not scale well with increase in dimensions and sparseness of data in a high-dimensional space, which makes calculating any statistics (and therefore the corresponding similarity measure) less reliable (Aggarwal, Hinneburg, and Keim 2001). For instance, in a sample text corpus of Table 3-1, documents d_1 and d_2 don't share any common word w_i ($1 \leq i \leq 4$). So, with a classical

similarity measure such as the ones provided by the Minkowski distance or the Cosine measure (Sedding and Kazakov 2004), their similarity equals zero (or is maximal in terms of distance). However, we can observe that both d_1 and d_2 share words with document d_3 meaning that words w^2 and w^3 have some similarities in the documents space. If we can associate w^2 and w^3 together, it is thus possible to associate a similarity value between d_1 and d_2 which will be, of course, smaller than the ones between d_1 and d_3 or d_2 and d_3 but not null. The justification is based on the fact that two documents discussing the same topic may contain many words that are not shared amongst them but may be found in the literature concerning such topic.

Table 3-1 Table showing the interest of co-clustering approach

A	w^1	w^2	w^3	w^4
d_1	2	1	0	0
d_2	0	0	3	4
d_3	0	4	2	0

Alternative algorithms exploiting semantic relationship in the data have been proposed such as those based on nearest neighbors and latent semantic spaces (section 2.4). Most of these algorithms exploit such relationships on a global scale over the whole feature set. Co-clustering algorithms (section 2.5) on the other hand exploit *localized* relationships and have been proposed for even for one-way clustering to improve clustering accuracy. These algorithms usually consider subsets (clusters) of features to exploit semantic relationships between instances. However, the number of feature clusters significantly affects the output of most co-clustering algorithms and this forces the user to provide an additional parameter when only clustering of the samples is required.

Our work is motivated by the work of (Bisson 1992) proposed for comparing similarities of entities in the first order logic. In their work, given two predicates E1 and E2 as follows:

E1: *Father* (Paul, Yves) *sex* (Yves, male) *age* (Yves, 13) *age* (Paul, 33)
E1: *Father* (John, Ann) *sex* (Ann, female) *age* (Ann, 28) *age* (Yves, 58)

Comparing two entities can be seen as a function of the entities with which they co-occur. For example, the similarity between "Paul" and "John" can be extended to the similarity between "Yves" and "Ann", which in turn is calculated on the basis of their sex, age, etc. We use the analogy in a document corpus, such that comparing documents can be seen as a function of comparing their words and vice versa.

In this thesis, we propose a *co-similarity* measure that is based on the concept of weighted distributional semantics (see section 3.2 below) using higher-order co-occurrences. In the case of text analysis, for example, document similarity is calculated based on word similarity, which in turn is calculated on the basis of document similarity. Thus, we use an iterative approach to increase similarity between documents that share similar words and make words that occur in similar together to have higher similarity values. Thanks to our method, it becomes possible to use any classical clustering method (*k*-means, Hierarchical clustering, etc) to co-cluster a set of data. The proposed method is founded on the concept of higher-order co-occurrences based on graph theory and can also be

extended to incorporate prior knowledge from a training dataset for the task of text categorization as we shall see later in this chapter (section 3.7).

3.2. Semantic Similarity, Relatedness, Association and Higher-order Co-occurrences

Before presenting our algorithm, it is necessary to understand a few basic concepts about semantic associations between words, as they form a crucial part behind the motivation of our algorithm.

Semantic similarity Semantic similarity is a concept that holds between lexical items having a similar meaning, such as "*palm*" and "*tree*" (Kolb 2009). It is related to the concept of *synonymy* and *hyponymy* and requires that words can be substituted for each other in context (Geffet and Dagan 2005).

Semantic Relatedness Semantic relatedness refers to words that may be connected by any kind of lexical association and is a much broader concept than semantic similarity. Words such as "*fruit*" and "*leaf*" can be considered to be semantically related (Kolb 2009) as they form a *meronymy* since they can be considered as a part of a more general thing (for instance, a "*tree*").

According to Budanitsky and Hirst (Budanitsky and Hirst 2006), semantic similarity is used when "similar entities such as apple and orange or table and furniture are compared" but that semantic relatedness can also be used for dissimilar things that may be semantically related such as in a is-a relationship like in (*car*, *tires*) , etc. From the clustering point of view, these associations between words are significant. Consider the two sentences - *Joe is a hardworking student* and *Joe is a meticulous pupil*, which more or less means the same thing. In a *bag-of-words* approach using a traditional measure, such as a Cosine measure, these phrases might not be considered as similar. It has been shown Gonzalo et al. (Gonzalo et al. 1998) that if we use the synonymous sets or sense (for example using *WordNet*®[8]), it can improve result in obtaining higher similarity between similar documents for information retrieval[9].

Distributional Semantics Repositories such as WordNet have to be hand-crafted and are not readily available for different languages or other specialized domains. Therefore, a different approach to similarity known as distributional similarity or association is defined as "Two words are associated when they tend to co-occur (for instance "doctor" and "hospital") (Turney 2008). Therefore, using distributional semantics is a more practical approach (in a computational way) of finding such relationships and associations from within a given corpus. Moreover, using distributional semantics help capture the relationship in the given text corpus since it is based on the concept that similar words occur in similar contexts (Harris 1968).

Higher Order Co-Occurrences The concept of 'higher-order' co-occurrences has been investigated (Livesay and Burgess 1998), (Lemaire and Denhière 2006), among many others, as a measure of semantic relationship between

[8] WordNet® is a large lexical database of English. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. (http://wordnet.princeton.edu/)

[9] The goal of information retrieval is to retrieve a sorted list of documents relevant to a query-vector

words. The underlying analogy is that humans do not necessarily use the same vocabulary when writing about the same topic. For instance, Lemaire and Denhière (Lemaire and Denhière 2006) report finding 131 occurrences of the words "*internet*" and 94 occurrences of the word "*web*" but no co-occurrence at all in a corpus of 24-million words collected from the French newspaper *Le Monde*. It is evident, however, that these two words have a strong relationship. This relationship can be brought to light if the two words co-occur with other words in the corpus. For example, consider a document set containing significant number of co-occurrences between the words "*sea*" and "*waves*" and another document set in which the words "*ocean*" and "*waves*" co-occur. We could infer that the words "*ocean*" and "*sea*" are conceptually related even if they do not directly co-occur in any document. Such a relationship between "*waves*" and "*ocean*" (or "*sea"* and "*waves"*) is termed as a first-order co-occurrence. This conceptual association between "*sea*" and "*ocean"* is called a second-order relationship. The concept can be generalized to higher (3rd, 4th, 5th, etc) order co-occurrences.

Semantic and higher order relationship have been used in automatic word sense disambiguation (Schütze 1998), improving stemming algorithms (J. Xu and Croft 1998), in text categorization (Chakraborti, Wiratunga, et al. 2007), etc. As opposed to semantic similarity and relatedness, distributional similarity can be estimated from a given text corpus. Words that statistically co-occur are thought to capture distributional relatedness while word spaces that are connected via higher-order co-occurrences can be said to capture distributional similarity (Sahlgren 2001). In our algorithm, we incorporate statistical distribution extracted from the corpus to present a new similarity measure, called χ-Sim that is presented in the next section.

3.3. The Proposed Similarity Measure (χ-Sim)

3.3.1. Notation

We will use the classical notations: matrices (in capital letters) and vectors (in small letters) are in bold and all variables are in italic.

Data Matrix Let **A** be the data matrix representing a corpus having m rows (documents) and n columns (words); A_{ij} denotes an element of **A** that contains the information of word j in document i. One possible definition of **A** is the 0/1 encoding which denotes the presence or absence of a word in a given document. Other definitions of **A** are also possible, for example, we can replace the indicator encoding by the number of times words j would occur in document i or the popular TF-IDF measure as discussed in Chapter 2, which in some cases leads to better clustering results (Zhou, X. Zhang, and Hu 2007). Let $E_{ij}, 1 \le i \le m$ and $1 \le j \le n$ denote the weight assigned to the occurrence of word j in document i. We consider here the generic case which is given by:

$$A_{ij} = \begin{cases} E_{ij} & if\ word\ \ j\ occurs\ in\ document\ i \\ 0 & otherwise \end{cases} \tag{3.1}$$

Using standard notations, we define $\mathbf{a}_i = [A_{i1} \ldots A_{in}]$ is the row vector representing the document i in the document set and $\mathbf{a}^j = [A_{1j} \ldots A_{mj}]$ is the column vector corresponding to word j. We will refer to a document as d_1, d_2, … when

talking about documents casually and refer to it as $\mathbf{a}_1$, $\mathbf{a}_2$,… when specifying its (row) vector in the matrix **A**. Similarly, we will refer to a word as w_1, w_2,… when talking about words 1, 2,… and use the notation $\mathbf{a}^1$,$\mathbf{a}^2$,… when emphasizing the word vector in the matrix **A**.

Similarity Matrices Let **R** and **C** represent the square and symmetrical row similarity and column similarity matrices of size *m*-by-*m* and *n*-by-*n* respectively with $R_{ij} \in [0,1]$, $1 \leq i,j \leq m$ and $C_{ij} \in [0,1]$, $1 \leq i,j \leq n$. A similarity value of zero corresponds to no similarity while a similarity value of 1 denotes maximal similarity. As before, we define vectors in the similarity matrices $\mathbf{c}_i = [C_{i1} \ldots C_{in}]$ (respectively $\mathbf{r}_i = [R_{j1} \ldots R_{jm}]$) as the similarity vector between the word *i* and all the other words (respectively between the document *j* and the other documents).

Similarity Function We define a function of similarity $f_s(.,.)$ as a generic similarity function that takes two elements A_{ij} and A_{kl} of **A** as input and returns a measure of the similarity $f_s(A_{ij},A_{kl})$ between these two elements. For the sake of brevity, we will also use the shorthand notation $f_s(A_{ij}, \mathbf{a}_k)$ (respectively $f_s(A_{ij}, \mathbf{a}^k)$) to represent a vector that contains the pair-wise similarity $f_s(.,.)$ between the scalar A_{ij} and each element of the vector $\mathbf{a}_k$ (respectively $\mathbf{a}^k$). The function $f_s(.,.)$ can be a multiplication as used in Cosine similarity or any other user-defined function. This makes the approach generic and adaptable to different specialized data types.

3.3.2. Calculating the Co-Similarities Matrices

As mentioned previously, the χ-Sim algorithm is a co-similarity based approach which builds on the idea of iteratively generating the similarity matrices **R** (between documents) and **C** (between words), each of them built on the basis of the other. First, we present an intuitive idea of how to compute the co-similarity matrix **R** between rows, the idea being similarly applicable to word-word similarity matrix **C**.

Usually, the similarity (or distance) measure between two documents d_i and d_j is defined as a function denoted here as $Sim(d_i,d_j)$ that is the sum of the similarities between words occurring in both d_i and d_j given as

$$Sim\left(\mathbf{a}_i,\mathbf{a}_j\right) = f_s\left(A_{i1},A_{j1}\right) + \ldots + f_s\left(A_{in},A_{jn}\right) \tag{3.2}$$

Other factors may be introduced, for instance to normalize the similarity in the interval [0, 1] or an exponent as in the case of Minskowski distance (section 2.2), but the main idea is always the same – we consider the values between columns having the same indices. Now let us suppose we have a matrix **C** whose entries provide a measure of similarity between the columns (words) of the corpus. Equation (3.2) can be re-written as follows without changing its meaning if $C_{ii}=1$:

$$Sim\left(\mathbf{a}_i,\mathbf{a}_j\right) = f_s\left(A_{i1},A_{j1}\right).C_{11} + \ldots + f_s\left(A_{in},A_{jn}\right).C_{nn} \tag{3.3}$$

Here our idea is to generalize equation(3.3) in order to take into account all the possible pairs of features (words) occurring in documents $\mathbf{a}_i$ and $\mathbf{a}_j$. In this way, not only do we *capture* the similarity of their common words but also the similarity coming from words that are not directly shared by these two documents but that are considered to be *similar*. For each pair of words not directly shared by the documents, we take into account their

similarity as provided by the **C** matrix. Thus the overall similarity between documents $\mathbf{a}_i$ and $\mathbf{a}_j$ is defined in equation (3.4) in which the terms in the boxes are those occurring in equation(3.3)

$$\begin{aligned} Sim(\mathbf{a}_i, \mathbf{a}_j) = & \boxed{f_s(A_{i1}, A_{j1}).C_{11}} + f_s(A_{i1}, A_{j2}).C_{12} + \ldots + f_s(A_{i1}, A_{jn}).C_{1n} + \\ & f_s(A_{i2}, A_{j1}).C_{21} + \boxed{f_s(A_{i2}, A_{j2}).C_{22}} + \ldots + f_s(A_{i2}, A_{jn}).C_{2n} + \\ & \bullet\bullet\bullet \\ & f_s(A_{in}, A_{j1}).C_{n1} + f_s(A_{in}, A_{j2}).C_{n2} + \ldots + \boxed{f_s(A_{in}, A_{jn}).C_{nn}} \end{aligned} \tag{3.4}$$

By using our shorthand notation for representing a row vector, we may re-write the above formula as follows

$$Sim(\mathbf{a}_i, \mathbf{a}_j) = f_s(A_{i1}, \mathbf{a}_j) \bullet \mathbf{c}_1 + f_s(A_{i2}, \mathbf{a}_j) \bullet \mathbf{c}_2 + \ldots + f_s(A_{in}, \mathbf{a}_j) \bullet \mathbf{c}_n \tag{3.5}$$

where "•" represents a dot product. Conversely, when we wish to express the similarity between two words (columns) $\mathbf{a}^i$ and $\mathbf{a}^j$ of **A**, we use the same approach. Hence,

$$Sim(\mathbf{a}^i, \mathbf{a}^j) = f_s(A_{i1}, \mathbf{a}^j) \bullet \mathbf{r}_1 + f_s(A_{i2}, \mathbf{a}^j) \bullet \mathbf{r}_2 + \ldots + f_s(A_{in}, \mathbf{a}^j) \bullet \mathbf{r}_n \tag{3.6}$$

In this framework, the similarity between any two documents depends on the similarity between the words appearing in these documents (weighted by the matrix **C**) and reciprocally the similarity of any two words depends on the similarity between the documents in which they occur (weighted by the matrix **R**). This is achieved by the combination of equations (3.5) and (3.6) which exploit the dual relationship between documents and words in a corpus.

Since χ-Sim is a similarity measure and its comparison with other similar measures is an integrated part of this thesis, we make a clear distinction here between two types of indices that contribute to the similarity measure of two documents $\mathbf{a}_1$ and $\mathbf{a}_2$ as given by equation(3.4). Traditional similarity measures such as the Cosine, Minkowski, Euclidean, etc only take into account terms of the form $f_s(A_{i1},A_{j1})C_{11}$, …, $f(A_{in}, A_{jn})C_{nn}$, for some suitable definition of $f_s(,)$, which corresponds to words directly shared by the documents. We refer to the similarity measure contributed by these terms to the overall similarity between $\mathbf{a}_i$ and $\mathbf{a}_j$ as *direct similarity*. The similarity measure contributed by all other terms, which corresponds to comparing words with different indices, is referred to as *induced similarity* because such a similarity is induced based on other direct similarities.

The values obtained by equations (3.5) and (3.6) cannot be directly used as elements R_{ij} and C_{ij} of the matrices **R** and **C** respectively. As stated previously, each element of the matrices **R** and **C** must be normalized to belong to the interval [0, 1], since we define the maximum similarity between two documents (or words) to be unity, but neither $Sim(\mathbf{a}_i, \mathbf{a}_j)$ nor $Sim(\mathbf{a}^i, \mathbf{a}^j)$ verify this property. Therefore, it is necessary to normalize these values. We define two normalization functions $\mathcal{N}^R(,)$ and $\mathcal{N}^C(,)$ — the row and column normalizations functions — that corresponds to the maximum possible value that can be obtained from equations (3.5) and (3.6) respectively. We will denote $\mathcal{N}(,)$ as the generic normalization function - it is evident which form is to be used depending on whether we are normalizing a document pair or a word pair. We will discuss further the similarity function $f_s(,)$ in the next

section and discuss the various normalization schemes in the section 3.3.5.

Now that we have defined (albeit partially) all the different concepts of the approach, we present a naïve version of the algorithm to compute the elements R_{ij} of the matrix **R** and C_{ij} of the matrix **C**. The two equations are given below

(3.7) $$\forall i, j \in 1..m, R_{ij} = \frac{Sim(\mathbf{a}_i, \mathbf{a}_j)}{\mathcal{N}(\mathbf{a}_i, \mathbf{a}_j)}$$

(3.8) $$\forall i, j \in 1..n, C_{ij} = \frac{Sim(\mathbf{a}^i, \mathbf{a}^j)}{\mathcal{N}(\mathbf{a}^i, \mathbf{a}^j)}$$

To compute the matrices **R** and **C** for each pair of documents or words, we use an iterative method. The steps are outlined as follows:

- Similarity matrices **R** and **C** are initialized with the identity matrix **I**, since at the first step and without any further information, only the similarity between a document (respectively word) and itself is considered as maximal. All other values (out of diagonal elements) are initialized with zero. We denote these matrices as $\mathbf{R}^{(0)}$ and $\mathbf{C}^{(0)}$, where the superscripts denote the iteration.
- The new matrix $\mathbf{R}^{(1)}$ between documents, which is based on the similarity matrix between words $\mathbf{C}^{(0)}$ is calculated using equation(3.7).
- Similarly, the new matrix $\mathbf{C}^{(1)}$ between words, which is based on the similarity matrix between documents $\mathbf{R}^{(0)}$ is calculated using equation (3.8).
- This process of updating $\mathbf{R}^{(k)}$ and $\mathbf{C}^{(k)}$ is repeated iteratively for k=1..t iterations.

```
Function χ-Sim
Input       : data matrix A
Output      : two similarity matrices R and C expressing the
              co-similarity between rows and columns of A
Initialize R and C with the identity matrix
for i = 1 to t
    for j=1 to m
      for k=1 to m
            R(i)jk = Sim(aj, ak) / N(aj, ak)
      end for
    end for
    for j=1 to n
      for k=1 to n
            C(i)jk = Sim(aj, ak) / N(aj, ak)
      end for
    end for
end for
```

FIGURE 3.1 The Naïve χ-Sim algorithm

The algorithm is defined in FIGURE 3.1. It is worth emphasizing here that computing the **R** matrix first and then the **C** matrix or the converse doesn't change the way the system of equations is evolving.

Classically, the overall complexity of computing a similarity matrix between documents represented by a matrix **A** of size dimensions D (assuming it is square matrix), is equal to $O(D^3)$ in terms of the number of calls to the function $f_s(.,.)$. However, in a naïve implementation of χ-Sim, the complexity is much higher. The number of comparisons given by one call to equation (3.7) is n^2 as is evident by looking at its expanded form given by equation(3.4). The computation of **R** involves comparing all pairs of documents, or $O(m^2)$ and each of this document pair comparison involves comparing all pair of words or $O(n^2)$. Similarly, the computation of **C** involves a pair-wise comparison of all pair of words or $O(n^2)$, each of which requires $O(m^2)$ comparisons. Thus for a square matrix of dimension D, we have a complexity of $O(D^4)$ in terms of calls to $f_s(,)$. That is clearly too high to cope efficiently with real datasets.

Another question that arises from the algorithm shown in FIGURE 3.1 is the number of times the algorithm needs to be iterated (the parameter t), and whether the algorithm converges towards a fixed point. As shall be seen in section 3.4, where we explain a theoretical interpretation of the algorithm, the value of t has an intuitive meaning in terms of paths in a bipartite graph. In practice, the value of t is relatively small, typically less than 5.

3.3.3. Optimization Techniques

Fortunately, there are two effective ways to deal with the problem of complexity, making the algorithm at the same level of complexity, $O(D^3)$, as the other distance measures (Cosine, Minkowski, etc). In the case where the values in **A** are discrete (i.e. belong to an enumerated type E containing |E| elements), a whole set of calculations for each pair of documents (or words) in the naïve $O(D^4)$ algorithm is repetitive and can be avoided. The second method is based on the definition of the function $f_s(,)$ itself. If the similarity function $f_s(A_{ij}, A_{kl})$ can be defined as a simple product (i.e. $f_s(A_{ij}, A_{kl}) = A_{ij}A_{kl}$), then the system of equations (3.7) and (3.8) used to compute the similarity matrices **R** and **C** can be expressed simply as a product of three matrices without any loss of generality. We will see the two cases individually.

Values Belong to an Enumerated Type

We consider the problem of optimization when the data type in the matrix **A** is of an enumerated type E i.e. $A_{ij} \in \{E\}$ where |E| is small. We show here the principle of optimization between two documents (objects) for computing the similarity value $Sim(\mathbf{a}_i,\mathbf{a}_j)$, the idea being symmetrical when applied to a pair of words(features). We illustrate the optimization with the help of an example as shown in Table 3-2. For the sake of brevity, we define A_{ij} to be of the type Boolean i.e. $A_{ij} \in \{0,1\}$.

Recall from Equation (3.5) that when we compare two documents, say d_i and d_j, each feature (word) of document i is compared with each feature of document j. Note that a given document say d_1 in Table 3-2 must be

compared with all other ($m-1$) documents[10]. Using the shorthand notation for the similarity function $f_s(,)$, one can see that the comparison of a given word w_k in d_i with all elements of d_j is given by $f_s(A_{ik},\mathbf{a}_j)\bullet\mathbf{c}_k$. We make the following two observations

- Firstly, for each feature k of the document d_i, we need to calculate the term $f_s(A_{ik},\mathbf{a}_j)$ with $A_{ik} \in \{E\}$. Therefore, the vector resulting after applying $f_s(A_{ik},\mathbf{a}_j)$ can take only $|E|$ possible values. For the case of Boolean values, for example, we have only two forms $f_s(0,\mathbf{a}_j)$ and $f_s(1,\mathbf{a}_j)$.
- Secondly, for each comparison of a document d_i with d_j, we have to repeat the computation of the scalar $f_s(A_{ik},\mathbf{a}_j)\mathbf{c}_k$ for a given word w_k. As seen previously, there are only $|E|$ vectors resulting from $f_s(A_{ik},\mathbf{a}_j)$ and therefore there exists only $|E|.n$ scalars of the form $f_s(A_{ik},\mathbf{a}_j)\bullet\mathbf{c}_k$.

Table 3-2 Illustration of the optimization principle for an enumerated data type

A	$\mathbf{w^1}$	$\mathbf{w^2}$	$\mathbf{w^3}$	$\mathbf{w^4}$
$\mathbf{d_1}$	1	1	0	0
$\mathbf{d_2}$	0	0	1	1
$\mathbf{d_3}$	1	1	1	0
$\mathbf{d_4}$	0	1	0	1

Imagine now that for a given document d_j, we pre-calculate all the possible scalar values $f_s(A_{ik},\mathbf{a}_j)\bullet\mathbf{c}_k$ for each document *i,* the calculation of the similarity value $Sim(\mathbf{a}_i,\mathbf{a}_j)$ amounts to n possible values (one for each attribute) corresponding to the value A_{ij}. Suppose that we want to calculate the similarity between d_1 and all the other documents. We first pre-calculate the two possible vectors $f_s(0,\mathbf{a}_1)$ and $f_s(1,\mathbf{a}_1)$. The cost of this operation is given by $O(E.n)$. Next, for each of these vectors, we calculate the scalars corresponding to the comparisons resulting from each possible word. There are 8 resulting scalars given by $f(0,\mathbf{a}_j)\bullet\mathbf{c}_1$, $f(0,\mathbf{a}_j)\bullet\mathbf{c}_2$, $f(0,\mathbf{a}_j)\bullet\mathbf{c}_3$, $f(0,\mathbf{a}_j)\bullet\mathbf{c}_4$ and $f(1,\mathbf{a}_j)\bullet\mathbf{c}_1$, $f(1,\mathbf{a}_j)\bullet\mathbf{c}_2$, $f(1,\mathbf{a}_j)\bullet\mathbf{c}_3$, $f(1,\mathbf{a}_j)\bullet\mathbf{c}_4$. Since each vector of the form $f_s(A_{ik},\mathbf{a}_j)$ has a dimension of n and each vector of the form $\mathbf{c}_k$ also has a dimension of n, the total cost of this operation is $O(E.m.n^2)$ for all documents.

Computing the similarity values $Sim(\mathbf{a}_i,\mathbf{a}_j)$ now reduces to performing a sum over a set of pre-computed scalar values. For example, the similarity value $Sim(\mathbf{a}_1,\mathbf{a}_2)$ is given by $f(0,\mathbf{a}_1)\bullet\mathbf{c}_1+f(0,\mathbf{a}_1)\bullet\mathbf{c}_2+$, $f(1,\mathbf{a}_1)\bullet\mathbf{c}_3+f(1,\mathbf{a}_1)\bullet\mathbf{c}_4$ or $O(n)$ operations. Thus, computing similarities between all pair of documents is given by $O(m^2n)$. Therefore, the complexity for calculating the row similarity matrix **R** is given by

$$\text{Complexity}_R = O(I.\max(m^2.n,|E|.m.n^2)) \text{ or in a general case } O(t.|E|.D^3) \text{ for } t \text{ iterations}$$

Similarly, the complexity for calculating the column similarity matrix **C** is given by

[10] In practice, a document d_i is compared with $i-1$ elements since the matrix **R** (respectively **C**) is symmetric.

$\text{Complexity}_C = O(I .\max(n^2.m, |E|.n.m^2))$ or in a general case $O(t.|E|.D^3)$ for t iterations

As noted previously, t is typically small. Therefore, the overhead of this optimization as compared to a classical similarity measure for example the Cosine, is a constant $K \approx 10$ (for t=5 and |E|=2). Hence, the complexity of the algorithm is given by $O(KD^3)$, where K is a constant, which is similar to a classical similarity measure. This optimization is efficient only when the cardinal of E is relatively small. Nevertheless it is interesting since it works for any kind of definition of $f_s(,)$.

Function $f(,)$ as the Scalar Product

The second optimization possibility is when we define the function $f_s(,)$ as a product of elements, given by $f_s(A_{ij},A_{kl})=A_{ij}.A_{kl}$ which is the same as in traditional similarity measures such as Cosine, Tanimoto, etc. The χ-Sim algorithm can now be expressed as a product of matrices as explained below. Replacing $f_s(,)$ as a product and by using the associative property of multiplication, we have

$$\begin{aligned} Sim(\mathbf{a}_i,\mathbf{a}_j) = &(A_{i1}.C_{11}).A_{j1} + (A_{i1}.C_{12}).A_{j2} + \ldots + (A_{i1}.C_{1n}).A_{jn} + \\ &(A_{i2}.C_{21}).A_{j1} + (A_{i2}.C_{22}).A_{j2} + \ldots + (A_{i2}.C_{2n}).A_{jn} + \\ &\bullet\bullet\bullet \\ &(A_{in}.C_{n1}).A_{j1} + (A_{in}.C_{n2}).A_{j2} + \ldots + (A_{in}.C_{nn}).A_{jn} \end{aligned} \tag{3.9}$$

By collecting terms, we can re-write equation 3.9 as follows

$$\begin{aligned} Sim(\mathbf{a}_i,\mathbf{a}_j) = &\left[(A_{i1}C_{11}) + (A_{i2}C_{21}) + \ldots + (A_{in}C_{n1})\right].A_{j1} + \\ &\left[(A_{i1}C_{12}) + (A_{i2}C_{22}) + \ldots + (A_{in}C_{n2})\right].A_{j2} + \\ &\bullet\bullet\bullet \\ &\left[(A_{i1}C_{1n}) + (A_{i2}C_{2n}) + \ldots + (A_{in}C_{nn})\right].A_{jn} \end{aligned} \tag{3.10}$$

Note that the elements within the squared brackets in equation (3.10) correspond to elements of the matrix (**AC**) of size m-by-n. The similarity measure between document $\mathbf{a}_i$ and $\mathbf{a}_j$ given by $Sim(\mathbf{a}_i,\mathbf{a}_j)$ now corresponds to element (i,j) of the triple matrix multiplication given by $(\mathbf{ACA}^T)_{ij}$ where $\mathbf{A}^T$ is the transpose of the matrix **A**.

In the rest of this thesis, we will consider this definition of $f_s(,)$ (i.e. $f_s(A_{ij},A_{kl}) = A_{ij}.A_{kl}$) for two reasons – firstly, this definition of $f_s(,)$ presents an interesting case since it renders the χ-Sim algorithm to a framework that results in a direct comparison with other classical similarity measures and enables us to elucidate the contribution of our approach. Secondly, defining $f_s(,)$ as a product brings down the calculation of the **R** and **C** similarity matrices as a simple product of matrices. This enables us to further explore the properties of these matrices and explore alternative theoretical insight into the working of the algorithm. Moreover, considering the algorithm as product of matrices can be easily implemented using programming languages such as Matlab® and allow the use of several pre-existing libraries for efficient matrix multiplication in other languages such as Java, etc.

3.3.4. A Generalization of the Similarity Approach

In this section, we discuss the connection between the χ-Sim approach and existing similarity measures. We consider them as special cases of the χ-Sim approach and show that the χ-Sim framework provides a generalized approach to measuring similarity between two entities.

We describe here a generic formulation of several classical similarity measures between two documents. Given two documents $\mathbf{a}_i$ and $\mathbf{a}_j$, the similarity measure can be expressed as a product of matrices given by

$$Similarity(\mathbf{a}_i, \mathbf{a}_j) = \frac{\mathbf{a}_i(\mathbf{C})\mathbf{a}_j^{\mathrm{T}}}{\mathcal{N}(\mathbf{a}_i, \mathbf{a}_j)} \tag{3.11}$$

where $\mathbf{a}^{\mathrm{T}}$ denotes the transpose of $\mathbf{a}$ and $\mathcal{N}(\mathbf{a}_i,\mathbf{a}_j)$ is a normalization function that depends on $\mathbf{a}_i$ and $\mathbf{a}_j$ used to map the similarity function to a particular interval, such as [0,1]. Equation (3.11) can be seen as a generalization of several similarity measures. For example, the Cosine similarity measure can be written using the above equation, where $\mathbf{C}$ is set to the identity matrix, as

$$\mathrm{Cosine}(\mathbf{a}_i, \mathbf{a}_j) = \frac{\mathbf{a}_i(\mathbf{I})\mathbf{a}_j^{\mathrm{T}}}{\|\mathbf{a}_i\|.\|\mathbf{a}_j\|} \tag{3.12}$$

where $\|\mathbf{a}_i\|$ represents the L_2 norm of the vector $\mathbf{a}_i$. Similarly, the Jaccard index can be obtained by setting $\mathbf{C}$ to $\mathbf{I}$ and $\mathcal{N}(\mathbf{a}_i,\mathbf{a}_j)$ to $|\mathbf{a}_i| + |\mathbf{a}_j| - \mathbf{a}_i\mathbf{a}_j^{\mathrm{T}}$ ($|\mathbf{a}_i|$ denotes L_1 norm), while the Dice Coefficient can be obtained by setting the $\mathbf{C}$ to $2\mathbf{I}$ and $\mathcal{N}(\mathbf{a}_i, \mathbf{a}_j)$ to $|\mathbf{a}_i| + |\mathbf{a}_j|$. Note that by setting the $\mathbf{C}$ matrix to identity, we define the similarity between a word and itself to be 1 (maximal) and between every non-identical pair of word to be 0.

The similarity value between documents $\mathbf{a}_i$ and $\mathbf{a}_j$, as expressed in equation(3.5), can be expressed in the form of equation(3.12). Here the numerator from equation (3.12) corresponds to the terms in equation (3.5) and the denominator corresponds to some normalization factor $\mathcal{N}(\mathbf{a}_i,\mathbf{a}_j)$ as described previously. We can now re-write how to compute the elements $R^{(k)}_{ij}$ $(1 \le i,j \le m)$ of $\mathbf{R}^{(k)}$ at a given iteration k as,

$$\forall i, j \in 1..m, R_{ij}^{(k)} = \frac{\mathbf{a}_i(\mathbf{C}^{(k-1)})\mathbf{a}_j^{\mathrm{T}}}{\mathcal{N}(\mathbf{a}_i, \mathbf{a}_j)} \tag{3.13}$$

Similarly, the elements $C^{(k)}_{ij}$ $(1 \le i,j \le n)$ at iteration k of the word similarity matrix $\mathbf{C}^{(k)}$ can be computed as

$$\forall i, j \in 1..n, C_{ij}^{(k)} = \frac{\mathbf{a}^i(\mathbf{R}^{(k-1)})\mathbf{a}^j}{\mathcal{N}(\mathbf{a}^i, \mathbf{a}^j)} \tag{3.14}$$

We now move on to define a normalization factor for equations (3.13) and (3.14) for normalizing $R^{(k)}_{ij}$ and $C^{(k)}_{ij}$ respectively. However, we first need to clearly understand the difference between the similarity values generated by χ-Sim measure and the other classical similarity measures mentioned above. As opposed to the other similarity

measures like the Cosine measure in equation (3.12), the similarity values in equations (3.13) and (3.14) differ in two aspects: firstly, the non-diagonal elements of the **C** matrix are not zero; and secondly, the values in the **C** matrix are defined as a function of another matrix **R** and the two are iteratively computed.

3.3.5. Normalization

Recall from the discussion in section 3.3.2 that neither of the functions $Sim(\mathbf{a}_i,\mathbf{a}_j)$ or $Sim(\mathbf{a}^i,\mathbf{a}^j)$ guarantee that the similarity value belongs to the interval [0,1]. We explore two possible ways to normalize the similarity matrix.

In the first case, we consider the normalization as a function of the similarity matrices **R** and **C**. Since the values of the similarity matrices are evolving at each iteration k ($1 \le k \le t$), the values $\boldsymbol{R}^{(k)}_{ij}$ and $\boldsymbol{C}^{(k}_{ij}$ are normalized as a function of matrices $\boldsymbol{R}^{(k)}$ and $\boldsymbol{C}^{(k)}$ respectively, given by

$$R_{ij}^{(k)} = R_{ij}^{(k)} / (\sum_{i=1..m} \sum_{j=1..m} R_{ij}^{(k)}) \tag{3.15}$$

and

$$C_{ij}^{(k)} = C_{ij}^{(k)} / (\sum_{i=1..n} \sum_{j=1..n} C_{ij}^{(k)}) \tag{3.16}$$

Note that equations (3.15) and (3.16) do not guarantee that the similarity of a document (or word) with itself is 1. Alternately, we could consider normalization based on the local document similarity distribution i.e. normalize each element of $\mathbf{R}^{(k)}$ and $\mathbf{C}^{(k)}$ on a vector basis. Thus,

$$R_{ij}^{(k)} = R_{ij}^{(k)} / (\sum_{j=1..m} R_{ij}^{(k)}) \tag{3.17}$$

and

$$C_{ij}^{(k)} = C_{ij}^{(k)} / (\sum_{j=1..n} C_{ij}^{(k)}) \tag{3.18}$$

As before, the similarity values between a document (or word) and itself is not maximal. In both cases, one can overcome this by forcing the diagonal value of $\mathbf{R}^{(k)}$ and $\mathbf{C}^{(k)}$ to be 1 and excluding the values from the denominator for $R^{(k)}_{ij}$ (respectively $C^{(k)}_{ij}$) for $i \neq j$. However the resulting normalized similarity matrix is not symmetric since $R^{(k)}_{ij}$ is normalized relative to the similarity distribution of $\mathbf{a}_i$ while $R^{(k)}_{ji}$ is normalized with respect to that of $\mathbf{a}_j$.

A more general drawback of using the above normalization methods, however, is that they do not take into account the length of the vectors. This is particularly important in the case of text clustering since text documents can vary significantly in size. In fact, considering the numerator of equation(3.11), it is clear that larger document pairs can lead to higher values even if they share a relatively smaller percentage of words than smaller document pairs sharing a higher percentage of words. Similarly, more frequent word pairs will have a higher similarity values and the effect is magnified when **R** and **C** are iteratively computed. Therefore, when comparing document pairs of unequal sizes, we need to take into account the greater probability that words will co-occur in larger documents to

avoid undue bias towards such documents.

A second approach focuses on the document (or word) vectors themselves to take their length into account as mentioned above. The Euclidean or L_2 Norm is an examples of such a normalization. The L_2 norm is defined as

(3.19) $$\mu_i = \sqrt{\sum_{k=1}^{n} (A_{ik})^2} \quad \text{and} \quad \mu^i = \sqrt{\sum_{k=1}^{m} (A_{ki})^2} \qquad \text{with} \qquad \mathcal{N}(\mathbf{a}_i, \mathbf{a}_j) = \mu_i \mu_j$$

and guarantees that the similarity of a document (respectively word) with itself at the first iteration of the algorithm is 1. In fact, using the normalization of equation (3.19) in equation (3.11) is equivalent to the Cosine similarity value for the first iteration, since only the direct similarity value is considered (C_{ij}=0 $\forall i \neq j$). After the first iteration, however, the resulting similarities values obtain when using equation (3.11) may be greater than 1. Therefore, even by setting the diagonal elements to 1 at each iteration, one cannot that the similarity value between an element and itself will be maximal.

Another possible normalization used under this approach is called the L_1 normalization. By definition, the maximum similarity value between two pair of words C_{ij} is 1. Therefore, it follows from equation (3.10) that the upper bound of $Sim(\mathbf{a}_i,\mathbf{a}_j)$ ($1 \leq i,j \leq m$) is given by the product of the sum of elements of $\mathbf{a}_i$ and $\mathbf{a}_j$. If we denote the sum of vectors $\mathbf{a}_i$ and $\mathbf{a}_j$ by μ_i and μ_j respectively, then the normalization function when comparing two documents is defined as

(3.20) $$\mathcal{N}(\mathbf{a}_i, \mathbf{a}_j) = \mu_i \mu_j \qquad \text{where } \mu_i = \sum_k A_{ik} \text{ and } \mu_j = \sum_k A_{jk}$$

Similarly, the normalization factor when comparing two pair of words $Sim(\mathbf{a}^i,\mathbf{a}^j)$ is given by

(3.21) $$\mathcal{N}(\mathbf{a}^i, \mathbf{a}^j) = \mu^i \mu^j \qquad \text{where } \mu^i = \sum_k A_{ki} \text{ and } \mu^j = \sum_k A_{kj}$$

This approach is particularly suited for textual datasets since it allows us to take into consideration the actual length of the document and word vectors when dealing between pairs of documents or words of uneven length, which is typically the case. Considering the rows and columns of **A** as components of a vector, the L_1 norm described above represents the length of a Manhattan walk along the components of the vector. Note that using the L1 normalization guarantees the similarity values between any pair of documents $\mathbf{a}_i$ and $\mathbf{a}_j$ or any pair of words $\mathbf{a}^i$ and $\mathbf{a}^j$ to lie in the interval [0,1], but does not satisfy that the similarity between a document (or word) and itself will be 1. As previously, we could force the elements of the diagonal to be 1 (by definition) and calculate only similarity between different pair of documents (or words).

3.3.6. The χ-Sim Similarity Measure

Now that we have defined the function $f_s(,)$ and the normalization function $\mathcal{N}(,)$, we proceed to formally define the χ-Sim algorithm in this section as a product of matrices. Equations (3.13) and (3.14) allow us to compute the similarities between two documents and between two words. The extension over all pair of documents and all pair of words can be generalized as a matrix multiplication. The algorithm follows:

1. We initialize the similarity matrices **R** (documents) and **C** (words) with the identity matrix **I**.
2. At each iteration k, we calculate the new similarity matrix between documents $\mathbf{R}^{(k)}$ by using the similarity matrix between words $\mathbf{C}^{(k-1)}$. To normalize the values, we take the Hadamard[11] product between the matrix $\mathbf{R}^{(k)}$ and a pre-calculated matrix **NR** defined as $\forall i,j \in [1,m]$, $\mathrm{NR}_{ij} = 1/\mathcal{N}(\mathbf{a}_i,\mathbf{a}_j)$. We do the same thing for the similarity matrix $\mathbf{C}^{(k)}$ and normalize it using **NC** given by $\forall i,j \in [1,n]$, $\mathrm{NC}_{ij} = 1/\mathcal{N}(\mathbf{a}^i,\mathbf{a}^j)$. The two relations are as follows:

(3.22) $$R_{ij}^{(k)} = \begin{cases} 1 & if\ i = j \\ \left[(\mathbf{A}\mathbf{C}^{(k-1)}\mathbf{A}^{\mathbf{T}}) \otimes \mathbf{NR} \right]_{ij} & otherwise \end{cases}$$

(3.23) $$C_{ij}^{(k)} = \begin{cases} 1 & if\ i = j \\ \left[(\mathbf{A}^{\mathbf{T}}\mathbf{R}^{(k-1)})\mathbf{A} \otimes \mathbf{NC} \right]_{ij} & otherwise \end{cases}$$

where '⊗' denotes the Hadamard multiplication, $\mathbf{NR}_{ij} = \frac{1}{\mu_i \mu_j}$ and $\mathbf{NC}_{ij} = \frac{1}{\mu^i \mu^j}$

3. Step 2 is repeated t times to iteratively update $\mathbf{R}^{(k)}$ and $\mathbf{C}^{(k)}$

In practice, equation (3.22) may be obtained by taking the matrix multiplication $\mathbf{R}^{(t)}=\mathbf{A}\mathbf{C}^{(t-1)}\mathbf{A}^{\mathrm{T}} \otimes \mathbf{NR}$ and then setting the diagonal to 1. Equation (3.23) may be similarly obtained. To update **R** (or **C**) we just have to multiply three matrices using equations (3.22) and (3.23) respectively. Given that the complexity of matrix multiplication[12] is in $O(D^3)$ (for a generalized matrix of size D by D), the overall complexity of χ-Sim is given by $O(tD^3)$ where t is the number of iterations.

Alternately, we could embed the normalization factor into the data matrix itself. For this, we need to define two variants of the original matrix **A** — one normalized by the row, $\mathbf{A}^{\mathrm{R}}$; and one normalized by the column, $\mathbf{A}^{\mathrm{C}}$. The two matrices are given by,

(3.24) $$A_{ij}^{\mathrm{R}} = \frac{A_{ij}}{\mu_i} \text{ and } A_{ij}^{\mathrm{C}} = \frac{A_{ij}}{\mu^j}$$

for some definition of μ_i and μ^j as given in section 3.3.5. The two equations to update the document and word similarity matrices can now be expressed as

(3.25) $$\mathbf{R}^{(i)} = \left[\mathbf{A}^{\mathrm{R}} \mathbf{C}^{(i-1)} \left(\mathbf{A}^{\mathrm{R}} \right)^{\mathrm{T}} \right]_{jj=1}$$

(3.26) $$\mathbf{C}^{(i)} = \left[\left(\mathbf{A}^{\mathrm{C}} \right)^{\mathrm{T}} \mathbf{R}^{(i-1)} \mathbf{A}^{\mathrm{C}} \right]_{jj=1}$$

[11] In a Hadamard product $\mathbf{A}=\mathbf{B}\otimes\mathbf{C}$, the elements A_{ik} of matrix **A** are defined as: $A_{ik}= B_{ik}.C_{ik}$
[12] The complexity of the Hadamard product is $O(D^2)$

where $[\ldots]_{ij}$ denotes setting the diagonal of the resulting matrix to 1.

FIGURE 3.2 shows the matrix representation of χ-Sim when using equations (3.25) and(3.26), explicitly showing the matrix dimensions. Note that using this form of the algorithm means that the (normalized) data matrix **A**, have to be stored twice, one matrix for each of the two normalizations. However, there is a small saving in the computation time as we do not have to normalize at every iteration. More precisely, the Hadamard product between the matrices **R** and **NR** and **C** and **NC** is not done at every iteration.

Note that we may chose not to force the entries in the diagonal of the similarity matrices to 1.This, however, has an adverse effect on the overall evolution of the similarity values. The diagonal values correspond to similarity for direct sharing of words between documents (and vice versa). Since, we use the L1 normalization and do not guarantee that the similarity between a document (or between a word) and itself with be unity, the diagonal values tends to decrease at each iteration. This means that direct co-occurrences (shared words or documents) contribute less towards the similarity measure at each subsequent iteration Moreover, the L1 normalization also does not gurantee that self simialarity between documents or words are maximal. This can lead to values where similarity between two pair of distinct words may be more than the similarity between each of those words with itself.

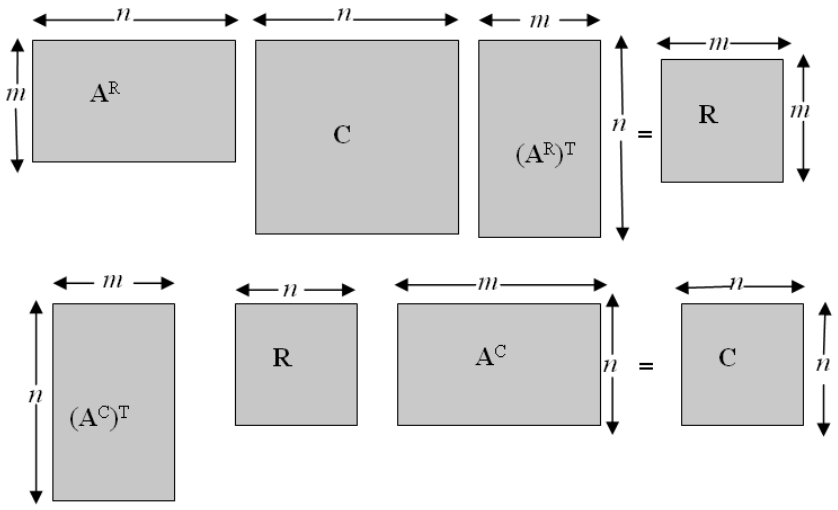


FIGURE 3.2 Representation of the matrix multiplications involved in χ-Sim Algorithm

3.3.7. An Illustrative Example

To illustrate the behavior of the algorithm, we take back the example described in section 2.4.1. The example contains the titles of 9 articles that can be categorized into two main types—d_1 through d_5 that describes computer science and d_6 through d_9 that are related to graph theory. The titles are described by a set of keywords and we reproduce the document by term matrix in Table 3-3.

There are two natural clustering of the data as shown by the dotted lines in Table 3-3. The two clustering are nearly perfect except for a word '*survey'* which is shared by d_2 from the first cluster and d_9 from the second cluster.

Table 3-3 The document-word co-occurrence matrix corresponding to the sentences

	Human	Interface	Computer	Survey	User	System	Response	Time	EPS	Trees	Graph	Minors
d_1	1	1	1	0	0	0	0	0	0	0	0	0
d_2	0	0	1	1	1	1	1	1	0	0	0	0
d_3	0	1	0	0	1	1	0	0	1	0	0	0
d_4	1	0	0	0	0	1	0	0	1	0	0	0
d_5	0	0	0	0	1	0	1	1	0	0	0	0
d_6	0	0	0	0	0	0	0	0	0	1	0	0
d_7	0	0	0	0	0	0	0	0	0	1	1	0
d_8	0	0	0	0	0	0	0	0	0	1	1	1
d_9	0	0	0	1	0	0	0	0	0	0	1	1

Table 3-4 The document similarity matrix at iteration t=1

$\mathbf{R^{(1)}}$	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8	d_9
d_1	1,00	-	-	-	-	-	-	-	-
d_2	0,06	1,00	-	-	-	-	-	-	-
d_3	0,08	0,08	1,00	-	-	-	-	-	-
d_4	0,08	0,08	0,19	1,00	-	-	-	-	-
d_5	0,00	0,17	0,08	0,00	1,00	-	-	-	-
d_6	0,00	0,00	0,00	0,00	0,00	1,00	-	-	-
d_7	0,00	0,00	0,00	0,00	0,00	0,50	1,00	-	-
d_8	0,00	0,00	0,00	0,00	0,00	0,33	0,33	1,00	-
d_9	0,00	0,06	0,00	0,00	0,00	0,00	0,17	0,22	1,00

Table 3-5 The word similarity matrix at iteration t=2

$\mathbf{C^{(1)}}$	Human	Interface	Computer	Survey	User	System	Response	Time	EPS	Trees	Graph	Minors
Human	1,00	-	-	-	-	-	-	-	-	-	-	-
Interface	0,25	1,00	-	-	-	-	-	-	-	-	-	-
Computer	0,25	0,25	1,00	-	-	-	-	-	-	-	-	-
Survey	0,00	0,17	0,17	1,00	-	-	-	-	-	-	-	-
User	0,25	0,13	0,13	0,17	1,00	-	-	-	-	-	-	-
System	0,00	0,00	0,25	0,33	0,13	1,00	-	-	-	-	-	-
Response	0,00	0,00	0,25	0,33	0,13	0,50	1,00	-	-	-	-	-
Time	0,25	0,25	0,00	0,17	0,38	0,00	0,00	1,00	-	-	-	-
EPS	0,00	0,00	0,25	0,17	0,13	0,25	0,25	0,00	1,00	-	-	-
Trees	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	1,00	-	-
Graph	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,17	0,22	1,00	-
Minors	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,25	0,17	0,33	1,00

Table 3-6 The document similarity matrix at iteration t=2

$\mathbf{R}^{(2)}$	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8	d_9
d_1	1,00	-	-	-	-	-	-	-	-
d_2	0,17	1,00	-	-	-	-	-	-	-
d_3	0,24	0,20	1,00	-	-	-	-	-	-
d_4	0,25	0,18	0,37	1,00	-	-	-	-	-
d_5	0,09	0,39	0,20	0,08	1,00	-	-	-	-
d_6	0,00	0,00	0,00	0,00	0,00	1,00	-	-	-
d_7	0,00	0,01	0,00	0,00	0,00	0,61	1,00	-	-
d_8	0,00	0,02	0,00	0,00	0,00	0,46	0,49	1,00	-
d_9	0,03	0,14	0,02	0,02	0,07	0,13	0,31	0,39	1,00

Table 3-7 A comparison of similarity values for different pairs of documents using χ-Sim and Cosine.

Document pair type	χ-Sim at iteration 1	Cosine similarity	χ-Sim at iteration 2
Same cluster with no shared words	$Sim(d_1,d_5) = 0.00$	$Cosine(d_1,d_5) = 0.00$	$Sim(d_1,d_5) = 0.09$
Same cluster with shared words	$Sim(d_7,d_9) = 0.17$	$Cosine(d_7,d_9) = 0.32$	$Sim(d_7,d_9) = 0.31$
Different clusters with no shared type	$Sim(d_2,d_7) = 0.00$	$Cosine(d_2,d_7) = 0.00$	$Sim(d_2,d_7) = 0.01$

We run the χ-Sim algorithm on the data given in Table 3-3 using equations (3.25) and (3.26) iteratively and the result of the first iteration on the document similarity matrix, **R** is given by Table 3-4 below.

The matrix now contains the similarity values between documents that are calculated purely on the basis of their shared word occurrence. For example, the documents d_2 and d_7 have no common words and the similarity between d_2 and d_7 at iteration 1 is zero. Similarly, d_1 and d_5, even though they belong to the same cluster, have a similarity value of zero since they do not have any word that co-occurs between them. This behavior is similar to what we would observe when using a different similarity measure such as the Cosine similarity.

However at the second iteration, both (d_1,d_5) and (d_2,d_7) get a similarity value which although small, is greater than zero. This similarity comes from the fact that documents d_5 share the word '*user*, '*response'*, and '*time'* with document d_2, which also contains the words '*computer*'. As result the word 'computer' gets a similarity value, albeit small, with each of '*user'*, '*response'* and '*time'* given in the similarity matrix $\mathbf{C}^{(1)}$ as shown in Table 3-5. Now at $\mathbf{R}^{(2)}$, when we compare the documents d_1 and d_5 again, they show a small similarity value since they now share some similar word. The similarity value also comes via document d_3 which generates some similarity between the word '*interface'* and the words '*user'* and '*system'*. This results in a non-zero similarity value between the document d_1 and d_5 at the second iteration.

Similarly, document d_7 and d_8 share the words '*trees'* and '*graph'*. This generates a similarity value between the words '*tree'* and '*graph'*. At the first iteration, the similarity value between the documents d_7 and d_9 was 0.17 as a result of a shared word, '*graph'*. However at the second iteration, the similarity value coming from similar words 'tree' and 'graph' generates an additional *induced similarity* measure between the documents d_7 and d_9. Therefore, even though documents d_7 and d_9 shared a common word, their similarity in iteration 2 increases since now they are

also thought to be sharing *similar words* in addition to some common words.

Finally, document d_2 and d_7 which belong to different clusters also share some similarity since d_2 shares the word '*survey'* with d_9 and d_9 shares the word '*graph'* with d_7. Thus, at the first iteration, a similarity between the words '*survey'* and '*graph'* is generated and this similarity is used at the second iteration to compare the documents d_2 and d_7.

A comparison of the similarity measure generated by χ-Sim and the Cosine similarity measures for different pair of documents is given in Table 3-7. Documents d_1 and d_5 belong to the same cluster but do not share any common word. Documents d_7 and d_9 also belong to the same cluster but they do share a common word while documents d_2 and d_7 do not belong to the same cluster. As seen from Table 3-7, the Cosine similarity[13] measure assigns a zero similarity between d_1 and d_5 and between d_2 and d_7. Thus, using the Cosine similarity measure, it is not possible to differentiate a pair whose documents belong to the same document cluster (but does not share any word) to a pair whose documents belong to different clusters.

At the first iteration of χ-Sim, we observe a similar behavior and the similarity between (d_1,d_5) and (d_2,d_7) is zero. This is because we initialize the word similarity matrix, $\mathbf{C}^{(0)}$, as an identity matrix and therefore are in the same framework as any classical similarity measure. However at the second iteration, we use similarity values coming from $\mathbf{C}^{(1)}$ and generate similarity values greater than zero for both (d_1,d_5) and (d_2,d_7). Similarly, the similarity value between d_7 and d_9 is increased. Hence, at each iteration, some new similarities are induced between objects that are not directly connected. Each induced similarity can be thought of as either strengthening an existing link or introducing a new bridge between otherwise unrelated documents (or less similar documents).

It is important to note here that documents d_2 and d_7 belong to different document clusters but still have been assigned a non-zero similarity value. However, this similarity value is significantly smaller (relatively speaking) to the similarity value of document d_2 with each of documents d_1,d_3,d_4 and d_5. Similarly, the similarity values between document d_7 and each of d_6-d_9 is significantly higher (relatively speaking) than document d_2. Nonetheless, it should be noted that the possibility exists that such (relatively) small similarity measures, if coming from numerous sources may add up to distort the similarity ranking between documents, particularly when the clusters are no so well separated. Such a similarity value can be described as '*noise*' since it associates a similarity value between objects pairs (either words or documents) that do not belong to the same cluster. We shall further discuss this in section 3.5.5.

In the next section, we define what we have observed in this example in a more formal method with foundations in graph theory. The theoretical explanation will provides us a intuitive way to reason the functioning of the algorithm and enable us suggest ways to both incorporate prior knowledge into the method (for supervised classification) and explore ways to reduce the effect of noise in the algorithm.

[13] Similar behavior is observed for other similarity/distance measures such as the Hamming distance, Euclidean distance, etc.

3.4. Theoretical Background

We now present a graph theoretical interpretation of the algorithm which would enable us to better understand the working of the algorithm. In the rest of this section, we will be using the concept of graphs and paths (or walks) in a graph, so we start by a few definitions.

Recall from section 2.1.2 of chapter 2 that a *bipartite graph* G={X,Y,E) is a function mapping pairs of X and Y. X and Y are finite collection of elements, enumerated $x_1,\ldots,x_m$ and $y_1,\ldots,y_n$. E is a set of weighted edges defining these mappings. G is called an *undirected graph* if the values in E are symmetric and a *directed graph* otherwise. Additionally, the mapping given by G enforces that there cannot exist a mapping between a pair in X or a pair in Y. Such a graph can be represented by a matrix as in our case, where the rows (documents) form one set of the vertices, the columns (words) forms the other set of the vertices and elements A_{ij} forms an edge between x_i and y_j. A *path* (or *walk*) of length p in G is a sequence of nodes $x_{i_p},\ldots,x_{i_{p+1}}$ (respectively $y_{i_p},\ldots,y_{i_{p+1}}$). The path is called a *circuit* if $i_1=i_{p+1}$ and is called a *simple path* if all indices i_k are distinct for $k=1,\ldots,p$. It is called a *loop* if the circuit has length 2. A loop is defined as $x_i \rightarrow y_j \rightarrow x_i$ (respectively $y_i \rightarrow x_j \rightarrow y_i$).

Consider the bi-partite graph representation of a sample data matrix in FIGURE 3.3 (a) having 6 documents d_1-d_6 and 6 words w_1-w_6. The documents and words are represented by rectangular and oval nodes respectively and an edge between a document d_i and a word w_j in the graph corresponds to the entry A_{ij} in the document-term matrix **A**. In the following explanation, we omit the normalization factor for the sake of clarity which we will re-introduce later.

There is only one order-1 path between documents d_1 and d_2 given by $d_1 \rightarrow w_2 \rightarrow d_2$. If we define the measure of similarity between d_1 and d_2, represented by our similarity matrix $R^{(1)}{}_{12}$ where the superscript represent an order-1 walk, as a dot product of words contained by d_1 and d_2, then the similarity value R_{12} is given by the product $A_{12}A_{22}$, using $f_s(A_{ij},A_{kl})= A_{ij}A_{kl}$. This the same value as obtained by the element $(\mathbf{AA}^T)_{12}$. Note that since the **C** matrix is initialized as identity, at the first iteration, R_{12} just corresponds to the dot product between the corresponding document vectors $\mathbf{a}_1$ and $\mathbf{a}_2$ (since $C_{kl}=0$ for all $k \neq l$) as given by equation (3.11). Similarly, there is only one order-1 path between the documents d_1 and d_3 given by $d_1 \rightarrow w_3 \rightarrow d_3$, and the corresponding similarity value is given by $A_{13}A_{33}$. This is also the same value given by $(\mathbf{AA}^T)_{13}$. The matrix $\mathbf{R}^{(1)} = \mathbf{AA}^T$ thus represents all order-1 paths between the pair of documents $\mathbf{a}_i$ and $\mathbf{a}_j$ ($i,j = 1..m$).

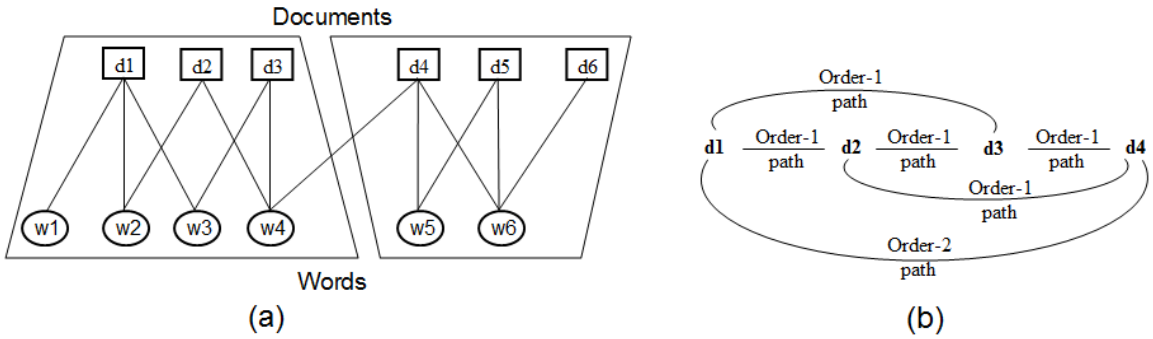


FIGURE 3.3 (a) A bi-partite graph view of the matrix **A**. The square vertices represent documents and the rounded vertices represent words, and (b) some of the higher order co-occurrences between documents in the bi-partite graph.

The same notion can be applied when comparing similarity values between words. Words w_2 and w_4 have only one order-1 path between them given by $w_2 \rightarrow d_2 \rightarrow w_4$. This value corresponds to the element (i,j) of the matrix multiplication of $\mathbf{A}^T$ with $\mathbf{A}$, given by $(\mathbf{A}^T\mathbf{A})_{ij}$. Therefore, each element of $\mathbf{C}^{(1)}=\mathbf{A}^T\mathbf{A}$ represents an order-1 path between words $\mathbf{a}^i$ and $\mathbf{a}^j$ ($i,j=1..n$). We omit the normalization factors as it clear from equations (3.22) and (3.23) which normalization is to be used.

Documents d_1 and d_4 do not have an order-1 path but are linked together by both d_2 and d_3. Such a path with one intermediate vertex is called an order-2 path. The link between d_1 and d_4 can be represented as a combination of two order-1 paths from d_1 to d_2 and from d_2 to d_4 (similarity from d_1 to d_3 and from d_3 to d_4). The similarity value contributed via the document d_2 can be explicitly represented as $d_1 \rightarrow w_2 \rightarrow d_2 \rightarrow w_4 \rightarrow d_4$. The sub-sequence $w_2 \rightarrow d_2 \rightarrow w_4$ represents an order-1 path between words w_2 and w_4 which is the same as $C^{(1)}_{24}$. The contribution of d_2 in the similarity of $R^{(1)}_{14}$ via d_2 can thus be re-written as $A_{12}C^{(1)}_{24}A_{44}$. This is just the partial similarity measure since d_2 is not the only document that provides a link between d_1 and d_4. The similarity via d_3 (see FIGURE 3.3 (b)) is given by $A_{13}C^{(1)}_{34}A_{44}$. To find the overall similarity measure between documents d_1 and d_4, we need to add these partial similarity values given by $A_{12}C^{(1)}_{24}A_{44} + A_{13}C^{(1)}_{34}A_{44}$. Incidentally, this is the same value as given by the product of the matrices $(\mathbf{A}\mathbf{C}^{(1)}\mathbf{A}^T)_{14}$. Hence, the similarity matrix $\mathbf{R}^{(2)}$ at the second iteration corresponds to paths of order-2 between documents in the original matrix **A**.

Using the same analogy, words w_1 and w_4 do not have an order-1 path but are linked together by w_2 and w_4. Their similarity value is given by $w_1 \rightarrow d_1 \rightarrow w_2 \rightarrow d_2 \rightarrow w_4 + w_1 \rightarrow d_1 \rightarrow w_3 \rightarrow d_3 \rightarrow w_4$. As before, the subsequence $d_1 \rightarrow w_2 \rightarrow d_2$ is given by $R^{(1)}_{12}$ and $d_1 \rightarrow w_3 \rightarrow d_3$ is given by $R^{(1)}_{13}$. Therefore, the similarity between w_1 and w_4 is given by $A_{11}R^{(1)}_{12}A_{24}+A_{11}R^{(1)}_{13}A_{34}$ which corresponds to the element $(\mathbf{A}^T\mathbf{R}^{(1)}\mathbf{A})_{14}$. Thus, the matrix $\mathbf{C}^{(2)}=\mathbf{A}^T\mathbf{R}^{(1)}\mathbf{A}$ provides all order-2 paths between a given pair of words. Using the same criteria, it is easy to show that $\mathbf{R}^{(3)}$, $\mathbf{R}^{(4)}$,… and $\mathbf{C}^{(3)}$, $\mathbf{C}^{(4)}$,… provide paths of increasing order between pairs of documents and pair of words respectively. This is also true for document (or word) pairs that might be connected by a lower order path. For example documents d_1 and d_2 are connected by an order-1 path provided by w_2 but also an order-2 path provided by, say d_3 given by $d_1 \rightarrow w_3 \rightarrow d_3 \rightarrow w_4 \rightarrow d_2$. In general, it can be shown similarly that the matrices

(3.27) $$\mathbf{R}^{(t)} = \mathbf{A}\mathbf{C}^{(t-1)}\mathbf{A}^{\mathrm{T}}$$

and

(3.28) $$\mathbf{C}^{(t)} = \mathbf{A}^{\mathrm{T}}\mathbf{R}^{(t-1)}\mathbf{A}$$

represent all order-t path between documents and between words respectively. Note that without the normalization factor, we can represent $\mathbf{R}^{(t)}$ and $\mathbf{C}^{(t)}$ (since $\mathbf{R}^{(0)}$ and $\mathbf{C}^{(0)}$ are defined as $\mathbf{I}$) as

(3.29) $$\mathbf{R}^{(t)} = (\mathbf{A}\mathbf{A}^{\mathrm{T}})^{t}$$

(3.30) $$\mathbf{C}^{(t)} = (\mathbf{A}^{\mathrm{T}}\mathbf{A})^{t}$$

Re-introducing the normalization factor (as described previously) enables us to influence the contribution of different paths towards the similarity value based on the size of the document or number of occurrences of a given word. Using $\mathbf{A}^{\mathrm{R}}$ and $\mathbf{A}^{\mathrm{C}}$ to describe a row normalized and a column normalized matrix respectively, we could re-write equations (3.27) and (3.28) as

(3.31) $$\mathbf{R}^{(t)} = \mathbf{A}^{R}\mathbf{C}^{(t-1)}(\mathbf{A}^{R})^{\mathrm{T}}$$

(3.32) $$\mathbf{C}^{(t)} = (\mathbf{A}^{C})^{\mathrm{T}}\mathbf{R}^{(t-1)}\mathbf{A}^{C}$$

Notice that equations 3.25 and 3.26 are similar to equations (3.31) and (3.32) with the added constraint that the similarity value of a loop has unit weight.

We can now define the number of iterations to perform for χ-Sim. At each iteration t, one or more new links may be found between previously disjoint objects (documents or words) corresponding to paths with length of order-t; and existing similarity measures may be strengthened since higher-order links signifies more semantic relatedness. Iteration t thus amounts to count the number of walks of t steps between nodes.

It is worth noting here that iterating χ-Sim will indeed result in a fixed point for similarity matrices **R** and **C**, for non-neagtive values of A (i.e. $A_{ij} \geq 0$) since $\mathbf{R}^{(t+1)} \geq \mathbf{R}^{(t)}$ and $0 \leq R_{ij} \leq 1$ and $\mathbf{C}^{(t+1)} \geq \mathbf{C}^{(t)}$ and $0 \leq C_{ij} \leq 1$ (see Appendix III for proof). Iterating a large number of times would result in values of **R** and **C** to converge towards 1. It has been shown that "in the long run", the ending point of a random walk does not depend on its starting point (Seneta 2006) and hence it is possible to find a path (and hence similarity) between any pair of nodes in a connected graph (Zelikovitz and Hirsh 2001) by iterating a sufficiently large number of times. Moreover, redundant paths (see section 3.5) results in higher values of similarities between any given pair of words or documents.

In practice, however, co-occurrences beyond the 3rd and 4th order have little semantic relevance and hence are not interesting (Bisson and F. Hussain 2008; Lemaire and Denhière 2006). Therefore, the number of iterations t is usually limited to 4 or less.

3.5. Non-Redundant Walks

The elements of the matrix after the first iteration, $\mathbf{R}^{(1)}$ are the weighted (by the strengths of the corresponding links) number of one-step paths in the graph: the diagonal elements R_{rr} correspond to the paths from each document to itself, while the non-diagonal terms R_{rs} count the number of one-step paths between a document r and a neighbor s ($\forall r,s \in 1..m$), which is just the number of words they have in common. The matrix $\mathbf{R}^{(1)}$ after the first iteration is thus the adjacency matrix of the documents graph. Iteration t amounts thus to count the number of paths of t steps between nodes.

Calculating paths as shown above, however, reveals a critical shortcoming. When counting the number of paths of a given length, we consider two kinds of paths: paths that are compositions of lower length walks, for example $R^{(2)}_{12}$ also contains a walk of the form $d_1 \rightarrow w_2 \rightarrow d_1 \rightarrow w_2 \rightarrow d_2$. We refer to such a path as a *non-elementary* or *redundant path* since it contains a circuit from w_2 to itself. The second kind of paths are one in which no node is repeated (i.e. it contains no circuit) which we refer to as a *non-redundant* or *elementary path*. The former, however, do not add any new information between a given pair of documents or words being considered. Only paths that are non-redundant contribute with new information between the document or word pair being considered since they represent previously non-existent links in the bipartite graph. When calculating similarity values as discussed previously, the number of redundant paths grows exponentially because they represent paths that go back and forth between nodes that have been already visited. At the same time the contribution of the non-redundant paths is relatively smaller with increasing t. Thus, it is possible that the new information in terms of new document links is overshadowed by links of increasing (but redundant) length of previously connected nodes.

We explore here an alternative approach of the algorithm where we are interested in finding only the elementary paths in equation ((3.22) and (3.23). Finding all t^{th} order elementary paths in a graph is a well known NP-Hard problem. However, as described previously, we are only interested in finding paths of lower order (typically $t \leq 4$). We now proceed to illustrate a method of finding elementary paths of order 1, 2 and 3 in a given bipartite graph. We concentrate on the document similarity matrix $\mathbf{R}$, the arguments being transposable to word similarity matrix, $\mathbf{C}$. To make the presentation more intuitive we will adopt hereafter the paradigm of documents and words instead of rows and columns.

3.5.1. Preliminary Consideration

Consider the mixed products of normalized matrices

$$\mathrm{L}^{R}_{rs} \equiv (\mathbf{A}^{R}(\mathbf{A}^{C})^{T})_{rs} \tag{3.33}$$

$$\mathrm{L}^{C}_{ij} \equiv ((\mathbf{A}^{C})^{T}\mathbf{A}^{R})_{ij} \tag{3.34}$$

of dimensions $m \times m$ and $n \times n$ respectively.

The above products may be written explicitly, to understand the burden introduced by the normalizing factors

(3.35) $$\mathrm{L}^{R}_{rs} = (\mathbf{A}^{R}(\mathbf{A}^{C})^{T})_{rs} = \frac{1}{\mu_r}\sum_{c=1}^{n}\frac{A_{rc}A_{sc}}{\mu^{c}}$$

(3.36) $$\mathrm{L}^{C}_{ij} = ((\mathbf{A}^{C})^{T}\mathbf{A}^{R})_{ij} = \frac{1}{\mu^{i}}\sum_{r=1}^{m}\frac{A_{ri}A_{rj}}{\mu_r}$$

The sum in equations (3.35) and (3.36) are not simple scalar product of vectors. The sums depend on the dummy normalization factors μ^c and μ_r respectively, making the interpretation of these products not straightforward, even for the diagonal elements. For example, the number of paths from document *r* to document *s* through word *c*, $A_{rc}A_{sc}$, are weighted by a factor $1/\mu^c$ that is higher the less frequent the word *c* in the whole database and a factor μ_r that depends on the number and frequency of words in document d_r. Rare words are thus more relevant than frequent ones, since they enhance the weight of paths through them. Similarly, the number of paths between the words *i* and *j* are weighted by a factor $1=\mu_r$ that is higher the less frequent the documents containing these words and a factor μ^i that depends on the number of documents in which the word occurs and its frequency. The overall normalization in each case makes it *non-symmetric*.

Due to their structure the matrices in equations (3.35) and (3.36) may be written as follows (since both matrices present the same structure we drop the superscript): $L_{rs} = \kappa_r\sigma_{rs}$ and $(\mathbf{L})^{T}_{rs} \equiv L_{sr} = \kappa_s\sigma_{sr}$, with $\sigma_{rs} = \sigma_{sr}$. Then $L_{r\alpha}\,(\mathbf{L}^{T})_{\alpha s} = \kappa_r\kappa_s\sigma_{r\alpha}\sigma_{\alpha s}$ is symmetric. As a consequence, the direct product $\mathbf{L}\times\mathbf{L}^{T}$ and the matrix product $\mathbf{L}\mathbf{L}^{T}$ are symmetric. Notice also that, if **S** is a symmetric matrix, then $\mathbf{L}\mathbf{S}\mathbf{L}^{T}$ is also symmetric. Using the relationship to calculate **R** and **C** given by equations (3.22) and(3.23), we can expand their evolution as follows:

Iteration 1: Using the initialization with identity matrices for $\mathbf{R}^{(0)}$ and $\mathbf{C}^{(0)}$, we have $\mathbf{R}^{(1)}=\mathbf{A}^{R}(\mathbf{A}^{R})^{T}$ and $\mathbf{C}^{(1)}=(\mathbf{A}^{C})^{T}\mathbf{A}^{C}$, which are symmetric matrices. Since these are order-1 paths, they do not contain sub-paths that form a circuit and are hence contains only non-redundant paths.

Iterations 2: The second iteration gives $\mathbf{R}^{(2)} = \mathbf{A}^{R}(\mathbf{C}^{(1)})\mathbf{A}^{R}) = \mathbf{A}^{R}(\mathbf{A}^{C})^{T}\mathbf{A}^{C}(\mathbf{A}^{R})^{T} = \mathbf{L}^{R}(\mathbf{L}^{R})^{T}$ and $\mathbf{C}^{(2)} = (\mathbf{A}^{C})^{T}(\mathbf{R}^{(1)})\mathbf{A}^{C} = (\mathbf{A}^{C})^{T}\mathbf{A}^{R}(\mathbf{A}^{R})^{T}\mathbf{A}^{C} = (\mathbf{L}^{C})^{T}\mathbf{L}^{C}$, which is symmetric.

Iteration 3: The third iteration gives $\mathbf{R}^{(3)} = \mathbf{A}^{R}(\mathbf{A}^{C})^{T}\mathbf{A}^{R}(\mathbf{A}^{R})^{T}\mathbf{A}^{C}(\mathbf{A}^{R})^{T}$ which may be written as $\mathbf{R}^{(3)}=\mathbf{L}^{R}(\mathbf{A}^{R}(\mathbf{A}^{R})^{T})(\mathbf{L}^{R})^{T}$. Similarly, $\mathbf{C}^{(3)} = (\mathbf{A}^{C})^{T}\,\mathbf{A}^{R}(\mathbf{A}^{C})^{T}\mathbf{A}^{C}(\mathbf{A}^{R})^{T}\,\mathbf{A}^{C} = \mathbf{L}^{C}((\mathbf{A}^{C})^{T}\mathbf{A}^{C})(\mathbf{L}^{C})^{T}$.

We now proceed to determine the contribution of the non-redundant paths in equations (3.22) and (3.23) (more precisely a variant of the algorithm where the diagonal paths are not set to 1). We will be using the matrices $\mathbf{L}^{R}$ and $\mathbf{L}^{C}$ and one has to handle them carefully, since they are not symmetric.

3.5.2. Order 1 walks

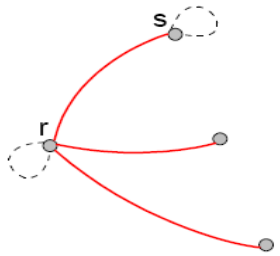


FIGURE 3.4 Elementary paths of order 1 between two different nodes (*r* and *s*) and between a node and itself (only represented with dashed lines for nodes *r* and *s*.) In red: the paths included in matrix $\mathbf{L}^{(1)}$.

FIGURE 3.4 above represent the elementary paths of order 1. As mentioned previously, $\mathbf{R}^{(1)}$ and $\mathbf{C}^{(1)}$ contains paths of order one only and hence do not contain any redundant paths.

3.5.3. Order 2 walks

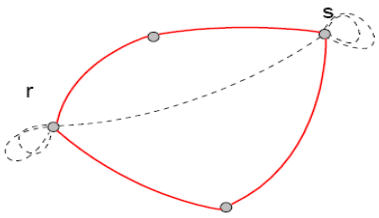


FIGURE 3.5 In red: elementary paths of order 2 between two different nodes (*r* and *s*). Any combination of 2-steps (dashed) paths is a redundant (not elementary) path. These are not counted in $\mathbf{L}^{(2)}$.

Initializing with the identity matrix, we get $\mathbf{R}^{(2)} = \mathbf{L}^{R}(\mathbf{L}^{R})^{T}$ and $\mathbf{C}^{(2)} = (\mathbf{L}^{C})^{T}\mathbf{L}^{C}$. In the following, we represent by a generic matrix **L** the adjacency matrix of the graph. We describe a way of calculating similarity based on non-redundant paths between documents, the approach being similar for word similarity. We drop the super-indices as it is clear which one of equation (3.33) or (3.34) is being referred to depending on whether we are calculating **R** or **C** respectively. Also we denote **L0** as a matrix that has the same out-of-diagonal elements as **L** but vanishing diagonal elements:

(3.37)
$$L0_{rs} = \begin{cases} 0 & for\ r = s \\ L_{rs} & otherwise \end{cases}$$

The elementary paths of order 2 are of the form $r \rightarrow \alpha \rightarrow s$ (where we use Greek letter for the dummy

variables in the sums) given by

$$L^{(2)}{}_{rs} = \sum_{a \neq r,s} L0_{ra} (\mathbf{L0})^T{}_{as} = \sum_{a} L0_{ra} (\mathbf{L0})^T{}_{as}$$

(3.38) $$L^{(2)}_{rs} = (\mathbf{L0}(\mathbf{L0})^{\mathrm{T}})_{rs}$$

Notice that, since the diagonal elements of matrix **L0** vanishes, the constraint $\alpha \neq r,s$ in the sum is automatically taken into account. FIGURE 3.5 represent some self-avoiding paths of order two between two different nodes. The diagonal elements $L^{(2)}{}_{rr}$ of $\mathbf{L}^{(2)}$ represent walks that goes from node r to a neighboring node and come back to r. We define $\mathbf{L0}^{(2)}$ that has the same out-of-diagonal elements as $\mathbf{L}^{(2)}$ but with vanishing diagonal elements.

3.5.4. Order 3 Walks

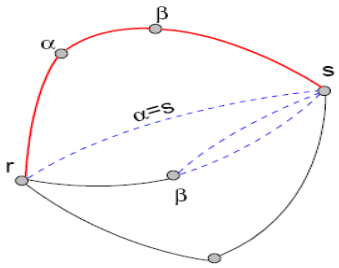


FIGURE 3.6 In red: elementary paths of order 3 between two different nodes (r and s). In blue (dashed lines): a redundant path.

Here, we are interesting in finding paths of the form $r \rightarrow \alpha \rightarrow \beta \rightarrow s$. The number of elementary paths of order 3 is given by $\mathbf{L}^{\mathrm{R}}(\mathbf{A}^{\mathrm{R}}(\mathbf{A}^{\mathrm{R}})^{\mathrm{T}})(\mathbf{L}^{\mathrm{R}})^{\mathrm{T}}$. If we substitute $\mathbf{A}^{\mathrm{R}}(\mathbf{A}^{\mathrm{R}})^{\mathrm{T}}$ as **B**, then we have

(3.39) $$L^{(3)}{}_{rs} = \sum_{\alpha \neq r,s;\beta \neq r,s;\alpha \neq \beta} L_{r\alpha} B_{\alpha\beta} \mathbf{L}^{\mathrm{T}}_{\beta s}$$

Using **L0** instead of **L** and **B0** instead of **B** eliminates the constraints $\alpha \neq r$ and $\alpha \neq \beta$, $r \neq \alpha$ and $\beta \neq s$ since it eliminates paths of the form $r \rightarrow \alpha \rightarrow \alpha \rightarrow s$, $r \rightarrow r \rightarrow r \rightarrow$s and $r \rightarrow s \rightarrow s \rightarrow s$. Hence $L^{(3)}{}_{rs}$ is given by

(3.40) $$L^{(3)}{}_{rs} = \sum_{\alpha \neq s;\beta \neq r} L0_{r\alpha} B0_{\alpha\beta} L0_{\beta s}$$

By relaxing the condition $\alpha \neq s$, we need to explicitly remove paths of the form $r \rightarrow s \rightarrow \beta \rightarrow s$. Therefore

$$L^{(3)}{}_{rs} = \sum_{\alpha,\beta \neq r} L0_{r\alpha} B0_{\alpha\beta} L0_{\beta s} - \sum_{\beta \neq r} L0_{rs} B0_{s\beta} L0_{\beta s}$$

Removing the constraint $\beta \neq s$ makes it necessary to remove paths of the form $r \rightarrow \alpha \rightarrow r \rightarrow s$

$$L^{(3)}{}_{rs} = \sum_{\alpha;\beta} L0_{r\alpha} B0_{\alpha\beta} L0_{\beta s} - \sum_{\beta \neq r} L0_{rs} B0_{s\beta} L0_{\beta s} - \sum_{\alpha} L0_{r\alpha} B0_{\alpha r} L0_{rs}$$

Since

$$\sum_{\beta \neq r} L0_{rs} B0_{s\beta} L0_{\beta s} = \sum_{\beta} L0_{rs} B0_{s\beta} L0_{\beta s} - L0_{rs} B0_{sr} L0_{rs}$$

We obtain, by collecting terms:

(3.41) $$L^{(3)}{}_{rs} = \left[\mathbf{L0B0(L0)}^{\mathrm{T}}\right]_{rs} - L0_{rs}\left[\mathbf{L0B0}\right]_{ss} - \left[\mathbf{L0B0}\right]_{rr}(\mathbf{L0})^{\mathrm{T}}_{rs} + \left[\mathbf{L0B0(L0)}^{\mathrm{T}}\right]_{rs}$$

This amounts to counting all the self-avoiding paths of two steps starting from r to any intermediate node α and then making one step to reach s. Since the matrices have vanishing diagonal terms, they guarantee that walks *like* $r \rightarrow s \rightarrow \beta \rightarrow s$ (see dotted paths in FIGURE 3.6) or $r \rightarrow \alpha \rightarrow r \rightarrow s$ are not counted.

We can enhance this, using the same method as presented above, to calculate paths of order-4 and above. As compared to equations 3.22 and 3.23 in section 3.3.6 which contains all paths (including redundant paths) between pairs of documents and pairs of words respectively, equations (3.38) and (3.41) contain only paths that corresponds to paths order-2 and order-3. Thus, to determine the similarity between documents (or words) at iteration t, one way could be to combine these individual similarity measures as follows (Chakraborti, Wiratunga, et al. 2007)

(3.42) $$\mathbf{R}^{(t)} = W_1 \cdot \mathbf{R}^{(1)} + W_2 \cdot (\mathbf{L}^{\mathrm{R}})^{(2)} + \ldots + W_t \cdot (\mathbf{L}^{\mathrm{R}})^{(t)}$$

(3.43) $$\mathbf{C}^{(t)} = W_1 \cdot \mathbf{C}^{(1)} + W_2 \cdot (\mathbf{L}^{\mathrm{C}})^{(2)} + \ldots + W_t \cdot (\mathbf{L}^{\mathrm{C}})^{(t)}$$

where W_1, W_2, etc are the weights associated with combining paths of order-1, order-2, etc.

3.5.5. Pruning Threshold

The method examined in the previous sub-section (3.5.4) enables us to calculate higher order co-occurrences between documents and words. A closer look at the method, however, highlights a couple of drawbacks.

- Firstly, calculating elements of $L^{(t)}{}_{rs}$ requires that we perform several matrix multiplication operations and store the intermediate result. This is necessary since we calculate all the existing paths between two objects and then successively remove those that are redundant. Moreover, to calculate the final similarity matrix $\mathbf{R}^{(t)}$ (or $\mathbf{C}^{(t)}$) we need to stock all lower-order similarity matrices $\mathbf{L}^{(i)}$ for $1 \leq i \leq t$. For large values of m and n, these matrices will have significant effect on the space complexity of the algorithm.
- Secondly, combining matrices as done in equations (3.42) and (3.43) requires finding weighting parameters W_1, W_2, etc which is not a trivial task for unsupervised learning (Chakraborti, Wiratunga, et al. 2007).

To understand the burden of higher order redundant paths in the original χ-Sim algorithm, we consider the example given in FIGURE 3.3 (an abstract of which is shown below for easy referencing) and examine paths of

order-2 between documents d_2 and d_4. The second order paths between d_2 and d_4 are listed below

(i) $d_2 \rightarrow w_2 \rightarrow d_2 \rightarrow w_4 \rightarrow d_4$

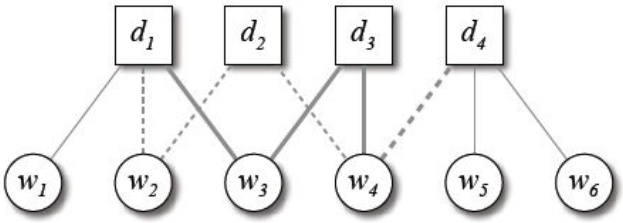
(ii) $d_2 \rightarrow w_4 \rightarrow d_2 \rightarrow w_4 \rightarrow d_4$ ♦

(iii) $d_2 \rightarrow w_4 \rightarrow d_3 \rightarrow w_4 \rightarrow d_4$ ♦

(iv) $d_2 \rightarrow w_4 \rightarrow d_4 \rightarrow w_4 \rightarrow d_4$ ♦

(v) $d_2 \rightarrow w_4 \rightarrow d_4 \rightarrow w_5 \rightarrow d_4$

(vi) $d_2 \rightarrow w_4 \rightarrow d_4 \rightarrow w_6 \rightarrow d_4$



Note that when using the word similarity matrix **C** as in equations 3.31 and 3.32, the paths indicated above with the symbol "♦" (paths (ii),(iii) and (iv)) shorten to $d_2 \rightarrow C_{44} \rightarrow d_4$. As described in section 3.3.6 where we explained the χ-Sim algorithm, elements along the diagonal of **R** and **C** are set to 1 at every iteration, hence $C_{44}=1$. The contribution towards the similarity between d_2 and d_4 resulting from these 3 paths will be $A_{24}A_{44}$ which is the same as the order-1 similarity between the documents as a result of sharing a common word. Paths (v) and (vi) contribute to the similarity value between d_2 and d_4 using word pairs that belong to different clusters (pairs (w_4,w_5) and (w_4,w_6)) while path (i) contribute to the similarity using word pairs that belong to the same cluster as shown in FIGURE 3.3.

It is interesting to observe that non-redundant paths of second order using equation (3.38) would have eliminated all the paths except (iii) since all other paths contain a circuit between d_2 or d_4. At the first iteration, the only similarity between d_2 and d_3 and between d_3 and d_4 comes via w_4. Hence at the second iteration when using non-redundant paths, the similarity between other words in documents d_2 and d_4 with their common word w_4 would have no influence on the similarity between d_2 and d_4. Using χ-Sim takes these similarities into account i.e. similarities between words w_2 of d_4 and words w_5 and w_6 of d_4 with w_4. Such similarities, however, would only make sense if they represent a strong (distributional) similarity relationship. Otherwise random co-occurrences that may not correspond to significant statistical relationships would overemphasize the similarity value since redundant paths could possibly bias the similarity between two objects as stated previously. Similarly, when comparing a document pair whose words belong to different clusters, such redundant paths could possibly over-emphasize their similarity value and distort the similarity rankings. As seen in our example of the χ-Sim algorithm in section 3.3.7, such similarity values tends to be smaller relative to word pairs that belong to the same cluster. Intuitively, words that belong to the same cluster have higher values since they occur in documents that are closely linked together than documents that appear in different clusters.

Based on this observation, we tentatively introduce in the χ-Sim algorithm a parameter, termed *pruning threshold* and denoted by ρ, that sets to zero the lowest ρ% of the similarity values in the matrices **R** and **C**. The idea behind this pruning threshold is as follows:

- To minimize the influence of random links between pairs of documents or words. Intuitively, document or word pairs that have very low similarity values are not semantically related and, therefore, any semantic links formed by such pairs may not carry the desired information.
- To minimize the effect of similarity propagation across document (and word) pairs that do not belong to the

same cluster as discussed previously.

The two formulae for calculating $\mathbf{R}^{(k)}$ and $\mathbf{C}^{(k)}$ at a given iteration k using the pruning parameter, ρ, is given by the following equations.

(3.44) $$R_{ij}^{(k)} = \left\{ \begin{array}{ll} 1 & if\ i = j \\ \left[\mathbf{A}^{\mathrm{R}} \mathbf{C}^{(i-1)} \left(\mathbf{A}^{\mathrm{R}} \right)^{\mathrm{T}} \right]_{>\rho} & \end{array} \right\}$$

(3.45) $$C_{ij}^{(k)} = \left\{ \begin{array}{ll} 1 & if\ i = j \\ \left[\left(\mathbf{A}^{\mathrm{C}} \right)^{\mathrm{T}} \mathbf{R}^{(i-1)} \mathbf{A}^{\mathrm{C}} \right]_{>\rho} & \end{array} \right\}$$

Where the subscript "$>\rho$" signifies that only elements higher than the lowest ρ % of the similarity values are considered. When using pruning, we will denote the algorithm as χ-Sim$_{\rho}$.

3.6. Relationship with Previous Work

Some of the algorithms introduced in chapter 2 (section 2.4) as alternate approaches to text clustering, use (either explicitly or implicitly) the underlying structure to estimate the underlying structure available within the dataset. In this section, we compare 3 other approaches which are closely related to our algorithm.

3.6.1. Co-clustering by Similarity Refinement

Jiang Zhang (J. Zhang 2007) proposed the 'co-clustering by similarity refinement' method which iteratively calculates two *refinement matrices* to generate similarity matrices. Using our notation where **R** and **C** are the two similarity matrices that contain similarity between objects (documents) and between features (words) respectively, we define two new matrices - $\mathbf{R}_{ref}$ and $\mathbf{C}_{ref}$ - their corresponding refinement matrices. The idea of using refinement matrices is to extend the classical object similarity measure (one which considers that features belong to an orthogonal space) to include the clustering structure of the feature space and vice versa. The concept is similar to using 'similar words' and 'similar documents' to contribute in the similarity measure as proposed in our algorithm. Ideally, the elements of the refinement matrices should correspond to 1 if the elements belong to the same cluster or 0 if they belong to different clusters. Since the true cluster structure is not known, the conditions on the refinement matrices are relaxed such that the values are either close to 1 or close to 0 depending on whether the elements are close together or not. In practice, the refinement matrices are estimated from the similarity matrices (as described below).

We define the similarity functions $sim^{\mathrm{R}}(\mathbf{a}_i, \mathbf{a}_j, \mathbf{C}_{ref})$ and $sim^{\mathrm{C}}(\mathbf{a}^i, \mathbf{a}^j, \mathbf{C}_{ref})$ that uses the refinement matrices to calculates the revised similarity values for **R** and **C** respectively. We will define these functions in a moment. Similarly, we define by $\xi(\mathbf{a}_i)$ the (normalized) coordinate vector of $\mathbf{a}_i$ in a k-dimensional space using spectral

embedding[14]. The algorithm is then given by:

1. Compute initial similarity matrices, **R** and **C** (using some similarity metric, e.g. Cosine measure)
2. Obtain $\forall \mathbf{a}_i \in \mathbf{A}$, $\xi(\mathbf{a}_i)$ *from* **R**

   and $\forall \mathbf{a}^i \in \mathbf{A}$, $\xi(\mathbf{a}^i)$ *from* **C**.
3. Construct refinement matrices:

   $(\mathbf{R}_{ref})_{ij} \leftarrow \xi(\mathbf{a}_i) \bullet \xi(\mathbf{a}_j)$

   $(\mathbf{C}_{ref})_{ij} \leftarrow \xi(\mathbf{a}^i) \bullet \xi(\mathbf{a}^j)$

   where "•" is the inner product.
4. Recompute similarity:

   $R_{ij} \leftarrow sim^{\mathrm{R}}(\mathbf{a}_i, \mathbf{a}_j, \mathbf{C}_{ref})$

   $C_{ij} \leftarrow sim^{\mathrm{C}}(\mathbf{a}^i, \mathbf{a}^j, \mathbf{R}_{ref})$
5. Cluster using **R** and **C**.

For two documents $\mathbf{a}_i$ and $\mathbf{a}_j$ with terms s and t that are closely related, we would like $A_{is} \cdot A_{jt}$ (and $A_{it} \cdot A_{js}$) contribute to the similarity between $\mathbf{a}_i$ and $\mathbf{a}_j$. Therefore, we use the refinement matrix to transform $\mathbf{a}_i$ and $\mathbf{a}_j$ such that each element that belongs to the same cluster is aligned after the transformation. For a refinement matrix $\mathbf{R}_{ref}$, $\hat{\mathbf{R}}_{ref}$ is obtained after normalizing each column vector of $\mathbf{R}_{ref}$ using the L-2 norm. Then for each document $\mathbf{a}_i$ and $\mathbf{a}_j$ the similarity is obtained by

$$Sim^{\mathrm{R}}(\mathbf{a}_i, \mathbf{a}_j, \mathbf{R}_{ref}) = \sum_{k=1..n} \sum_{l=1..n} ((\hat{\mathbf{r}}_{ref})^k \bullet (\hat{\mathbf{r}}_{ref})^l) \frac{A_{ik}}{\|\mathbf{a}_i\|_2} \frac{A_{jl}}{\|\mathbf{a}_j\|_2} \tag{3.46}$$

where $(\hat{\mathbf{r}}_{ref})^k, (\hat{\mathbf{r}}_{ref})^l$ represent the k^{th} and l^{th} column vectors of $\hat{\mathbf{R}}_{ref}$, and $(\hat{\mathbf{r}}_{ref})^k \bullet (\hat{\mathbf{r}}_{ref})^l$ will be close to 1 if terms i and j belongs to the same cluster and close to 0 otherwise. Steps 2 through 4 may be repeated several times to refine the values of the similarity matrices.

As compared to the χ-Sim algorithm, the similarity refinement algorithm differs in two ways. Firstly, instead of using the similarity values between rows (or columns) directly, an intermediate refinement value is used that is based on the similarity distribution (across the documents or terms) in a k-dimensional embedded space. Secondly (as discussed in section 3.3.5), the L-2 normalization used in their algorithm may not be well suited when calculating this kind of similarity values, since $\|\mathbf{a}_i\|_2 . \|\mathbf{a}_j\|_2 \leq (\sum(\mathbf{a}_i))\ (\sum(\mathbf{a}_i))$ and the numerator might increase faster then the denominator, thereby the refinement may overestimate the similarity value.

3.6.2. Similarity in Non-Orthogonal Space (SNOS)

The SNOS algorithm (N. Liu et al. 2004) is quite similar to our χ-Sim algorithm and was proposed to deal with

[14] Spectral embedding computes the k eigenvectors with the largest Eigen values of **R** (or **C**). Let e_1, e_2, …e_k be these eigenvectors, then the coordinates of $\mathbf{a}_i$ are the normalized version of the vector $\langle e_1(i), e_2(i), \ldots e_k(i) \rangle$

situations where the input space is non-orthogonal such as in text clustering. Using the notation introduced for the χ-Sim algorithm, we define the SNOS algorithm as follows

(3.47) $$\mathbf{R}^{(i)} = \lambda_1 \mathbf{A}\mathbf{C}^{(i-1)}\mathbf{A}^{\mathrm{T}} + (\mathbf{L}_1)^{(i-1)}$$

(3.48) $$\mathbf{C}^{(i)} = \lambda_2 \mathbf{A}^{\mathrm{T}}\mathbf{R}^{(i-1)}\mathbf{A} + (\mathbf{L}_2)^{(i-1)}$$

Where $\mathbf{L_1}^{(i)}=\mathbf{I}$ - $diag(\lambda_1\mathbf{A}\mathbf{R}^{(i)}\mathbf{A}^{\mathrm{T}})$ and $\mathbf{L_2}^{(i)}=\mathbf{I}$ - $diag(\lambda_2\mathbf{A}^{\mathrm{T}}\mathbf{C}^{(i)}\mathbf{A})$. λ_1 and λ_2 are normalizing factors satisfying the property $\lambda_1 < \|\mathbf{A}\|_1^2$ and $\lambda_2 < \|\mathbf{A}\|_\infty^2$ corresponding to the *1-norm* and *infinity norm* of the matrix **A**. The *1-norm* of a matrix is the maximum column sum of the matrix ($\|\mathbf{A}\|_1 = \max_{1\le j\le n}\left\{\sum_{i=1}^{m}|A_{ij}|\right\}$) and the *infinity norm* of a matrix is the maximum row sum of the matrix ($\|\mathbf{A}\|_1 = \max_{1\le i\le m}\left\{\sum_{j=1}^{n}|A_{ij}|\right\}$.

The two matrices $\mathbf{L}_1$ and $\mathbf{L}_2$ acts as reinitializing the diagonal elements to 1 at each iteration and SNOS differs primarily in the normalization step when compared to χ-Sim. Instead of normalizing by the length of the corresponding row/vectors, SNOS defines the normalization as $\lambda_1=\lambda_2=0.9/max\{\|\mathbf{A}\|_1, \|\mathbf{A}\|_\infty\}$. Although the normalization guarantees that the similarity values belong to the interval {0,1}, it is not suitable for comparison of vectors of different lengths (section 3.3.5).

3.6.3. The Simrank Algorithm

The Simrank algorithm was first proposed by (Jeh and Widom 2002) as a structural context based similarity measure that exploits the object-to-object relationships found in many application domains, such as web link where two pages are related if there are hyperlinks between them. The Simrank algorithm analyzes the graphs derived from such datasets and computes the similarity between objects based on the structural context in which they appear.

Recall from the definition of a graph in section 2.1.2 that a bi-partite graph G={X,Y,E} consists of 2 sets of vertices X and Y and a set of edges E between nodes of X and nodes of Y. For a node *v* in the graph, we will use the authors notation and denote I(*v*) as the set of *in-neighbors* and O(*v*) the set of *out-neighbors* where individual in-neighbors are denoted by $I_i(v)$ for $1 \le i \le I(v)$ and individual out-neighbors as $O_i(v)$ for $1 \le i \le O(v)$. If we consider documents as containing words in the document-by-term matrix, then each word in a document is its *out-neighbor* and each document which contains a word is its *in-neighbor*. Let us denote by $S(a,b)$ the similarity obtained using Simrank between the objects a and b ($S(a,b)\in[0,1]$). Similarly, let $S(c,d)$ denote the similarity between terms *c* and *d* ($S(c,d)\in[0,1]$), then

- for $a\neq b$, we define $S(a,b)$ as

(3.49) $$S(a,b) = \frac{C_1}{|\mathrm{O}(a)||\mathrm{O}(b)|}\sum_{i=1}^{|\mathrm{O}(a)|}\sum_{j=1}^{|\mathrm{O}(b)|} s(\mathrm{O}_i(a),\mathrm{O}_j(b))$$

- and for $c\neq d$, we define $S(c,d)$

(3.50) $$S(c,d) = \frac{C_2}{|\mathrm{I}(c)||\mathrm{I}(d)|} \sum_{i=1}^{|\mathrm{I}(c)|} \sum_{j=1}^{|\mathrm{I}(d)|} s(\mathrm{I}_i(c), \mathrm{I}_j(d))$$

For $a=b$ and $c=d$, we define $S(a,b)=1$ and $S(c,d)=1$ respectively. C_1 and C_2 are constants (whose values are less than 1) and are used so that all similarity values $S(a,b)$ with $a \neq b$ are less than 1. This was introduced as *a measure of uncertainty* when comparing documents (words) of different indices since we can not be sure they represent the same document (words). As in χ-Sim, we start by initializing **R** and **C** with the identity matrices **I** and iteratively re-compute **R** and **C** using equations (3.49) and (3.50) respectively.

The Simrank algorithm provides for an interesting comparison as it can be shown to be a special case of the χ-Sim algorithm. Equations (3.49) and (3.50) corresponds to a non-directed, non-weighted bipartite graph (weights on the edges are 1). This corresponds to a binary matrix **A**, where only the existence of non-existence of a term in a document is represented by 1 or 0. Without loss of meaning, we could re-write equations (3.49) and (3.50) for the case where $i \neq j$ and using our notation as

(3.51) $$S(\mathbf{a}_i, \mathbf{a}_j) = \frac{C_1}{\sum_{k=1..n} A_{ik} \sum_{k=1..n} A_{jk}} \sum_{k=1}^{n} \sum_{l=1}^{n} A_{ik} \cdot S(\mathbf{a}^k, \mathbf{a}^l) A_{jl}$$

(3.52) $$S(\mathbf{a}^k, \mathbf{a}^l) = \frac{C_2}{\sum_{i=1..m} A_{ik} \sum_{i=1..m} A_{il}} \sum_{i=1}^{m} \sum_{j=1}^{m} A_{ik} \cdot S(\mathbf{a}_i, \mathbf{a}_j) A_{jl}$$

since $A_{ik} \in [0,1]$ $\forall i,j$. Equations (3.51) and (3.52) form a generation of (3.49) and (3.50). As a result of the change in normalization, we can now have equations (3.51) and (3.52) take any values in **A**. Equations (3.51) and (3.52) are equivalent to equations (3.22) and (3.23) (except for the constants C_1 and C_2). In this sense, one could consider χ-Sim as a more generalized version of the Simrank algorithm that can deal with non-boolean values in **A**. (but still differ since χ-Sim deos not use the Coefficients C_1 and C_2) This is of significant importance since not only can we use weighted input matrices (such as *tf-idf*, section 2.1) for document (co-)clustering, but we can also use the χ-Sim algorithm to deal with other data such as those involving gene expressions (Chapter 5) which is not possible using the Simrank approach.

3.6.4. The Model of Blondel et al.

Several algorithms have been proposed for the case of matching similarity values between vertices of graphs. Kleinberg (Kleinberg 1999) proposed the Hypertext Induced Topic Selection (HITS) algorithm that exploits the hyperlink structure of web, considered as a directed graph, to rank query results. The premise of the HITS algorithm is that a web page serves two purposes: to provide information and to provide links relevant to a topic. The concept is based on the Hub and Authority structure. *Authorities* are web pages that can be considered relevant to a query, for instance, the home pages of Airbus, Boeing, etc are good authorities for a query "Airplanes", while web pages that point to these homepages are referred to as *hubs*. We can derive a simple recursive relation between *hubs* and *authorities* — a good *hub* is one that points to many *authorities* while good *authorities* are those that are pointed to

by many *hubs*.

Given a graph G=(V,E), let $\mathbf{h}^k$ and $\mathbf{a}^k$ be the vectors corresponding to the *hub* and *authority* scores associated with each vertex of the graph at iteration k, we denote $\mathbf{a}^k(p)$ and $\mathbf{h}^k(p)$ to identify the *authority* and *hub* scores of a vertex p. The HITS algorithm can be expressed in the following iterative manner

(3.53) $$\mathbf{a}^{k+1}(p) = \sum_{q \to p} \mathbf{h}^k(q)$$

(3.54) $$\mathbf{h}^{k+1}(p) = \sum_{p \to q} \mathbf{a}^k(q)$$

where $q \to p$ denotes that the webpage q contains a hyperlink towards page p. The initial values $\mathbf{a}^0(p)$ and $\mathbf{h}^0(p)$ are initialized to 1 and then iteratively updated using equations (3.53) and (3.54) respectively. The values of $\mathbf{a}^k$ and $\mathbf{h}^k$ are normalized at each iteration such that $\sum_i \mathbf{a}^k(i) = \sum_i h^k(i) = 1$.

The HITS algorithm described above allows for the comparison of each node of a graph of the following structure,

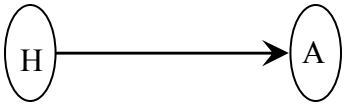


Figure 3.7 A Hub→Authority graph

Blondel et al. (Blondel et al. 2004) proposed a generalization of the HITS algorithm for all graphs. Blondel et al. considered similarity measures of a directed graph i.e. based on asymmetric adjacency matrices, which allows us to compare the similarity between nodes of one graph, say G_1, and another graph, say G_2. The concept is similar to other methods we have studied (SimRank, χ-Sim, etc) but extended to two graphs and applied on directed graphs — a node i in G_1 is similar to a node j in G_2 if the neighborhood of i in G_1 is also similar to the neighborhood of j in G_2. Let n_1 and n_2 be the number of vertices in G_1 and G_2 respectively, we define a similarity matrix $\mathbf{S}$ (of dimensions n_1 by n_2) given by the following recursive equation,

(3.55) $$\mathbf{S}^{k+1} = \mathbf{L}_2 \mathbf{S}^k \mathbf{L}_1^{\mathrm{T}} + \mathbf{L}_2^{\mathrm{T}} \mathbf{S}^k \mathbf{L}_1$$

where $\mathbf{L}_1$ and $\mathbf{L}_2$ are adjacency matrices of G_1 and G_2 respectively and $\mathbf{S}_0$ is initialized by setting all elements to 1. One may introduce a normalization factor to ensure convergence, such as dividing $\mathbf{S}_{k+1}$ by its Frobenius Norm (see section 2.5.1) which has been shown to converge for even and odd iterations (Blondel et al. 2004).

A special case for the similarity matrix in equation (3.55) for computing similarity between vertices of the same graph, where $G_1=G_2=G$ and $\mathbf{L}_1=\mathbf{L}_2=\mathbf{L}$ is given by,

(3.56) $$\mathbf{S}^k = \frac{\mathbf{L}\mathbf{S}^{k-1}\mathbf{L}^{\mathrm{T}} + \mathbf{L}^{\mathrm{T}}\mathbf{S}^{k-1}\mathbf{L}}{\left\| \mathbf{L}\mathbf{S}^{k-1}\mathbf{L}^{\mathrm{T}} + \mathbf{L}^{\mathrm{T}}\mathbf{S}^{k-1}\mathbf{L} \right\|_F}$$

The method of Blondel et al. has been used for the extraction of synonyms with considerable success (Blondel et al. 2004). The data consists of a matrix, which corresponds to a directed graph, constructed from a dictionary where each node of the graph is a word and there exists a link (edge) between vertex (representing a word) *i* and vertex *j* if vertex *j* appears in the definition of *i*. A word, *w*, corresponding to a query, is chosen for which its neighborhood graph G_w is constructed. Note that G_w is a sub-graph of G representing only vertices in which *w* appears. A sample sub graph for the word "likely" is given in Figure 3.8. They then compare the similarity score of the vertices of the graph G_w with the central vertex of the structure graph

$$1 \rightarrow 2 \rightarrow 3$$

and the resulting words are ranked by decreasing score. In other words, we rank each word w' based on the similarity score between words that has w' in their definition with those words that have *w* in their definition, and the similarity score between words that appear in the definition of w' with those that appear in the definition of *w*.

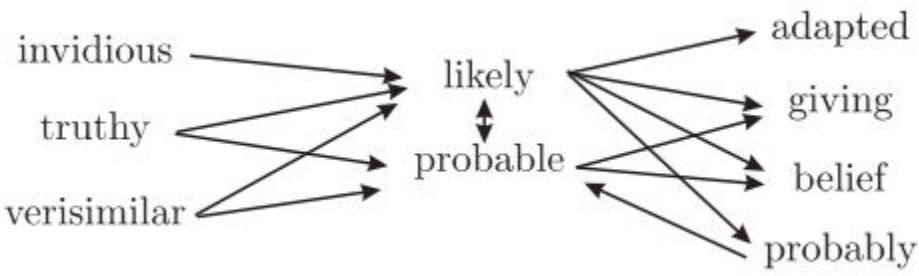


Figure 3.8 Part of the neighborhood graph associated with the word "likely". The graph contains all words used in the definition of likely and all words using likely in their definition **(Blondel et al. 2004)**.

Comparison with the χ-Sim Measure

The similarity measure given in equation (3.56) corresponds to the similarity score between vertices in a directed graph. We can consider a special case where the adjacency matrices are symmetric i.e. undirected graph, and $G_1=G_2=G$. The resulting similarity matrix can be expressed using the matrices as

$$\mathbf{S}^k = \frac{1}{\left\|\mathbf{M}\mathbf{S}^{k-1}\mathbf{M}\right\|_F}\mathbf{M}\mathbf{S}^{k-1}\mathbf{M} \tag{3.57}$$

since $\mathbf{L}=\mathbf{L}^T$. Note that if we drop the normalization factors and the fact that we define $R^k_{ii}=1$ (i.e. set the diagonal to 1 at each iteration), we can observe a similarity between our χ-Sim similarity measure and the score obtained using Blondel et al. By replacing the values of **C** in equation (3.22) from equation(3.23), we can obtain the following recursive equations for χ-Sim (note that in our case $\mathbf{A}\mathbf{A}^T=\mathbf{L}$),

$\mathbf{R}^{(0)}=\mathbf{I}$ $\mathbf{C}^{(0)}=\mathbf{I}$

$\mathbf{R}^{(1)}=\mathbf{A}\mathbf{C}^{(0)}\mathbf{A}^T=\mathbf{A}\mathbf{A}^T=\mathbf{L}$ $\mathbf{C}^{(1)}=\mathbf{A}^T\mathbf{R}^{(0)}\mathbf{A}=\mathbf{A}^T\mathbf{A}$

$\mathbf{R}^{(2)}=\mathbf{A}\mathbf{C}^{(1)}\mathbf{A}^T=\mathbf{A}\mathbf{A}^T\mathbf{A}\mathbf{A}^T=\mathbf{L}\mathbf{R}^{(0)}\mathbf{L}$ $\mathbf{C}^{(2)}=\mathbf{A}^T\mathbf{R}^{(1)}\mathbf{A}=\mathbf{A}^T\mathbf{A}\mathbf{A}^T\mathbf{A}$

$\mathbf{R}^{(3)}=\mathbf{A}\mathbf{C}^{(0)}\mathbf{A}^T=\mathbf{A}\,\mathbf{A}^T\,\mathbf{A}\mathbf{A}^T\,\mathbf{A}\,\mathbf{A}^T=\mathbf{L}\mathbf{R}^{(1)}\mathbf{L}$ $\mathbf{C}^{(1)}=\mathbf{A}^T\mathbf{R}^{(0)}\mathbf{A}=\mathbf{A}^T\mathbf{A}\mathbf{A}^T\mathbf{A}\mathbf{A}^T\mathbf{A}$

…

and in general $\mathbf{R}^{(k)}=\mathbf{L}\mathbf{R}^{(k-2)}\mathbf{L}$. We can now re-introduce the normalization factor as previously (section 3.5.1). Let $\mathbf{L}^R$ and $\mathbf{L}^C$ be the two row and column normalized adjacency matrices given by equations (3.35) and (3.36) respectively, then

$$\mathbf{R}^{(k)} = \mathbf{L}^R\mathbf{R}^{(k-2)}(\mathbf{L}^R)^T \tag{3.58}$$

and

$$\mathbf{C}^{(k)} = \mathbf{L}^C\mathbf{C}^{(k-2)}(\mathbf{L}^C)^T \tag{3.59}$$

for $k\geq 3$.

These recursions show that the calculation of matrices **R** and **C** can be disentangled. Comparing equations (3.58) to (3.57), we observe that (apart from the difference in normalization), the similarity values at iteration k depends on those of k-2 rather than k-1. One can interpret this graphically as follows: multiplying the adjacency matrix $\mathbf{A}\mathbf{A}^T$ compares vertices i and j by their common neighbors. Therefore, one needs two iterations to compare the similarity between the neighbors of i with the neighbors of j as expressed in terms of the matrix **L** in equation(3.58).

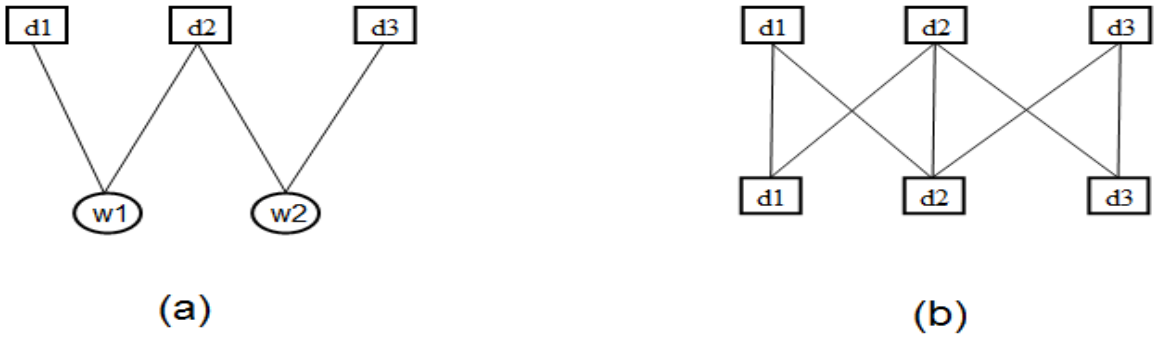


Figure 3.9 A sample graph corresponding to (a) the adjacency matrix **A**, and (b) the adjacency matrix **L**

There are certain differences, however, between the model of Blondel et al. and our method. Firstly, our normalization is dependent on the document and word vector sizes. This, we believe, is particularly important in the case of document clustering as will be seen in the experimental part (section 4.5.4) where we compare our method to SNOS method (section 3.6.2) that primarily differ in the normalization. Secondly, the initialization used by Blondel et al. corresponds to the matrix **1** i.e. setting all elements of $\mathbf{R}^{(0)}$ to 1 while our initialization uses the identity matrix.

Thirdly, as mentioned previously (section 3.3.6), we set the diagonal values of $\mathbf{R}^{(k)}$ and $\mathbf{C}^{(k)}$ to 1 at each iteration which is not the case in the method of Blondel et al.

One may also compare χ-Sim with the method of Blondel et al. by using the adjacency matrix **L** instead of **A** as the input matrix. The difference between the two is shown in Figure 3.9. Using χ-Sim to compute the document similarity matrix **R** as given in equation(3.22), we compute the similarity between vertices (documents) i and j (R_{ij}) by taking the element A_{ik} that corresponds to the edge between document i and word k (Figure 3.9a) and multiplying it by the element $(A_{jl})^T$ which corresponds to the edge from word l to document j ($\forall i,j \in 1..m$, $\forall k,l \in 1..n$). When comparing word k and l, we multiply by their similarity C_{kl}. Now if we use the adjacency matrix **L**, using the same analogy, comparing documents i and j now corresponds to multiplying L_{ik} with L_{jl} ($\forall i,j,k,l \in 1..m$) and comparing document k with document l (Figure 3.9b) requires the similarity R_{kl}. The corresponding equation is given by,

$$\mathbf{R}^{(k)} = (\mathbf{L} \otimes \mathbf{NR})\mathbf{R}^{(k-1)}(\mathbf{L} \otimes \mathbf{NR}) \tag{3.60}$$

since $\mathbf{L}=\mathbf{L}^T$ and **NR** is a normalization matrix as defined in section 3.3.6. We see that equation (3.60) is very similar to equation (3.57) of Blondel et al. The difference, of course, is in the normalization.

Notice that the similarity value given by equation (3.60) is no longer a co-similarity measure since the similarity between documents i and j is given by their shared documents irrespective of the individual words that contribute to this similarity i.e. documents are similar if they have *similar* (document) neighbors but not necessarily if they share *similar* words.

3.6.5. Other Approaches

The proposed measure could also be compared to several other approaches for instance the Latent Semantic Analysis (LSA) technique described in section 2.4.1. LSA also considers the words when analyzing documents since using SVD is reduces the dimensionality of both rows and columns. Moreover, as was discussed previously in the same section, a low rank approximation to the original matrix A also takes (implicitly) certain higher-order co-occurrences into account. However, unlike our approach LSA is not iterative in nature and doesn't explicitly compute similarities between rows or between columns while generating the low rank matrix.

Here we compare another we would like to mention another such technique called Correspondance Analysis (CA). Correspondence analysis is an exploratory technique designed to analyze simple two-way and multi-way tables containing some measure of correspondence between the rows and columns. These methods were originally developed primarily in France by Jean-Paul Benzérci in the early 1960's and 1970's (e.g., see for instance (Benzécri 1969) but have only more recently gained increasing popularity in English-speaking countries (Carroll, Green, and Schaffer 1986). [15]

As opposed to traditional hypothesis testing designed to verify *a priori* hypotheses about relations between variables, exploratory data analysis is used to identify systematic relations between variables when there are no (or

[15] Note that similar techniques were developed independently by different authors, where they were known as optimal scaling, reciprocal averaging, optimal scoring, quantification method, or homogeneity analysis

incomplete) *a priori* expectations as to the nature of those relations. Its primary goal is to transform a table of numerical information into a graphical display, in which each row and each column is depicted as a point. Let **x** denote the vector corresponding to the mean score of the rows, called "row scores" and **y** correspond to a vector of "column scores". Let $Sum(\mathbf{a}_i)$ denote the sum of the elements of the row vector $\mathbf{a}_i$ and $Sum(\mathbf{a}^j)$ denote the sum of the elements of the column vector $\mathbf{a}^j$. Also let $\mathbf{D}_r$ denote the diagonal matrix of row sums and $\mathbf{D}_c$ to denote the diagonal matrix of column sums. We can calculate the calibrated row scores based on the column scores as $x_i^{(t+1)} = \sum_j A_{ij} y_j^{(t)} / sum(\mathbf{a}_i)$ and the column scores, now based on the calibrated row scores as $y_i^{(t+1)} = \sum_j A_{ij} x_j^{(t)} / sum(\mathbf{a}^i)$. The corresponding equations for calculating **x** and **y** are

(3.61) $$\mathbf{x}^{(t+1)} = \mathbf{D}_r^{-1}\mathbf{A}\mathbf{y}^{(t)}$$

(3.62) $$\mathbf{y}^{(t+1)} = \mathbf{D}_c^{-1}\mathbf{A}^{\mathrm{T}}\mathbf{x}^{t}$$

The process is known as *two way averaging algorithm* and its eigenvectors are the solution to the correspondence analysis problem of the matrix A (Hill 1974). The row analysis of a matrix consists in situating the row profiles in a multidimensional space and finding the low- dimensional subspace, which comes closest to the profile points. The row profiles are projected onto such a subspace for interpretation of the inter-profile positions. Similarly, the analysis of column profiles involves situating the column profiles in a multidimensional space and finding the low-dimensional subspace, which comes closest to the profile points. In a low *k*-dimensional subspace, where *k* is less than *m* or *n*, these two *k*-dimensional subspaces (one for the row profiles and one for the column profiles) have a geometric correspondence that enables us to represent both the rows and columns in the same display. The row and column analyses are intimately connected. If a row analysis is performed, the column analysis is also *ipso facto* performed, and vice versa. The two analyses are equivalent in the sense that each has the same total inertia, the same dimensionality and the same decomposition of inertia into principal inertias along principal axes.

Correspondance analysis uses a similar concept to ours that updates values of **x** based on values of **y** and vice versa.However, in this case we are not finding similarity values between row objects or column objects as given by our matrices **R** and **C**.

3.7. Extension to the Supervised Case

3.7.1. Introduction

Automated text categorization has received a lot of attention over the last decade mostly due to the increased availability of text documents in digital form. Automated text categorization is defined as the task of assigning predefined categories to new unlabeled text documents. There are two potential ways to incorporate background knowledge in text mining – by using external knowledge such as external thesauri or repositories (Zelikovitz and Hirsh 2001; Gabrilovich and Markovitch 2005), or by incorporating background knowledge about category

information of documents from a training dataset whose category labels are known. Since the χ-Sim algorithm uses both the matrix regarding similarity about documents **R** and the similarity about terms **C**, it is possible to incorporate such additional knowledge coming from both ways within the framework of the algorithm. For example, if we had before hand knowledge about the semantic relationship between words in a data corpus – such as those coming from thesaurus or Wordnet®, we could potentially incorporate this information in the word similarity matrix **C**, by modifying the initialization step and using such similarity values instead of an identity matrix.

In this section, we are concerned with the case of supervised text categorization, with a training dataset whose category labels are known beforehand. Several works in the LSI framework have shown that incorporating class knowledge can increase the result of the classification task (Gee 2003; Chakraborti, Mukras, et al. 2007; Chakraborti et al. 2006). Similarly, using word clustering has also been shown to improve the result of the text categorization (Takamura and Matsumoto 2002; Bekkerman et al. 2003). Motivated by their results, we seek to expand the χ-Sim clustering algorithm to the supervised classification task by exploiting information about class knowledge from the training dataset. This explicit class knowledge, when taken into account, can lead to improved similarity estimation between elements of the same class while reducing similarity between elements of different classes.

Given a training dataset $\mathbf{a}_1, \ldots, \mathbf{a}_{m1}$ with $m1$ labeled examples defined over n words with discrete (but not necessarily binary) category labels, we denote a document i from this dataset as $\mathbf{a}_i^{\text{train}}$ to denote the fact that its category label is known. The *r1* examples in the training set form the document-term matrix $\mathbf{A}_{\text{train}}$. Let $\hat{\mathrm{X}}$ be a set of k possible document categories ($\hat{\mathrm{X}}=\{\hat{x}_1,\hat{x}_2,\ldots,\hat{x}_k\}$ and $|\hat{\mathrm{X}}|=k$), we denote by $\mathrm{C}_{\mathrm{X}}(\mathbf{a}_i^{train}) \rightarrow \hat{x}_j$, $1 \le i \le m1$, $\hat{x}_j \in \hat{\mathrm{X}}$ a document i in the training set whose class label is characterized by $\hat{x}_j$. Similarly, let $\mathbf{a}_1^{\text{test}}, \ldots, \mathbf{a}_{m2}^{test}$ be a test dataset of size *m2* (also with discrete but not necessarily binary category labels) but whose labels are not known to the algorithm beforehand. The *m2* examples in the test dataset form the document-term matrix $\mathbf{A}_{test}$ also defined over n words. Note that n is the size of the dictionary of the corpus. We wish to learn the matrix **C** that reflects category labels of the documents in $\mathbf{A}_{\text{train}}$ as given by equation 3.13, and use the learnt **C** matrix to categorize each document in $\mathbf{A}_{test}$ to one of $\hat{\mathrm{X}}$ categories.

The similarity measure described in the previous section is purely unsupervised. The **R** and **C** matrices are initialized with an identity matrix since in an unsupervised environment we have no prior knowledge about the similarity between two documents or two words. As such, only the similarity between a document and itself or a word and itself is initialized with 1 and all other values are set to 0. However, with the availability of category labels from $\mathbf{A}_{train}$, it is possible to influence the similarity values such that documents belonging to the same category are brought closer together and farther away from documents belonging to a difference category. In order to influence the similarity between two documents $\mathbf{a}_i$ and $\mathbf{a}_j$ is defined by equation (3.13), we would like to influence the word similarity matrix **C** such that words representative of a document category (a word representative of a document category is one that occur mostly in documents of one category) are brought closer to each other and word representative of different document categories drawn farther apart.

But word similarities, in turn, are given by equation 3.14 so we would also like to incorporate the category information in the matrix **R** such that documents belonging to the same category are brought closer to each other and

farther away from documents belonging to difference categories. The overall objective is that when using a classification algorithm such as the *k* Nearest Neighbor (*k*-NN) approach, a document vector $\mathbf{a}_i$ will find its *k* nearest neighbors within the same document categories as itself since documents belonging to this category have higher similarity values.

We present below a two-pronged strategy of incorporating category information as

1) increasing within category similarities, and
2) decreasing out-of-category similarities.

In the following, we explore two methodologies to incorporate class knowledge into the learned similarity matrices. As a first case, we try the 'sprinkling' mechanism proposed in (Chakraborti et al. 2006) for the supervised LSI. The idea behind this is that since the χ-Sim algorithm explicitly captures higher-order similarities, adding class discriminating columns to the original document-term matrix will force the higher-order similarities between documents belonging to the same class. The second approach (S. F Hussain and Bisson 2010) is a more intuitive and well adapted to this algorithm in that it embeds class knowledge into the document-document similarity matrix utilized by the algorithm. The basic concept is that a word pair will be assigned a higher similarity value if they occur (either directly or through higher order occurrences) in the same document category.

3.7.2. Increasing within Category Similarity Values.

As stated above, our intention is to bring words representative of a document category closer together. One way of doing this, since we know the category labels of the documents, is by padding each document in the training set by an additional dummy word that incorporates the category information for that class. The original and revised document-term matrix $\mathbf{A}_{train}$ is shown in FIGURE 3.10.

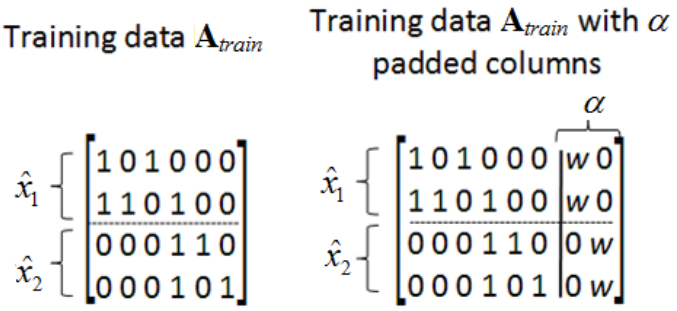


FIGURE 3.10 Incorporating category information by padding α columns.

The entries for each of these padded columns are weighted by *w*. When we apply equation 3.14 to calculate word similarities in the revised matrix, the padded columns force co-occurrences between words in the same class. As a result, the word representatives of a class are brought closer together. The value of *w* is used to emphasize the

category knowledge. Using a very small value for *w* would result in little influence on the similarity values. On the other hand, using a large value for w will distort the finer relationships in the original data matrix. The value of *w* can be determined empirically from the training dataset.

3.7.3. Decreasing Out of Category Similarity Values

For the document similarity matrix, we proceed with the same objective in mind as previously. Augmenting class information as described above not only brings words closer together but also enhances the similarity between documents of the same class since each of the padded dummy word is present in all documents of the same document category

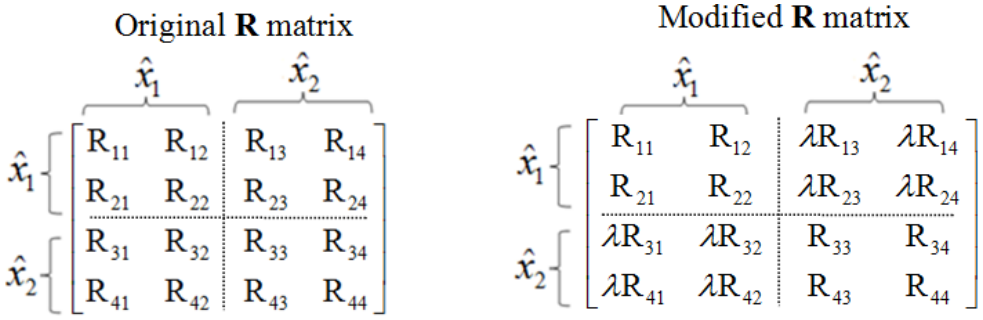


FIGURE 3.11 Reducing similarity between documents from different categories.

However, many words transcend document categorical boundaries and, as such, generate similarities between documents belonging to difference categories. This phenomenon is cascaded when **R** and **C** are iterated. The drawback of this is that higher order word similarities are mined based on co-occurrences resulting from documents from different categories. We wish to reduce the influence of such higher order co-occurrences when mining for word similarities. We propose a weighting mechanism that reduces the similarity value between documents that do not belong to the same category. The mechanism is shown in FIGURE 1.1. At each iteration, the values R_{ij} ($C_X(\mathbf{a}_i) \neq C_X(\mathbf{a}_j)$) are multiplied by a weighting factor λ ($0 \leq \lambda \leq 1$). The parameter λ determines the influence of category information on the higher order word co-occurrences. If λ=0, we force word co-occurrences to be generated from documents of the same category only while λ=1 corresponds to the previous unsupervised framework which relaxes this constraint and all higher order occurrences contribute equally to the similarity measure. The value of the parameter λ can be thought of as task dependent.

Intuitively, for the task of document categorization, highly discriminative words should have relatively higher similarity values than words that occur frequently in more than one document category, hence a small value of λ is desirable. Intuitively, for the task of document categorization, highly discriminative words should have higher similarity and hence a small value of λ is desirable. We will see the effect of λ on the empirical results in section 4.6 when we perform document categorization task on text dataset.

3.7.4. Illustrated Example

We describe here a small example to analyze the effect of incorporating category knowledge on the evolution of document and word similarity matrices using a toy example given in FIGURE 3.12. Given a training data document-by-term matrix, $\mathbf{A}_{train}$ with 4 documents belong to two categories ($\hat{x}_1$ and $\hat{x}_2$) and defined over 6 words. There is only one word that provides a link between the documents in the two categories (column 4). FIGURE 3.12 shows the input document-term matrix $\mathbf{A}_{train}$, the document similarity matrix **R** and the word similarity matrix **C** at iteration 1 and 2 respectively using various methods of incorporating category information. FIGURE 3.12 (a) shows the evolution of the similarity matrices without incorporating any class knowledge.

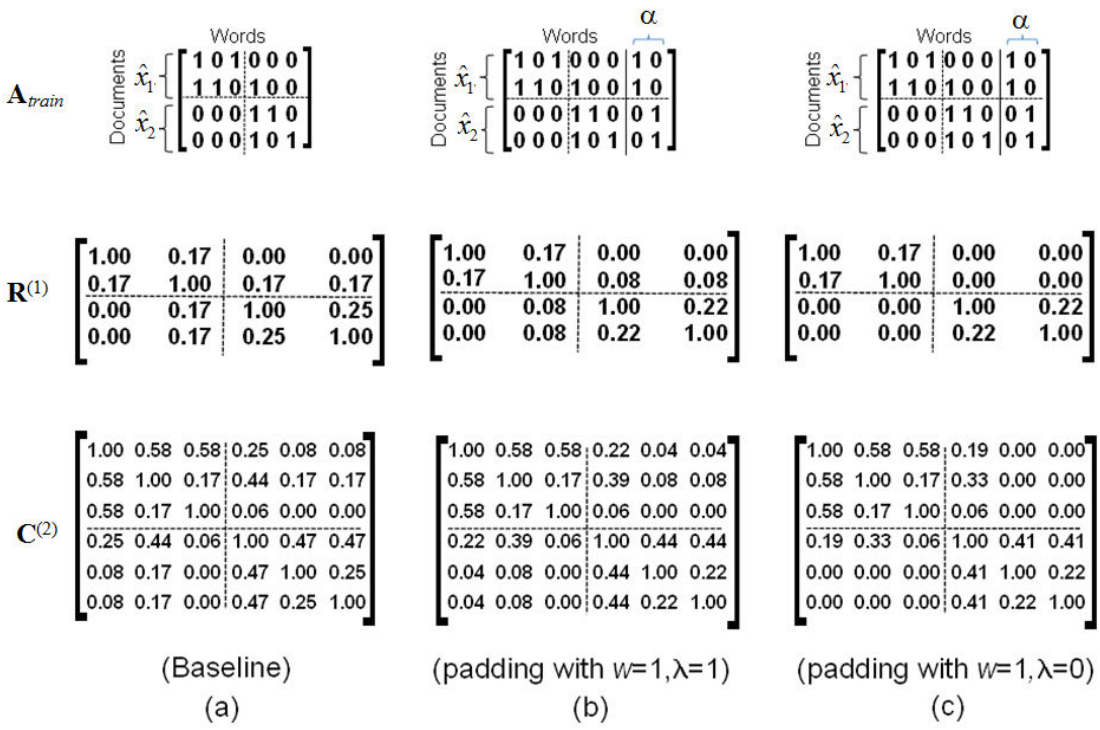


FIGURE 3.12 Sample **R** and **C** matrices on (a) the training dataset $\mathbf{A}_{train}$ with no categorical information, (b) padding $\mathbf{A}_{train}$ with α dummy words incorporating class knowledge, and (c) padding the $\mathbf{A}_{train}$ matrix and setting similarity values between out-of-class documents to zero.

Documents d_1 and d_2 belong to category 1 while documents d_3 and d_4 belong to category 2. First we consider the document similarity between documents of the same category, for example d_1 and d_2, and documents corresponding to different categories, for example d_2 and d_3. Without augmenting $\mathbf{A}_{train}$ with category discriminating columns, the similarity between d_1 and d_2 and between d_2 and d_3 is the same i.e. 0.17. Next we add a category discriminating column for each class as described in the previous subsection (see FIGURE 3.11). For the sake of simplicity, we set the value of w=1 as shown in FIGURE 3.12 (b). One can immediately see the effect of this on the

document similarity matrix $\mathbf{R}^{(1)}$. Notice that the similarity between d_1 and d_2 remains 0.17 instead of increasing. This is because of the normalization factor since $\mathcal{N}(\mathbf{a}_1,\mathbf{a}_2)$ in our case is a product of the L_1 norms of d_1 and d_2. The normalization factor typically increases more than the numerator in equations 3.13 and 3.14. In comparison to the similarity value between d_2 and d_3, however, the similarity between d_1 and d_2 is now relatively stronger as a result of adding the category information.

Now consider the similarities between w_1 and other words in FIGURE 3.12 (a). Words w_1 and w_2 for example, both of which only appear in documents of the first category, have a similarity value of 0.58 while words w_1 and w_5 where w_5 only occurs in documents of category 2 have a similarity of 0.08. The link between w_1 and w_5 is provided by w_4 which is shared by documents d_2 and d_3. Adding category information slightly decreases the similarity value between words 1 and 5 to 0.04 as can be seen in FIGURE 3.12 (b). This reduction in similarity of objects belonging to different classes is a result of the cascading effect of inter-twining **R** and **C** matrices since the documents 2 and 3 now have a lower similarity value as seen above. When embedding the category knowledge into the **R** matrix, the effect is much stronger. This effect is even greater in FIGURE 3.12 (c) where the similarity value between w_1 and w_5 vanishes since the value R_{23} was zeroed by the weighting factor, λ. By varying the value of λ, we can explicitly control the contribution of such links to the similarity measure.

3.7.5. Labeling the Test Dataset

This section describes two approaches to label test data. Firstly, we use the popular *k*-Nearest Neighbors algorithm to associate a test dataset to exactly one of the $\hat{\mathrm{X}}$ document categories. To compare a document vector in the test matrix, say $\mathbf{a}_j^{test}$, with a document vector from the training matrix, say $\mathbf{a}_i^{train}$, we compute their similarity value $Sim(\mathbf{a}_i^{train},\mathbf{a}_j^{test})$ as

$$Sim(\mathbf{a}_i^{\text{train}},\mathbf{a}_j^{\text{test}}) = \frac{\mathbf{a}_i^{\text{train}}(\mathbf{C})\mathbf{a}_j^{\text{test}^{\mathrm{T}}}}{\mu_i^{\text{train}}\mu_j^{\text{test}}} \tag{3.63}$$

where $(\mathbf{a}_i^{train})^{\mathrm{T}}$ is the transpose of the vector $(\mathbf{a}_i^{train})$. The category of each test document is then decided by a majority weight of its *k*-Nearest Neighbors with highest similarity values.

The above approach necessitates the comparison of each test document with all training documents which could be an expensive operation when the number of documents in the training set is large. Therefore, we also explore a second approach which is significantly faster. Instead of comparing each test document will all training documents from every category, we form a category vector $\mathbf{v}_i^{cat}, j\in[1..\hat{\mathrm{X}}]$. The category vector $\mathbf{v}_i^{cat}$ is defined as a vector containing the component wise sum of all the document vectors belonging to the category $\hat{x}_i$ i.e. the j^{th} element of v_i^{cat} is the number of times words *j* can occurs in documents whose category label is $\hat{x}_i$. The test documents is then assigned to the category having the highest similarity value with the $\mathbf{a}_j^{test}$, given by

(3.64) $$Sim(\mathbf{v}_i^{cat}, \mathbf{a}_j^{test}) = \frac{\mathbf{v}_i^{cat}(\mathbf{C})\mathbf{a}_j^{test\,T}}{\mu_i^{cat}, \mu_j^{test}}$$

We call this approach the nearest category instead of the nearest neighbor and denote it as NC.

Notice that in this case, we only need to do $|\hat{X}|$ comparisons (where $|\hat{X}|$ is the number of document categories). This approach is similar to the Rocchio classification algorithm (Joachims 1997). However, unlike Rocchio, our category prototype is defined solely by summing the category documents and does not take into account the out of category documents.

3.8. Conclusion of the Chapter

In this chapter, we presented a new co-similarity based technique to compute the similarity measure between two objects. The proposed technique takes into account the structure of the graph resulting from a bipartite representation of documents and words in a data corpus. We started by the basic concept of comparing similarity between documents based on the shared words and introduce the concept of induced similarity. We proposed an iterative method to compute weighted similarity between documents and between words each based on the other to exploit the dual nature of the problem. An illustrative example is also used to demonstrate the advantage of using χ-Sim as opposed to classical similarity measures. A theoretical interpretation of the algorithm was also provided that uses the concept of weighted higher orders paths. The resulting discussion on this interpretation provides an intuition of the working of the algorithm and enabled us to explore variants of the algorithm. Finally, an extension of the algorithm to the supervised learning task was also proposed in this chapter.

Of course, we need to validate the hypothesis proposed in this chapter and demonstrate the quality of the results experimentally on actual datasets. In the next chapter, we provide the experimental results used to validate the proposed algorithm and demonstrate the improvement using our method.

Chapter 4
Application to Text Mining

In the previous chapter, we proposed a new measure for measuring similarity values between objects that takes into account the similarity between their features. An algorithm was presented to compute this similarity measure that renders it in the same time complexity as traditional similarity measures such as Cosine, Euclidean, etc. This chapter provides the empirical evaluation of the proposed algorithm on the task of text clustering and categorization. We start by describing the pre-processing performed, the datasets used, and the evaluation criteria used to judge the quality of the results. Then, we evaluate the behavior of the proposed algorithm by varying different parameters on both simulated and real world datasets, and also the results obtained for both the unsupervised and the supervised learning problem. This chapter also serves as an empirical comparison between our proposed method and various existing algorithms on text clustering and categorization.

4.1. Introduction

The similarity measure proposed in the previous chapter exploits the structural relationship within the data to compare two instances, such as two documents in a corpus. This opens the way for many prospective application of the algorithm. We consider χ-Sim primarily as a similarity measure that could be used in applications such as clustering. We shall evaluate the performance of χ-Sim on 2 applications in text mining in this chapter and on bio-informatics data in the next chapter (Chapter 5).

Text mining is a domain that attempts to gather meaningful information from large amounts of texts. It may (loosely) be characterized as the process of analyzing textual data with the intention of extracting information. With the advent of the web age, text generation has grown many folds and it has, thus, become essential to find ways to analyze text automatically by computers. As such, the term "text mining" usually refers to the discovery of new, previously unknown, information by automatically extracting information from a collection of text sources.

Various applications of text mining are available such as (among many others): *document clustering* or the unsupervised grouping of documents, and *text categorization* (or *text classification*) which is the assignment of natural language documents to predefined categories according to their content (Sebastiani 2002). These two tasks have found many applications in text mining such as *document retrieval* (sometimes referred to as *information retrieval*) or the task of identifying and returning the most suitable documents, from a collection of documents, based on a user's need, usually expressed as a form of a query, language identification or categorizing documents based on their language, etc. Thus, both clustering and categorization tasks are central to text mining applications and are based on the concept of similarity between documents, which is also the focus of our proposed similarity measure. Therefore, we choose these two tasks as an application to evaluate our (co-)similarity measure.

This chapter provides a practical analysis and comparative evaluation of our proposed co-similarity measure. We will be using the Vector Space Model (section 2.1.1) for data representation. Using this model, a similarity measure usually has the following properties

- The similarity score between two documents (or words) increases as the number of common terms between them increase, and
- The score of a similarity measure also depends on the weight of the terms (in common) between any pair of documents. The higher the weights, the higher will be the similarity measure.

We can relate these properties to the popular similarity measures used in measuring document similarities, such as the Cosine similarity measure. Both the Cosine similarity measure and χ-Sim measure exhibits these properties. A further property of our measure, however, is that

- The similarity score generated between two documents depends on the relationship between their words and vice versa. Thus, the relationship between documents and between words in a dataset can lead to a similarity score between two documents even if they share little to no terms between themselves.

It is this property that forms the central idea of our proposed similarity measure. It reflects our idea that analyzing relationships in a dataset via higher-order co-occurrences can provide meaningful information when comparing a pair of documents and thus helps to compute the corresponding similarity score.

The experimentation reported in this chapter has been applied to these two tasks of text clustering and text categorization on textual documents, the set of which constitutes a collection, or a *corpus*. We emphasize here that χ-Sim measure is a (co-)similarity measure that generates a proximity matrix, which can be used in several clustering/classification algorithms, for instance the Agglomerative Hierarchical clustering(section 2.3.1) and the *k* Nearest Neighbors (section 3.7.5) respectively. Evaluation measures, such as MAP and NMI (discussed in section 4.2), allows us to quantify the quality of results obtained by the task of clustering and categorization by different methods. We shall utilize these measures to evaluate our proposed algorithm on the different datasets and compare them with the results of other algorithms.

Before comparing our clustering results with others, however, we first need to analyze our proposed algorithm on the various datasets so as to verify some of our hypotheses and adapt it for the document clustering and categorization task. Our algorithm contains two parameters — the number of iterations and the pruning parameters.

In order to study these two parameters and their effect on the results, we first recall the hypothesis we made in the previous chapter (sections 3.4 and section 3.5.5 respectively) as follows

- The number of iterations needed to exploit meaningful higher order relations is relatively small and further iterations do not improve the quality of the task,
- Pruning can be used to minimize the effect of less informative words and increase the quality of the clustering task.

Therefore, we start by studying the effect of the number of iterations and the pruning factor on the clustering result on both real and synthetic data.

4.2. Validation Criteria

It is usually hard to make a reliable evaluation of the clustering results, partly because the concept of a "cluster" could be quite subjective. In a study by Mackskassy et al. (Macskassy et al. 1998), ten people performed a manual clustering of a few small set of documents that were retrieved by a search engine and, interestingly enough, it was observed that no two people had made a clustering that was similar.

Measuring the quality of a clustering, however, is fundamental to the clustering task since every clustering algorithm will turn out some partitioning of the data. The objective then is to measure whether this partitioning corresponds to something meaningful or not. This need arises naturally since the user needs to determine whether this clustering could be relied upon for further data analysis and/or applications. Moreover, measuring the quality of the clustering results can also help in deciding which clustering model should be used on a particular type of problem. As a result, several quality measurement methods have been proposed in the literature.

From a theoretical perspective, the evaluation of a clustering algorithm is still an open issue and a current subject of research (Ackerman and Ben-David 2008). Kleinberg (Kleinberg 2003) suggested a set of axioms for clustering algorithms that should be "independent of any particular algorithm, objective function, or generative data model". Kleinberg's goal was to develop a theory of clustering, which can then be used to asses any the clustering algorithm. He proposed 3 axioms, *function scale invariance, function consistency and function richness* (see (Ackerman and Ben-David 2008) for details), that should form a theoretical background for any clustering algorithm and then went on to prove the impossibility of any clustering algorithm satisfying them at the same time. Ackerman and Ben-David (Ackerman and Ben-David 2008) adapted these 3 axioms as a basis of quality measures for validating clustering outputs rather than the algorithm itself. They argue that any quality measure for clustering should satisfy these 3 axioms. Our interest here, however, is not on the general theory of clustering or their quality measures but rather we wish to evaluate the performance of our similarity measure on practical clustering task (as mentioned before, we are using a well known clustering algorithm for the actual clustering process).

As mentioned in (Candillier et al. 2006), there are 4 basic ways to validate a clustering result, as follows:

- Use of artificial data sets to evaluate the result of clustering, since the desired groups are known. The advantage of this is that different kinds of artificial data sets can be generated to evaluate the algorithm(s)

under a range of different conditions. Synthetic data, however, does not always corresponds to real world data and hence, may not be sufficient evaluation criteria.

- Use supervised data whose category labels are known, ignore the labels while clustering and evaluate how the clustering results retrieve the known categories. In this case, the groupings resulting from the clustering method is directly compared to known groupings and thus, the usefulness of the clustering can be measured.
- Work with experts that will validate the groupings to see if meaningful groupings have been discovered. This is particularly useful when pre-labeled groupings are not available and when several groupings could be considered as a meaningful. The drawback, of course, is that it is time consuming and requires considerable human resources.
- Use internal quality measures that measure the cluster separation. Using these measures, however, is rather subjective since they are based on some pre-conceived notion of what is a good clustering such as some geometrical/mathematical properties of a class (spherical, etc).

Usually, internal measures are used to verify the cluster hypothesis i.e. when clustering a set of objects using a similarity measure, the objects belonging to the same category are drawn closer together (more similar) and are rendered farther apart then objects that belong to a different category. Hence, it is reasonable to evaluate the similarity measures obtained from a clustering algorithm to see how *cohesive* the elements of a given cluster are and how *well separated* clusters they are from elements of other clusters.

Internal measures usually try to quantify how well the clustering result achieves the inherent hypothesis of clustering (section 1.1 and 2.3), which measure the cohesiveness and separation of elements in a clustering. Two such measures are the *average intra-cluster similarity* and the *average inter-cluster similarity* defined as follows,

(4.1) $$\phi_{\text{intra}}(\hat{\mathrm{X}}) = \sum_{i \in \hat{\mathrm{X}}} \frac{1}{\left|\hat{x}_i\right|^2} \left(\sum_{d_j \in \hat{x}_i} \sum_{d_k \in \hat{x}_i} Sim(d_j, d_k) \right)$$

(4.2) $$\phi_{\text{inter}}(\hat{\mathrm{X}}) = \sum_{i \in \hat{\mathrm{X}}} \sum_{\substack{j \in \hat{\mathrm{X}} \\ j \neq i}} \frac{1}{\left|\hat{x}_i\right| \left|\hat{x}_j\right|} \left(\sum_{d_k \in \hat{x}_i} \sum_{d_l \in \hat{x}_{ij}} Sim(d_k, d_l) \right)$$

where $\hat{X}$ is the set of clustering on a corpus X having documents d_i ($1 \leq i \leq |X|$) and $\hat{x}_1, \hat{x}_2, \ldots$ are the clusters.

The value in equation (4.1) measures the average cohesiveness among documents of the same cluster. The higher this value, the more the clustering reflects the underlying class structures of the documents and the lower this value, the lower the cohesiveness. The second value reflects the average inter cluster similarity. Ideally, we would like this value to be low since it means that documents belonging to different clusters are less similar.

From a practical perspective, however, a direct comparison using these measures between two algorithms may not always make sense since the absolute values of similarities usually depend on the nature of the similarity measure and might be greatly affected by such things as the normalization factor used in the similarity measure. One may try to overcome this by taking the ratio between these measures. However, some of these indices may be biased

if the clustering algorithm uses the same values in its objective function thus favoring a certain clustering method. Furthermore, an internal quality measure does not tell us whether the clustering result is practically useful or not. For example, being able to discover groups of documents that correspond to similar topics is more useful from a practical point of view.

In this chapter, we will focus the first methods of validation as mentioned in (Candillier et al. 2006) list above. We evaluate our results on both a synthetically generated datasets whose classes are known and using corpora whose document classes. For the clustering task, the classes are ignored during clustering. We validation is then performed using external quality measures which take advantage of the externally available manual categorization. These external measures focus on evaluating whether or not the clustering is useful i.e. reflects our knowledge about the class. Moreover, since we only compare our algorithm for single labeled documents (i.e. documents belong to only one class), the evaluation measures discussed are also confined to this criteria.

We discuss two popular evaluation measures that have been used in the literature. Before we formulate the two measures, we present what is known as a *confusion matrix*. A confusion matrix is simply a contingency table. A confusion table for binary classification is shown in Table 4-1 below.

Table 4-1 A confusion matrix

	Predicted Positive	Predicted Negative
Actual Positive	*a*	*b*
Actual Negative	*c*	*d*

Let I denote the set of the texts which are actually positive, and J denote the set of the texts which are predicted as positive by a classifier. Table 4-1 means that *a* is the size of $\mathrm{I} \cap \mathrm{J}$, *b* is the size of $\mathrm{I} \cap \overline{\mathrm{J}}$, *c* the size of $\overline{\mathrm{I}} \cap \mathrm{J}$, and *d* is the size of $\overline{\mathrm{I}} \cap \overline{\mathrm{J}}$.

Micro-Average Precision

Micro-Averaged Precision (MAP) is defined as the percentage of correctly classified documents, and is generally used to evaluate single-label text clustering and classification tasks (see, for instance,(Nigam et al. 2000; Dhillon et al. 2003; Long et al. 2005; Bisson and F. Hussain 2008)). It is represented as a real value between 0 and 1. Furthermore, it can be shown that,

Accuracy = micro-averaged F1 = micro-averaged precision = micro-averaged Recall

since each document can be correctly classified or not in only one class, so the number of *false positives* is the same as the number of *false negatives*. Therefore, we define micro-averaged precision in our case as

(4.3) $$MAP = \frac{(a+d)}{(a+b+c+d)}$$

It should be noted here that there is no prior way of determining which cluster belongs to which document class. Hence, an optimal assignment algorithm, such as the Hungarian algorithm (Munkres 1957), is used to map the predicted clusters to the actual clusters resulting in the best MAP score.

The micro-averaged precision measure is used because it is intuitive in nature. However, MAP could potentially lead us to a wrong impression of the quality of the results when the numbers of instances in positive and negative classes are not well balanced. For example, a trivial prediction that all points are negative (or positive) may achieve a high precision in certain dataset. Therefore, a second index is also defined, which we consider below.

Normalized Mutual Index (NMI)

The Normalized Mutual Index (NMI) (Banerjee and Ghosh 2002) is used as a measure of cluster quality especially when number of clusters is small and cluster size is uneven. For example 2 clusters of 50 documents each can have a clustering algorithm assign 95 documents to one and only 5 to the other yet having an accuracy value of 55 percent. NMI is defined as the mutual information between the cluster assignment and the true labeling of the dataset normalized by the mean of the maximum possible entropy of the empirical marginal. NMI is calculated as follows

(4.4) $$NMI(X,Y) = \frac{I(X,Y)}{\sqrt{H(X)H(Y)}}$$

Where X is a random variable for cluster assignment, Y is a random variable for actual labels, and $I(,)$ and $H()$ denotes mutual information and entropy respectively. The value of NMI ranges from 0 to 1 with a larger value signifying a better clustering and a value of 1 defining perfectly retrieving the original clusters.

Statistical Significance Tests

One way of comparing the results given by two models, or in our case two clustering algorithms, is to compare some quality measure such as MAP as discussed above. However, if we would like to check whether the difference of the measures is statistically significant or not, we can use the so-called statistical tests for that purpose.

At the center of significance test is the *p*-value (Y. Yang and X. Liu 1999). The p-value is the probability that, given the data, the statistics computed under a null hypothesis is realized. In other words, *p*-value is the smallest significance-level with which the null hypothesis can be rejected. If the *p*-value is smaller than the significance-level (usually 0.05 is used), then the null hypothesis is rejected.

Suppose we wish to compare two models – model A and model B in terms of their predictions. One usual null hypothesis could be that "the accuracy measure given by A and the accuracy measure by B have an identical distribution". This implies that the accuracy values given by A and B comes from some identical distribution and any observed difference could be because of randomly sampling for A and B from this distribution. If the *p*-value is small enough (smaller than the significance-level being tested), the null hypothesis is rejected. That is, the two accuracy values of A and B have different distributions. In our case, since these distributions correspond to accuracy

values of the predicted clusters, it signifies that one model is superior to the other. Therefore, small *p*-values are desired to reject the null hypothesis.

Several such significance tests could be used to test different hypotheses. For our comparison of χ-Sim with different other algorithms, we use the *t*-test at significance level of 0.05. The *t*-test tries to determine whether two samples from a normal distribution (say in model A and model B) could have the same mean when the standard deviation is unknown but assumed equal. More precisely, for each dataset we use the predicted cluster value that is correctly classified by the different algorithms to form such a distribution and test whether these distributions could have been generated from a normal distribution with the same mean value.

4.3. Datasets

In this section, we describe the datasets used for our experimentation. These datasets have been widely used for evaluation of both text clustering and classification. We used subsets of the Newsgroup and Classic3 datasets for evaluating our χ-Sim algorithm for document clustering and subsets of Newsgroup, Reuters and Lingspam for supervised document categorization. The datasets are described below. In addition to these, we also used a synthetic dataset which is detailed in section 4.3.5.

4.3.1. Newsgroup Dataset

Table 4-2 The 20-Newsgroup dataset according to subject matter

comp.graphics comp.os.ms-windows.misc comp.sys.ibm.pc.hardware comp.sys.mac.hardware comp.windows.x	rec.autos rec.motorcycles rec.sport.baseball rec.sport.hockey	sci.crypt sci.electronics sci.med sci.space
misc.forsale	talk.politics.misc talk.politics.guns talk.politics.mideast	talk.religion.misc alt.atheism soc.religion.christian

The 20-Newsgroup dataset (NG20)[16] consists of approximately 20,000 newsgroup articles that have been collected evenly from 20 different Usenet groups. Many of the newsgroups share similar topics and about 4.5% of the documents are cross-posted making the boundaries between some newsgroups fuzzy. The texts in 20-newsgroup are

[16] http://people.csail.mit.edu/jrennie/20Newsgroups/

not allowed to have multiple category labels. In addition, the category set has a hierarchical structure (e.g. "sci.crypt", "sci.electronics", "sci.med" and "sci.space" are subcategories of "sci (science)").

Table 4-2 shows the categories in 20-newsgroup. There is no fixed way to split 20-newsgroup into a training set and a test set. This table also shows that the sizes of categories are relatively uniform. We used the "by date" version of the data that is sorted by date; duplicates and some headers removed resulting in 18846 documents.

4.3.2. Reuters-21578 Dataset

The Reuters-21578[17] is a collection of documents that appeared on Reuters newswire in 1987. All the documents contained in this collection appeared on the Reuters newswire and were manually classified by personnel from Reuters Ltd. This collection is quite skewed, with documents very unevenly distributed among different classes.

Reuters-215781 is probably the most widely used data set for text categorization. Although the original data set contains 21578 texts, researchers use a data-splitting method to extract a training set and a test set. The most popular data-splitting method is ModAptesplit, which extracts 9603 training texts and 3023 test texts. However, there are still unsuitable texts in the 9603 training texts. We combine the training and text dataset and used a subset of this dataset which contains 3 classes – Acq, crude and earn. Furthermore, we use only documents that occur in a single category. The resulting dataset is shown in Table 4-3 below.

Table 4-3 A subset of the Reuters dataset using the ModApte split

	No. of training documents	No. of Test documents	Total
earn	2725	1051	3776
acq	1490	644	2134
crude	353	164	517

4.3.3. Classic3 Dataset

The Classic3 dataset comes from the SMART collection[18]. It contains documents from 3 collections – the MEDLINE collection containing 1033 documents which are abstracts from medical research papers, the CISI collection containing 1460 documents which are abstracts from papers on information retrieval, and the CRANFIELD collection containing 1398 documents abstracts relating to aerodynamics. The dataset is summarized in Table 4-4.

[17] http://www.daviddlewis.com/resources/testcollections/reuters21578/

[18] ftp://ftp.cs.cornell.edu/pub/smart

Table 4-4 Summary of Classic3 dataset

	MED	CISI	CRAN
No. of documents	1033	1460	1398

4.3.4. LINGSPAM Dataset

Ling-Spam is a mixture of spam messages, and legitimate messages sent via the Linguist list, a moderated mailing list about the science and profession of linguistics. The corpus consists of 2893 messages:

- 2412 legitimate messages, obtained by randomly downloading digests from the list's archives, breaking the digests into their messages, and removing text added by the list's server.
- 481 spam messages, received by one of the authors of (Sakkis et al. 2003). Attachments, HTML tags, and duplicate spam messages received on the same day have not been included.

The Linguist messages are less topic-specific than one might expect. For example, they contain job postings, software availability announcements, and even flame-like responses.

4.3.5. Synthetic Dataset

Text corpora datasets, such as those described above, are good in evaluating the performance of an algorithm on real world applications. Such datasets, however, have a fixed level of difficulty for the clustering task which is determined by the domain of the corpus and the composition of the various classes of documents. In order to explore the behavior of our similarity measure at different levels of difficulty of the clustering problem, we also used a synthetic dataset whose parameters can be tuned to make the task of finding clusters progressively difficult.

As discussed in section 4.2, synthetic datasets are a useful way to validate the clustering since we know beforehand the clusters in the dataset. A good synthetic dataset, in our case, would be one that tries to mimic (as much as possible) the characteristics of the real dataset. This, however, is not as easy task for text corpora which are rather complex. It is well known (Zipf 1949) that the frequency of a word in natural languages such as English follows the inverse power law of its rank or the its position according to frequency of occurrence (later generalization such as by Mandelbrot, etc also exist; see for example (W. Li 1992)). This would enable us to generate probabilities of a word at a given rank, for example to mimic the statistical characteristics of a given natural language text. However, it does not tell us the distribution of words amongst the various classes in the text, or the correlation between words, etc. For example, a document talking about aviation might contain several terms related to aeronautics, avionics and planes together such as *cockpit, in-flight entertainment*, *rudder*, etc. These terms will usually show a high level of correlation and occur together in certain types of documents. Modeling such a dataset is not a trivial task.

As a result, we simplified the problem and used a simpler approach. Our model is similar to the one used in (Long et al. 2005) that generates a bipartite graph and randomly assign links between the two set of vertices, using

for example an exponential distribution. Instead of using an exponential distribution, however, we generate a matrix (corresponding to the bipartite graph) with the desired number of document (row) and word (column) clusters and fill the matrix while respecting certain constraints as described below.

FIGURE 4.1 gives a graphical representation of the synthetic dataset matrix. A co-cluster is represented by the pair $(\hat{x}_i \hat{y}_i)$ whose elements belong to word cluster $\hat{y}_i$ and document cluster $\hat{x}_i$. Elements in $(\hat{x}_i \hat{y}_j)$ with $i \neq j$ represent out-of-co-cluster elements. An ideal case will be to have (1) non-zero elements in the co-clusters (shown in grey shade), and (2) a value of 0 otherwise in the matrix. Essentially, we vary these two conditions to change the complexity of the clustering problem. As discussed previously, χ-Sim tries to overcome the problem of sparse data in high dimensionality by measuring the structural relationship between documents and words to estimate the similarity values. We represent the percentage of non-zero elements in the matrix by a parameter called *density*. By reducing the *density* in the matrix, it becomes increasingly harder for classical similarity measure such as Cosine (and χ-Sim at the first iteration) to cluster the data since fewer terms would be shared between documents.

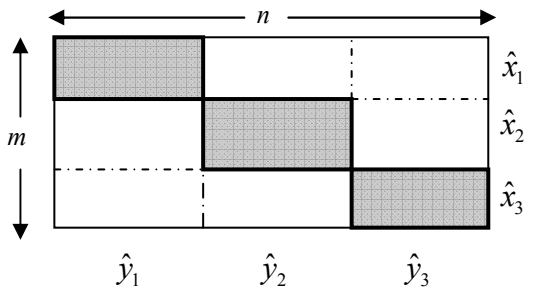


FIGURE 4.1 Model representing the synthetic dataset

A second parameter that controls the complexity of the clustering problem is the number of non-zero elements outside of the co-clusters as mentioned earlier. We control this by a parameter called *overlap*. Overlap denotes the percentage of non-zero elements in a document cluster that are also in its corresponding word cluster i.e. they form diagonal co-clusters. The higher the overlap, the more 'pure' the clusters while a low overlap signify the presence of many "unspecific" words in the sense that these are not good features allowing to predict the cluster and makes the clusters rather fuzzy. By increasing this parameter, we bring documents that belong to different clusters closer together. We would like to analyze the effect of our pruning parameter (section 3.5.5) in trying to minimize this effect.

In summary, the generator uses the following parameters- (i) the number of rows *m* (with each row representing a document), (ii) the number of columns *n* (each column representing a word), (iii) the number of clusters, (iv) the *density* of the matrix in terms of percentage of non-zero values in the matrix, and (v) *overlap* in terms of percentage of non-zero elements occurring inside the co-clusters. We would like to analyze the behavior of the number of iterations in χ-Sim to study the additional benefits of taking into account higher order co-occurrences to overcome this problem. We fix the number of co-clusters to 3 with each co-cluster having 100 documents and 100 words. The two parameters–*density* and *overlap*– can then be used to create different datasets of varying complexity. Both

density and *overlap* are defined as percentages or corresponding values between 0 and 1. A density of 10% (or 0.1), for example, corresponds to (100*100)*0.1 or 1000 non-zero elements in the matrix. The elements are equally distributed among the document clusters, thus each of $\hat{x}_i$ contains 1000/3 non-zero elements. Within the document cluster, the distribution of words is determined by the overlap parameter and (1000/3)*(*overlap*) are contained in the co-cluster ($\hat{x}_i\hat{y}_i$) while the rest are distributed in ($\hat{x}_i\hat{y}_j$) where $i \neq j$.

The actual data points are generated randomly, but guarantee that the density and overlap percentages hold. In order to minimize any statistical bias, we generate 10 datasets at each such pair and report the average and standard error for each.

4.4. Data Pre-processing

Data pre-processing is an essential and very important phase in document clustering/categorization. In the Vector Space Model (section 2.1.1), a text is represented as a vector whose components are the frequencies of words. Without any prior treatment, using the Bag-of-Words (BOW) approach all words are indexed in the document-word matrix (Banerjee et al. 2005; Shafiei and Milios 2006). This leads to a high dimensionality even for a small corpus. However, not all of these words are helpful in aiding to identify the different classes in a corpus. In general, 40-50% of the total numbers of words in a document consist of such words (G. Salton 1983).

While the basic text clustering and categorization algorithms can be considered as general purpose learning algorithms, the growing interest on textual information has given rise to more dedicated research on text mining. As a result, standard text pre-processing steps have been developed in the literature which are almost always employed before undertaking the task of text clustering or categorization and which was also adopted in this thesis. We discuss them in the following subsections.

4.4.1. Stop Word Removal

This is the first step in the pre-processing task. Stop words are words that occur frequently in a large text set and function words such as '*and*', '*or*', '*to*', '*the*', '*when*', etc that do not contain any information regarding the category of the document. The document is parsed through and each word in the document is compared to a list of such stop words. Such stop-word lists can be constructed for each language, for example the most commonly used stop-words for English is the SMART list (ftp://ftp.cs.cornell.edu/pub/smart/english.stop) of stop words which contains 529 such commonly occurring words in the English language. If the word is a member of the stop word list, it is simply discarded. We used the software package, Rainbow[19], for the text parsing which uses the SMART list of stop words.

[19] http://www.cs.cmu.edu/~mccallum/bow/rainbow/

4.4.2. Stemming

It is the pruning of word suffices that play only grammatical role in the language such as the words "*runner*" and "*running*" mentioned above. Stemming is a standard way of reducing various forms of a word into a basic or stemmed word. In our example, both "*runner*" and "*running*" could be replaced by the word "*run*". Sometimes, the stemmed word doesn't exist in the language such as '*cycle'* and '*cycling'* may be reduced to '*cycl'*, but it forms the basic part of the word from which different morphological meanings of the word are constructed. Such rules are normally created by hand for a given language. Stemming reduces statistical sparseness since different forms or a word are mapped into one word and counted together. However, care must be taken to avoid bias in the mapping of unambiguous words such as '*mining*' into an ambiguous word, '*mine'*.

Porter's stemming algorithm (Porter 1997) has been popularly used in text mining applications to prune suffices and convert words to their stemmed form. We also used this stemming algorithm, which is provided with the parsing software package, Rainbow.

4.4.3. Feature Selection

Feature selection is a common pre-processing step common to many pattern recognition algorithms and also adopted in text clustering and classification. Feature selection is a process commonly used in machine learning in general and text mining in particular wherein a subset of the features describing the data is selected prior to the application of a learning algorithm. The aim of feature selection is to select a subset of features that contains the most information about the data thus focusing only on relevant and informative data for use in text mining. This subset of features ideally retains much of the original information of the original data and the remaining, less important, dimensions are discarded. Moreover, most text corpora contain a huge set of vocabulary (features) that may render the application of many machine learning algorithms to be computationally expensive. Thus, feature selection also helps in reducing the computational cost of applying a learning algorithm.

Two types of feature selection mechanisms are typically used depending on the type of learning environment – unsupervised feature selection that do not require any external knowledge; and, supervised feature selection that normally uses a human input in the form of manual labeling of the categories on a training dataset used to find the most discriminatory features. Both these types of feature selection aim to remove the non-informative terms according to the corpus statistics. Typical examples include discarding words based on frequency, for example words that occur in fewer than x% of the documents or those that occur in more than 50% of the documents in a corpus, for the unsupervised case and selecting the top k words based on Mutual Information (MI) or Information Gain (IG) for the supervised case. See (T. Liu et al. 2003; Mladenic 1998; Y. Yang and Pedersen 1997) for a survey of most commonly used feature selection methods in text mining.

The feature selection steps taken in generating the data for clustering and categorization are discussed in sections 4.5.1 and 4.6.2 respectively.

4.5. Document Clustering

4.5.1. Experimental Settings

To evaluate our proposed similarity measure, we perform a series of tests on both the synthetic and real datasets described in the previous section. We extracted several subsets from these datasets described that have been widely used in the literature and provide a benchmark for document clustering, thus allowing us to compare our results with those reported in the literature.

We created 6 subsets of the 20-Newsgroup dataset namely M2, M5 and M10 used in (Dhillon et al. 2003; Long et al. 2005; Bisson and F. Hussain 2008) that contains 2, 5 and 10 newsgroup categories respectively, as well as the subsets NG1, NG2, and NG3 used by (Long et al. 2006; Bisson and F. Hussain 2008) that contains 2, 5 and 8 newsgroups respectively. Details about these various subsets are given in Table 4-5. For instance, the M10 dataset has been created by randomly selecting 50 documents (from about 1000) for each of the 10 subtopics, thus forming a subset with 500 documents.

Table 4-5 Summary of the real text dataset used in our experiments

Subset	Newsgroup included	docs/cluster	# of docs
M2	Talk.politics.mideast, talk.politics.misc	250	500
M5	comp.graphics, rec.motorcycles,rec.sport.baseball, sci.space , talk.politics.mideast.	100	500
M10	alt.atheism, comp.sys.mac.hardware, misc.forsale, rec.autos, rec.sport.hockey, sci.crypt, sci.electronics, sci.med, sci.space, talk.politics.gun.	50	500
NG1	rec.sports.baseball, rec.sports.hockey	200	400
NG2	comp.os.ms-windows.misc, comp.windows.x,rec.motorcycles,sci.crypt,sci.space	200	1000
NG3	comp.os.ms-windows.misc, comp.windows.x, misc.forsale, rec.motorcycles, sci.crypt, sci.space, talk.politics.mideast, talk.religion.misc	200	1600
Classic3	MEDLINE, CRANFIELD, CISI	1033,1460,1400	3893

All datasets underwent a pre-processing stage. We randomly choose documents from the different classes to generate the dataset considering documents that contains at least two words. Some of the datasets contained subject lines which contains words that might contain information of the newsgroup topic and were therefore removed. All words were stemmed using the standard Porter's stemming algorithm (Porter 1997) and we removed stop words from the corpus vocabulary. The pre-processing steps described above were done as they have been popularly employed in the literature for generating datasets. Therefore, we used the same datasets and pre-processing steps to make our comparison easier with those reported in the literature.

Feature selection was performed using different strategies. As mentioned earlier (section 4.4.3), the feature

selection step aims at reducing the number of words used in order to improve the results by removing words that may not be useful in distinguishing between the different clusters of documents. Mutual information based feature selection has been commonly used in text mining to select a set of most informative features. There are usually two kinds of feature selection based on mutual information —

- Unsupervised Mutual Information (UMI) that assigns a score to each term in the document by considering its mutual information with the document as described in (N. Slonim and N. Tishby 2000). The terms are then ranked according to their scores and the top *k* terms are selected.
- Supervised Mutual Information (SMI) that uses the information available from the class labels of the documents to assign mutual information score to terms as described in (Y. Yang and Pedersen 1997).

Selecting features based on their mutual information with the clusters clearly is a supervised technique since it requires prior knowledge of document categories. This method can be controversial with regards to document clustering as it introduces some bias since it eases the problem by building well-separated clusters. Nonetheless, it provides an interesting scenario where topics could be considered more distinct and many words are peculiar to a single topic. Thus, we generate the datasets in Table 4-5 by selecting the top 2000 words based on mutual information. We refer to this as a *knowledge intensive approach* to emphasize the fact that selected terms in this dataset contain more information about the categories. We will denote these datasets by the subscript SMI (Supervised Mutual Information) to specify that these datasets have been generated using a supervised mutual information based feature selection.

In real applications, however, it is impossible to use a supervised feature selection for unsupervised learning. Thus, to explore the potential effects of this bias, we also generated a separate dataset by using unsupervised feature selection methods. Firstly, we use an unsupervised version of mutual information criteria that only takes into account the mutual information of a word with a document rather than with its class. We will denote the dataset generated by this method with the subscript, UMI (Unsupervised Mutual Information). As a second method, we used representative feature subset that was selected using the Partitioning Around Medoids (PAM) (Kaufman and Rousseeuw 1990) algorithm. This algorithm is readily available, more robust to outliers than the standard *k*-means algorithm and less sensitive to initial values as well. The procedure is the following– first, we remove words appearing in just one document from the corpus as they do not provide information about the overall cluster. Then, we run PAM to get 2000 feature classes corresponding to a selection of 2000 words. We used the implementation of PAM provided in the R-Project[20] with the Euclidean distance and setting the *do_swap* parameter to *false*, to speed up the process. We denote these datasets using the subscript, PAM. We refer to the later two approaches as *unsupervised approaches*. As will be seen later in this chapter (section 4.5.3 and 4.5.4), where we will make a comparison of clustering accuracies between datasets generated using mutual information and using PAM, the unsupervised feature selection highlights the bias introduced by a supervised feature selection technique and will emphasize the importance of our *pruning* parameter.

[20] http://www.r-project.org/

To overcome any potential bias in our evaluation, we repeat this process 10 times, each time randomly choosing documents from the subtopics to generate 10 different datasets for each of M2, M5, M10, NG1, NG2, and NG3. Unless stated otherwise, the results presented in this chapter correspond to the mean and standard deviation over these 10 randomly generated datasets for each dataset.

To perform a clustering of the documents from the document similarity matrix, **R**, we used an Agglomerative Hierarchical Clustering (AHC) method on the similarity matrix with Ward's linkage, as it was found to perform better among the different AHC methods. As Ward's linkage usually takes a distance matrix as its input, we convert our proximity matrix into a distance matrix[21] by taking its complement. Since the highest similarity value defined using the χ-Sim algorithm is 1 and the lowest 0, we subtract the proximity values from 1 to form the corresponding distance matrix. We used MatLab® to implement χ-Sim and to perform AHC on the resulting distance matrices.

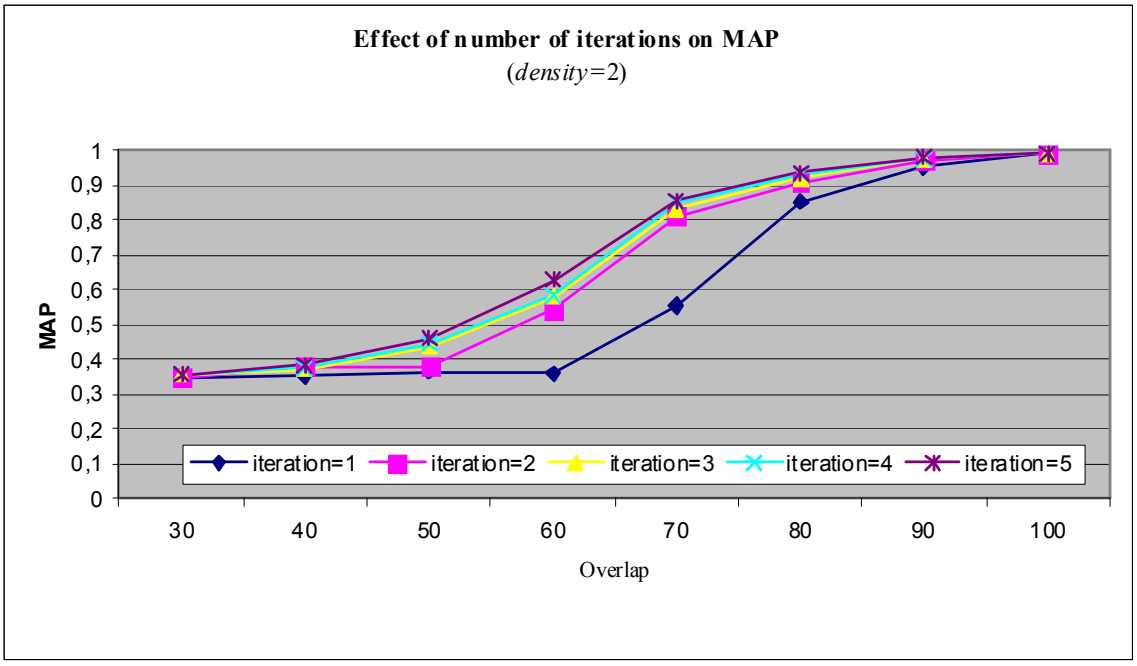
4.5.2. Effect of Iteration on χ-Sim

In our first test, we measure the effect of successive iterations on the χ-Sim algorithm. Our aim here is to verify our earlier hypothesis that using higher order co-occurrences relations enhances the similarity value between documents of the same category, and hence provide a better measure of similarity. This test also forms as a baseline test for our algorithm since using higher-order relationships is core to our approach.

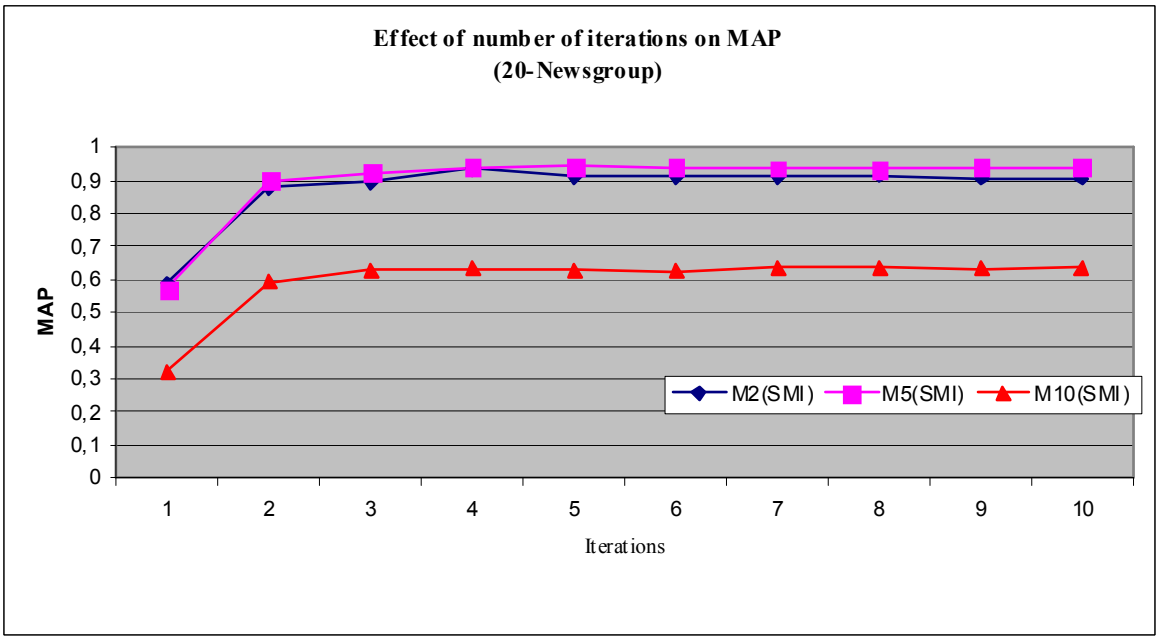
Using Synthetic Dataset

The MAP score of χ-Sim at different iterations is shown in FIGURE 4.2(a) below. The values corresponding to the figures can be found in Appendix IV. The figure shows the various measures on a sample synthetic dataset with density=2%. Since the dataset has 3 document categories, an overlap of 30% roughly corresponds to random distribution with no distinct classes. Therefore, the clustering accuracy at that value corresponds to a random clustering and show little variation with the number of iterations performed. Similarly, a high value of >90% corresponds to almost perfect co-clusters which are easy to discover. The interesting points correspond to overlaps between 50 and 80 which contain sufficient but not too much information about the cluster. As can be seen from the figure, using higher order co-occurrences significantly, in this interval, improves the accuracy of the results. The greatest increase in accuracy is from simple co-occurrences to 2nd order co-occurrences. As mentioned in the previous chapter, two documents belonging to the same document cluster need not share the same set of words but a collection of documents of the same cluster is assumed to share a collection of words from a word cluster, even though individual documents might only span a subset of these words. Using second (and higher) order co-occurrences can help alleviate this problem, which is caused as a result of sparseness of the documents.

[21] The term "distance matrix" here is used to denote the notion of a dissimilarity matrix rather than one based on a strict distance metric.



(a)



(b)

FIGURE 4.2 Effect of number of iterations on (a) values of *overlap* for *density* =2 on the synthetic dataset; and (b) 3 datasets of the 20-Newsgroup dataset

Using Real Dataset

A similar effect can be seen on the real datasets generated from the 20-Newsgroup dataset as shown in FIGURE 4.2 (b). Note that the real datasets usually corresponds to a lower *density* (typically <0.1%) and, therefore, the impact of computing higher order co-occurrences is much higher. With a low density, there are fewer shared words amongst documents and hence using only co-occurrence terms results in a lower MAP score. Higher order co-occurrences exploit structural relationships and helps in reducing the effects of data sparseness. As can be seen from the figure, the increase in accuracy is much higher as a result of going from simple co-occurrences to 2nd order co-occurrences. There are subsequent, but less significant, increases in the MAP score as the number of iterations is increased to 3, 4, 5 …,

Moreover, it can be seen that the accuracy values almost stabilizes for iterations greater than 4 which is consistent with previous studies (Lemaire and Denhière 2006; Kontostathis and Pottenger 2006) as well as our theoretical interpretation of iterations, as discussed in section 3.4. For the rest of this chapter, unless otherwise stated, we will use the number of iterations to be 4.

4.5.3. Effect of Pruning

Using Synthetic Dataset

As we saw in section 3.5.5, the pruning threshold was introduced as an approximate solution to minimize the effect of *empty words* (words that are less discriminative to the document classes) since one would expect such words to result in lower similarity values. In this test, we examine the effect of the pruning parameter against the measured accuracy of the algorithm. As previously, we first test the pruning parameter on the synthetic dataset with density=2%. We run the χ-Sim algorithm on different datasets generated by varying the overlap from 30 to 100 and measuring the accuracy value of each dataset at different pruning parameter values ranging from 0 to 100. A value of 0 represents no pruning while a value of 100 means all similarity values are pruned. The results are shown in FIGURE 4.3.

From the figure, it can be observed that the pruning parameter shows some correlation with the "degree of difficulty" (overlap) of the data. For high overlap values, for instance greater than 80%, the co-clusters are relatively 'pure' and are easily discoverable even with direct co-occurrences as seen in the previous section. We explain this by considering that the similarity values generated by higher-order co-occurrences, will be much higher for document (or word) pairs that belong to the same category than similarity values because of overlap between document (or word) pairs that belong to different categories. Similarly, for very low overlap values, for instance those less than 50%, the accuracy measure does not show much variation since the clusters are simply too overlapped to be recovered effectively.

As in the previous section, we will focus on the overlap range between the values of 50 and 80 as they correspond to the more interesting results. Firstly, we observe that using pruning does indeed increase the accuracy of the clustering results. For example, for an overlap value of 55, using pruning enhances the maximum accuracy value to 0.62% as opposed to 0.48 without pruning. Similar results can also be seen for other values of overlap. It

can also be observed that the pruning threshold has a fairly large *plateau* between the values of 20 and 80 for most of the datasets where the precision value is increased as compared to no pruning. A smaller value for the pruning parameter (<20%) will have little to no effect on the evolution of the **R** (and **C**) matrices and a higher value (>80%) means that almost all values are pruned. Additional results corresponding to comparison using different values for density are given in Appendix IV.

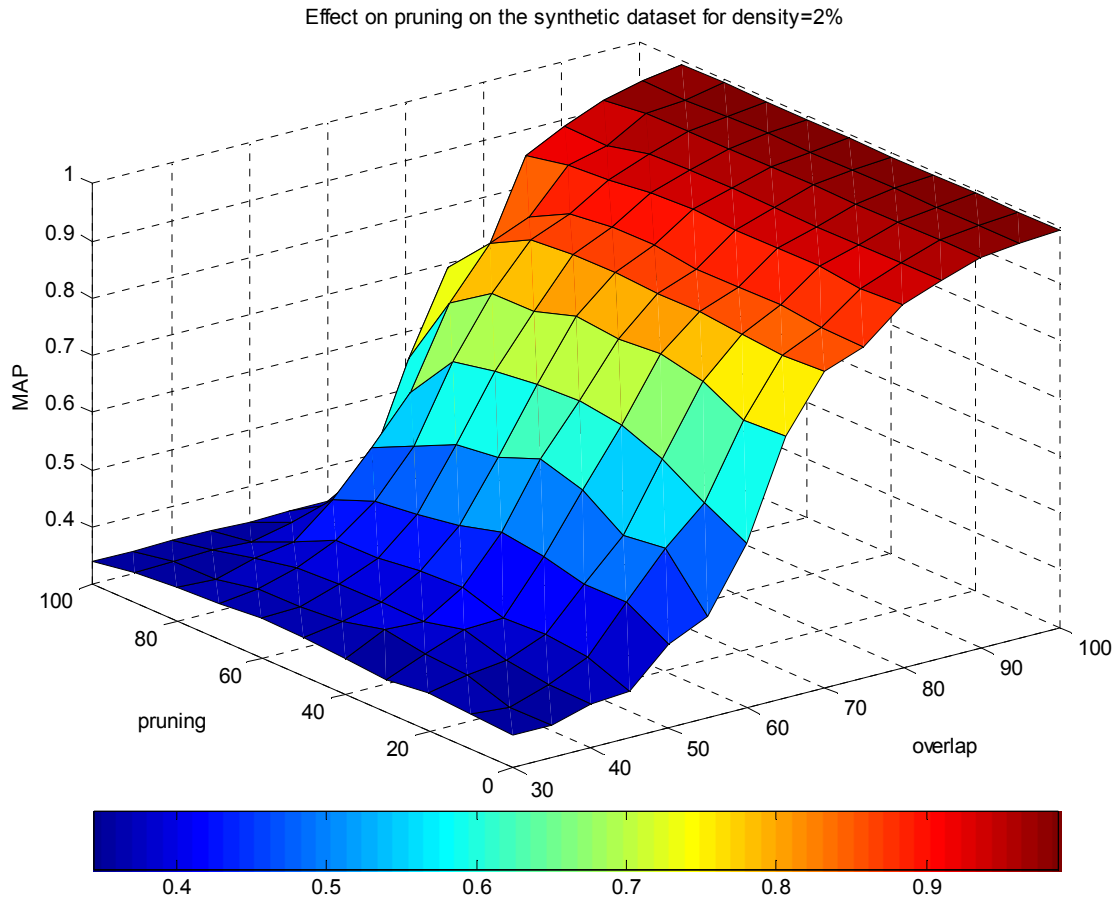


FIGURE 4.3 Effect of pruning on the synthetic dataset

Using Real Dataset

Here we examine the effect of the pruning threshold parameter on the real datasets. We test the χ-Sim algorithm on the various datasets generated from the 20-Newsgroup i.e. M2, M5 and M10 and NG1, NG2 and NG3. As previously, we try different pruning thresholds ranging from 0 to 100 with intervals of 10 on all the datasets and present the results in Table 4-6 and Table 4-7 below.

In order to find empirical evidence that pruning can lead to increase in accuracy of the recovered clusters, we use both datasets – one generated using SMI and the other using PAM that does not need a pre-labeled datasets as discussed in 4.5.1. The supervised feature selection method results in words that have high mutual information with

a given class, and hence the resulting document clusters are more 'pure' in the sense that words usually occur only (or at least much more frequently) in only one document cluster.

Table 4-6 MAP precision values with different levels of pruning using Supervised Measure Information (SMI)

		Pruning Percentage											Max Gain
		0	10	20	30	40	50	60	70	80	90	100	
Datasets	$M2_{SMI}$	0.91	0.94	0.93	0.94	0.93	0.93	0.88	0.68	0.62	0.61	0.51	0.03
	$M5_{SMI}$	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.95	0.91	0.61	0.21	0.00
	$M10_{SMI}$	0.69	0.69	0.69	0.70	0.73	0.72	0.73	0.71	0.66	0.57	0.13	0.04
	$NG1_{SMI}$	0.96	0.96	0.97	0.96	0.96	0.96	0.96	0.96	0.92	0.65	0.51	0.01
	$NG2_{SMI}$	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.91	0.86	0.61	0.21	0.00
	$NG3_{SMI}$	0.79	0.78	0.78	0.79	0.79	0.80	0.81	0.84	0.84	0.67	0.14	0.05

Table 4-7 MAP precision values with different levels of pruning when using Partitioning Around Medoids (PAM)

		Pruning Percentage											Max Gain
		0	10	20	30	40	50	60	70	80	90	100	
Datasets	$M2_{PAM}$	0.58	0.59	0.62	0.62	0.62	0.65	0.65	0.63	0.57	0.55	0.51	0.07
	$M5_{PAM}$	0.62	0.67	0.65	0.68	0.64	0.58	0.56	0.51	0.48	0.39	0.22	0.06
	$M10_{PAM}$	0.43	0.42	0.43	0.45	0.47	0.47	0.47	0.47	0.44	0.34	0.11	0.04
	$NG1_{PAM}$	0.54	0.54	0.54	0.55	0.60	0.61	0.62	0.59	0.60	0.57	0.51	0.07
	$NG2_{PAM}$	0.60	0.63	0.59	0.60	0.62	0.60	0.58	0.58	0.59	0.53	0.50	0.03
	$NG3_{PAM}$	0.47	0.48	0.52	0.57	0.56	0.54	0.51	0.47	0.43	0.35	0.13	0.10

Therefore, the clustering is relatively easier and many of the (redundant) paths, in fact, just grow within documents of the same cluster. The results from different pruning threshold values are given in Table 4-6. As can be observed, the improvement from using the pruning parameter is negligible particularly in datasets with fewer classes, such as NG1 and NG2 and M5. Only the more difficult clustering tasks (relatively speaking) of M10 and NG3 with 10 and 8 document categories show any significant gain from using the pruning threshold, as shown by the "Gain" column

which gives the maximum gain in precision value when using pruning.

In comparison, the results in Table 4-7 correspond to an unsupervised feature selection where a heuristic (PAM) is used for estimating the quality of the features. As can be seen, the results are quite different. As expected using an unsupervised feature selection method results in a generally lower accuracy value for all the datasets. However, with the unsupervised feature selection, the impact of the pruning threshold is now much more contrasted. As seen from the last column in Table 4-7, the gain in precision is relatively higher even for datasets with smaller number of classes. In particular, the NG3 dataset shows a significant increase in precision value when using the pruning threshold on the unsupervised feature selection dataset as opposed to when using mutual information to select features.

This re-enforces our assumption that pruning can be a good way of dealing with 'noise' in the data. While we do not provide an effective way to estimate the best pruning threshold, our experiments show that the pruning parameter generally improves the accuracy measure over a fairly large range of values. Defining a technique to automatically estimate an optimal pruning parameter for a given dataset is an interesting prospect and will be a focus on our future research.

4.5.4. Comparison with Other Methods

In this sub-section, we compare the results of our approach with those of other classical similarity and co-clustering based algorithms. There exists a variety of algorithms that have been proposed for clustering and related disciplines that could also be used for clustering. Our choice of algorithms depended on their relationship with the proposed approach, relevance to the document clustering task, availability of a running implementation, and their performance (see also section 4.7).

As mentioned previously, the Cosine similarity measure has been traditionally used as a similarity measure in text clustering and categorization. It also provides us a baseline algorithm since it is based on the vector space model essentially using only order-1 paths. Both the Latent Semantic Analysis (LSA) method and the Similarity in Non-Orthogonal Space (SNOS) either implicitly or explicitly uses higher order co-occurrences and hence, provide for an interesting comparison. LSA has been used in several text mining applications and has been shown to "capture" the underlying semantics in a text corpus, as mentioned in section 2.5.3. SNOS is similar to χ-Sim but uses a different normalization factor. The comparison with SNOS is particularly interesting since it highlights the importance of taking the document (and word) lengths into account when comparing documents of unequal sizes. Three other co-clustering algorithms which have been proposed for the document clustering task were used for comparisons.

With a collection of our benchmarks datasets, we compared our co-similarity based measure, χ-Sim, with the following methods:

- Cosine: a classical similarity measure (G Salton 1971) that has been typically used for computing document similarities. We consider Cosine as a baseline for comparison since it does not capture higher order similarities2.2).
- LSA: Latent Semantic Analysis as described in (Landauer et al. 2007) (see also 2.4).

- ITCC: Information Theoretic Co-Clustering algorithm (Dhillon et al. 2003) (see also section 2.5.3).
- BVD: Block Value Decomposition (Long et al. 2005) (see also section 2.5.1).
- RSN: *k*-partite graph partitioning algorithm (Long et al. 2006).
- SNOS: Similarity in Non-Orthogonal Space (N. Liu et al. 2004) (see also section 3.6.2)
- SRCC: Similarity Refined based Co-clustering (J. Zhang 2007) (see also section 3.6.1)

Creation of the Clusters

For χ-Sim, Cosine and LSA, in order to create the clusters, we used an Agglomerative Hierarchical Clustering (AHC) method with Ward's linkage on the distance matrices generated by these algorithms respectively. We cut the clustering tree at the level corresponding to the number of document clusters we are waiting for (2 for subset M2, 5 for subset M5, etc).

We implemented our algorithm (χ-Sim) and SNOS in Matlab® to generate the similarity matrices **R** and **C**. Similarity, for the Cosine similarity, SVD (for LSA), and AHC, we used the built in functions available in Matlab®. For ITCC, we used the implementation provided by the authors[22]. We set the number of document clusters to the original documents clusters. For BVD and RSN, as we don't have a running implementation, we quote the best values from (Long et al. 2005) and (Long et al. 2006) respectively.

In order to find the optimal values for the parameters of LSA, ITCC and χ-Sim methods, we proceeded as follows – For LSA, we tested the LSA algorithm iteratively keeping the *s* highest singular values from $s = 10$ to 200 by steps of 10. We report the best value as the one which provides the highest average accuracy amongst all the tested values of *s* for all the randomly generated datasets of that particular dataset. For the ITCC algorithm, we ran the algorithm using different number of word clusters, as suggested in (Dhillon et al. 2003), for each of the datasets. Taking into account that the results of ITCC are based on random initialization, we repeated these experiments 3 times in each case and report the best results.

Comparison using a Knowledge Intensive Approach

As mentioned in previously (section 4.5.1), we tested our algorithm on both datasets generated by selecting features based on their Supervised Mutual Information (SMI) with the document category as well as datasets generated using purely unsupervised approaches. First, we provide the results using SMI based feature selection. We provide both the micro-averaged precision value (Pr) as well as the NMI score to facilitate the comparison of results (Long et al. 2005) and (Long et al. 2006) respectively.

The experimental results (see Table 4-8 and Table 4-9) show that χ-Sim significantly outperforms the other tested methods on both Pr and NMI scores over all the datasets. The subscript "ρ" denotes that the results correspond to the best pruning parameter (see section 3.5.5). The column named "Runtime" reports the average runtime in seconds needed by χ-Sim to compute the two matrices, **R** and **C**, with 4 iterations.

[22] Available at http://userweb.cs.utexas.edu/users/dml/Software/cocluster.html

Table 4-8 Comparison of MAP and NMI scores on the 20-Newsgroup datasets with fixed number of total documents and the CLASSIC3 dataset. With increasing number of clusters, the documents per cluster decrease**.**

	Time		χ-Sim$_p$	Cosine	LSA	ITCC	SNOS	BVD
M2$_{SMI}$	*12.01s*	Best Pr	**0.96**	0.72	0.94	0.92	-	0.95
		Pr$_{(avg)}$	**0.93 ± 0.01**	0.62 ± 0.04	0.78 ± 0.15	0.71 ± 0.15	0.55± 0.02	-
		NMI$_{(avg)}$	**0.65 ± 0.05**	0.15 ± 0.05	0.37 ± 0.23	0.21 ± 0.21	-	-
M5$_{SMI}$	*10.93s*	Best Pr	**0.98**	0.76	0.95	0.56	-	0.93
		Pr$_{(avg)}$	**0.94 ± 0.04**	0.6 ± 0.08	0.82 ± 0.09	0.48 ± 0.06	0.25± 0.02	-
		NMI$_{(avg)}$	**0.85 ± 0.07**	0.55 ± 0.11	0.65 ± 0.14	0.27 ± 0.09	-	-
M10$_{SMI}$	*13.53s*	Best Pr	**0.80**	0.55	0.67	0.33	-	0.67
		Pr$_{(avg)}$	**0.75 ± 0.04**	0.45 ± 0.07	0.54 ± 0.09	0.28 ± 0.04	0.24± 0.06	-
		NMI$_{(avg)}$	**0.65 + 0.05**	0.43 ± 0.07	0.45 ± 0.08	0.16 ± 0.04	-	-
Classic3$_{SMI}$	*164.38s*	Pr	**0.99**	0.89	0.98	0.97	0.95	-
		NMI	**0.96**	0.63	0.93	0.90	-	-

Table 4-9 Comparison of MAP and NMI scores on the 20-Newsgroup datasets with fixed number of total documents and the CLASSIC3 dataset. With increasing number of clusters, the documents per cluster decrease.

	Time		χ-Sim$_p$	Cosine	LSA	ITCC	SNOS	RSN	SRCC
NG1$_{SMI}$	*6.60s*	Best Pr	**1**	0.97	0.98	0.83	-	-	
		Pr$_{(avg)}$	**1 ± 0**	0.90 ± 0.11	0.95 ± 0.02	0.67 ± 0.12	0.51± 0.01	-	
		NMI$_{(avg)}$	**1 ± 0**	0.62 ± 0.20	0.75 ± 0.08	0.13 ± 0.13	-	0.64 ± 0.16	0.90 ± 0.04
NG2$_{SMI}$	*14.56s*	Best Pr	**0.94**	0.72	0.87	0.72	-	-	
		Pr$_{(avg)}$	**0.93 ± 0.01**	0.60 ± 0.08	0.82 ± 0.03	0.57 ± 0.08	0.24± 0.02	-	
		NMI$_{(avg)}$	**0.81 ± 0.04**	0.51 ± 0.07	0.64 ± 0.03	0.35 ± 0.09	-	0.75 ± 0.07	0.81 ± 0.02
NG3$_{SMI}$	*20.20s*	Best Pr	**0.91**	0.62	0.80	0.65	-	-	
		Pr$_{(avg)}$	**0.85 ± 0.06**	0.60 ± 0.04	0.72 ± 0.05	0.55 ± 0.06	0.22± 0.05	-	
		NMI$_{(avg)}$	**0.77 + 0.02**	0.55 ± 0.03	0.59 ± 0.03	0.48 ± 0.05	-	0.70 ± 0.04	0.75 ± 0.03

It is interesting to observe the behavior of χ-Sim on various datasets (M2, M5, M10) when we keep the total number of documents constant while decreasing the number of documents per cluster and increasing the cluster numbers. As the number of clusters increases, the inherent structure of the dataset gets more complicated and naturally one would expect a decrease of the accuracy of all the algorithms. However, we note that the performance of our proposed co-similarity based measure decreases much slowly than other techniques like LSA, ITCC or BVD. Even when the number of documents per cluster is just 50 (in the case of M10), χ-Sim is able to retrieve the clusters more accurately obtaining a micro-averaged precision of up to 0.798 which is better than the best results with LSA and BVD at 0.668 and 0.67 respectively.

It is also interesting to note from the experiments that χ-Sim seems more consistent on both the Pr and NMI scores across the 10 randomly generated sets of each dataset, showing a maximum standard deviation of 0.07. In particular, it is more consistent across datasets with smaller numbers of clusters (M2, M5, NG1 and NG2), where we observe a greater variation in the results of Cosine, LSA and ITCC.

Of particular interest in the comparison with the SNOS algorithm which (as mentioned in section 3.6.2) is significantly similar to our algorithm. SNOS have performed relatively poorly on all the datasets, even less than the Cosine measure. This shows the importance of the normalization factor in the algorithm. As discussed in the previous chapter (section 3.3.5), it is important to normalize by the size of the document and word vectors so as to take into account the different sizes of the documents and words.

It should be noted here that the results using BVD, RSN and SRCC are quoted from their papers as provided by the authors, since we did not have a running implementation of either. Therefore, the results are not computed on the same datasets are the rest of the algorithms. Secondly, while the authors mention that they generated their datasets "by duplicating the datasets as in (Dhillon et al. 2003)", our communication with the authors (of BVD) in an effort to get their datasets or implementation resulted in an ambiguity since they suggested having used the Rainbow package which provides a feature selection criteria using the supervised mutual information. Therefore, we quote their results in this section. A proper comparison, however, can only be done by running those algorithms on the same datasets that we have used (or vice versa). Therefore, care must be taken when comparing these results.

Table 4-10 Results of significance test of χ-Sim versus other algorithms on the various datasets.

	χ-Sim					
	$M2_{SMI}$	$M5_{SMI}$	$M10_{SMI}$	$NG1_{SMI}$	$NG2_{SMI}$	$NG3_{SMI}$
Cosine	<<	<<	<<	<<	<<	<<
LSA	<	<<	<<	<	<<	<<
SNOS	<<	<<	<<	<<	<<	<<
ITCC	<<	<<	<<	<<	<<	<<

Table 4-10 shows the result of these significance tests between χ-Sim and the other tested algorithms over the 10 folds that were generated for each dataset. The symbol "<<" is used to denote a *p*-value of less than 1e-10 and

"<" is used to denote when the p-value is between 0.05 and 1e-10. It can be seen that the p-values are almost always very low, thus strongly rejecting the hypothesis that the two models (χ-Sim vs. any other) are drawn from the same distribution. Thus, the results shown generated by our proposed method are all statistically significant.

Comparison using Unsupervised Approach

Here, we use the same methodology as before except that the feature selection method is now done using totally unsupervised methods. We first test on both the UMI criteria and using PAM based feature selection. The results are summarized in Table 4-11 for the unsupervised mutual information and in Table 4-12 using PAM. The values indicate the micro-averaged precision (MAP) score.

Table 4-11 Comparison of MAP score on various NG20 datasets using unsupervised mutual information based feature selection

	χ-Sim$_\rho$	Cosine	LSA	SNOS	ITCC
$M2_{UMI}$	**0.61 ± 0.01**	0.58 ± 0.01	**0.61** ± 0.04	0.51 ± 0.02	0.58 ± 0.02
$M5_{UMI}$	**0.50 ± 0.04**	0.36 ± 0.05	0.45 ± 0.04	0.22 ± 0.03	0.32 ± 0.02
$M10_{UMI}$	**0.37 ± 0.04**	0.27 ± 0.03	0.34 ± 0.03	0.16 ± 0.03	0.22 ± 0.01
$NG1_{UMI}$	**0.63 ± 0.01**	0.58 ± 0.05	**0.63 ± 0.08**	0.50 ± 0.00	0.60 ± 0.05
$NG2_{UMI}$	**0.37 ± 0.04**	0.29 ± 0.04	0.33 ± 0.03	0.23 ± 0.05	0.28 ± 0.01
$NG3_{UMI}$	**0.24 ± 0.03**	0.18 ± 0.01	0.23 ± 0.02	0.14 ± 0.01	0.16 ± 0.03

Table 4-12 Comparison of MAP score on various NG20 datasets using PAM based feature selection

	χ-Sim$_\rho$	Cosine	LSA	SNOS	ITCC
$M2_{PAM}$	0.65 ± 0.09	0.61 ± 0.04	**0.79 ± 0.09**	0.51 ± 0.02	0.58 ± 0.05
$M5_{PAM}$	**0.68 ± 0.06**	0.54 ± 0.08	0.66 ± 0.05	0.26 ± 0.04	0.54 ± 0.05
$M10_{PAM}$	**0.47 ± 0.04**	0.39 ± 0.03	0.44 ± 0.04	0.20 ± 0.02	0.29 ± 0.05
$NG1_{PAM}$	**0.62 ± 0.01**	0.52 ± 0.01	0.56 ± 0.05	0.51 ± 0.00	0.61 ± 0.06
$NG2_{PAM}$	**0.63 ± 0.04**	0.60 ± 0.05	0.61 ± 0.06	0.24 ± 0.01	0.44 ± 0.08
$NG3_{PAM}$	**0.57 ± 0.04**	0.49 ± 0.02	0.52 ± 0.03	0.22 ± 0.02	0.49 ± 0.07

As can be seen, the results are very different. Since the datasets are not generated using class discriminatory words, the precision values of all the algorithms are lower. Overall, χ-Sim still seems to perform best on average across the different datasets. However, the relative difference in results between χ-Sim and the other algorithms is now lower. This is particularly true for LSA, which also (implicitly) uses higher-order co-occurrences. As before,

SNOS performs poorly across the different datasets in both cases, while Cosine and ITCC performs comparably in the case of UMI while Cosines tends to be better in the case of PAM.

One may try to explain these results by considering that when the classes are well separated (as in the case of using SMI), higher order co-occurrences generated using χ-Sim provide a richer information since these words are highly discriminative and tend to occur mostly in the same class. However, when using an unsupervised approach, the words are less discriminating and many cross class boundaries. This results in the propagation of similarity values between documents (and words) belonging to different classes as discussed in section 3.5.5. This explanation is also consistent with our previous observation in 4.5.3, where we saw the effect of the pruning parameter is more significant in the case of unsupervised feature selection. Nonetheless, the higher order co-occurrences brings some new information that helps in better judging the similarity values as in evident since both χ-Sim and LSA outperforms the other methods. As stated earlier, SNOS suffers from the fact that it does not take document (and word) length into account which we believe is responsible for its poor performance.

4.6. Text Categorization

This section provides another empirical study to show the benefits of our proposed measure for text categorization. We evaluate the proposed method with several real text corpora. Specifically, we perform the task of text categorization which involves a training dataset whose document category labels are provided to the algorithm and use the learned word similarities to assign each document of the test dataset into one of the document categories. We will start this section by giving a brief overview of some existing methods against which we tested the performance of our algorithm.

4.6.1. Related Work

Several clustering algorithms have been extended to the supervised classification case. For example (N. Slonim and N. Tishby 2001) expands the information bottleneck approach and (Takamura and Matsumoto 2002) the maximum likelihood criteria for the text categorization task. We focus here on approaches that use the inherent document and/or word semantic relationships that might exist within the data matrix and use category labels from the training dataset to exploit these relationships, as it provides a direct comparison with our proposed method.

Higher Order Co-occurrences using Case Retrieval Networks

Closer to our work, (Chakraborti, Wiratunga, et al. 2007) have proposed using higher order co-occurrences as a measure of word similarities. They propose an algorithm to calculate paths of order-1, 2, and 3 between words which they store in different similarity matrices, say **C1**, **C2** and **C3**.

The first matrix, **C1** is calculated by multiplying the transpose of the original data matrix, $\mathbf{A}^T$, with **A** to yield first order co-occurrence values. However, instead of normalizing the similarity values in **C1** based on the lengths of the word vectors as done in our approach, every non-zero entry in **C1** is replaced by 1. Thus, the binary first order

co-occurrence matrix only shows whether or not two words co-occur in any document but neither takes into account their frequency of co-occurrence nor their vector length.

The second order similarity matrix, **C2**, is generated by multiply **C1** with itself after setting the diagonal elements to zero. The diagonal elements of **C1** correspond to an order-1 walk between a word and itself. Since **C2** constitutes walks of order-2 within a bi-partite graph, by setting the diagonal of **C1** we can be sure to avoid any redundant walk of order-1 and **C2** consequently contain only elementary walks of order-2.

The elements of **C3** are subsequently constructed by taking the cube of **C1** i.e. (**C3**) = (**C1**)^3. However, this will result in some redundant walks of order-3 that corresponds to sub-walks of a lower order (order-1 and order-2). (Chakraborti, Wiratunga, et al. 2007) provide a technique to rewrite the procedure as a matrix manipulation that can quantify the number of such non-elementary (self avoiding) walks. By building a 'discount matrix' **D**, whose elements D_{ij} correspond to ($|\mathbf{a}^i| + |\mathbf{a}^j| - 1$), we can quantify the number of non-elementary paths of order-3 by multiplying **D** by **C1** (for proof, see Chakraborti, Wiratunga, et al. 2007). Thus the final matrix **C3** is computed by

$$\mathbf{C3} = (\mathbf{C1})^3 - \mathbf{D} * \mathbf{C1} \tag{4.5}$$

A genetic algorithm is then used to search for (locally) optimal weighting mechanism for combining these matrices i.e. $\mathbf{C} = \alpha\mathbf{C1} + \beta\mathbf{C2} + \gamma\mathbf{C3}$, where **C** is the resulting similarity matrix and α, β and γ are weighting parameters which are to be optimized. Class knowledge is exploited by adding class discriminating words (sprinkling) into the original data matrix as described in 3.7.2 of chapter 3, and a CRN (Case Retrieval Network) is used to assign class labels to documents in the text set using a *k*-NN approach.

This approach can be compared with the higher order non-redundant path technique discussed in section 3.5. Unlike the χ-Sim algorithm, however, their approach does not exploit the dual nature of the relationship between documents and words i.e. words are similar irrespective of the relationship between documents that generate the higher order paths. Furthermore, the algorithm requires maintaining different similarity matrices of different order-paths although the overall complexity remains comparable to χ-Sim.

Supervised Latent Semantic Indexing

Several works has been done to incorporated prior knowledge into the SVD based LSI approach which has also been shown to implicitly exploit higher order relationships (Kontostathis and Pottenger 2006). The sprinkling approach was proposed by (Chakraborti et al. 2006) as a measure of enriching the reduced lower-dimensional 'concept space'. As was mentioned previously (section 3.7.2) where we used sprinkling to enhance within class similarity in χ-Sim, sprinkling amounts to a set of artificial words, corresponding to the class labels of the documents, being appended to the training document-term matrix. SVD is then performed on this augmented matrix and a lower rank approximation is obtained. This approximate matrix has the same dimensions as the augmented matrix. The augmented words are then dropped and categorization of test documents is performed using *k*-NN. The framework of the sprinkling method is shown in FIGURE 4.4.

The main idea here is that, in the absence of overlapping words between documents of different classes, the top *k* singular values after performing a SVD operation would correspond to the class structure in the original document-

term matrix (see 2.5.1.). Thus, by augmenting class specific words, we wish to emphasize the implicit word relationships with a given class by forcing co-occurrences between words that mostly appear in that document class. This results in promoting the *k* singular values that represent the class structure in the original document-term matrix. This is shown in FIGURE 4.5.

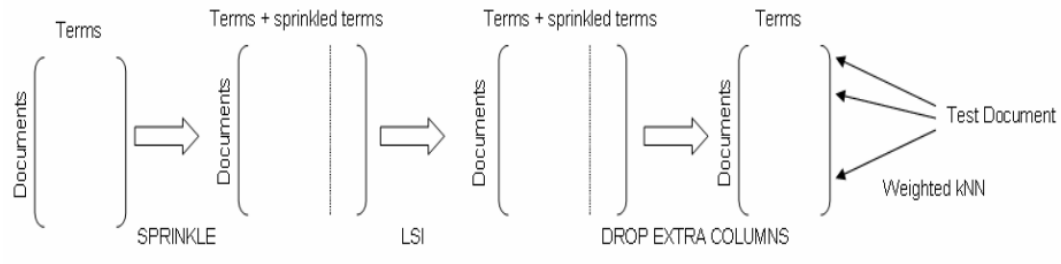


FIGURE 4.4 Classification using sprinkled LSI **(Chakraborti et al. 2006)**

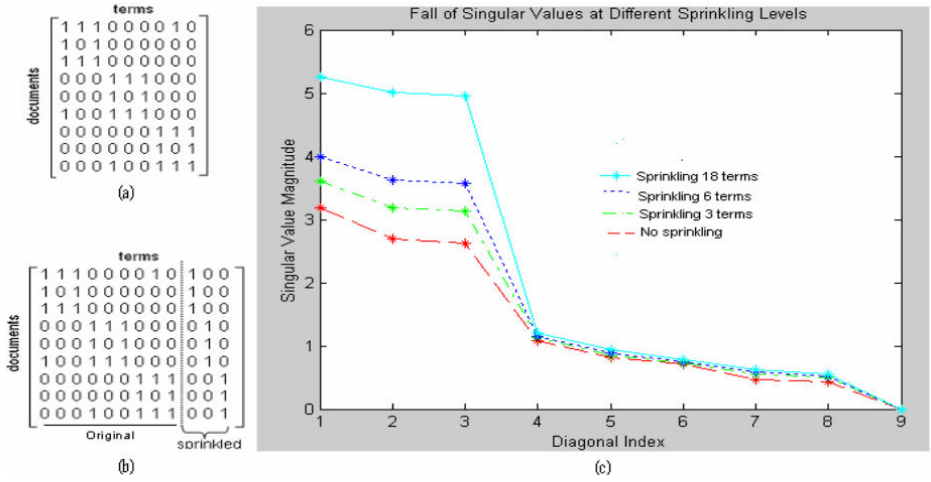


FIGURE 4.5 (a) the original document by term matrix, (b) the augmented matrix by 3 words, and (c) the effect of sprinkling on Singular values **(Chakraborti et al. 2006)**

A subsequent development (Chakraborti, Mukras, et al. 2007) of the approach was proposed by the same authors as a way to select different number of words for different classes. The idea behind this improvement is that different document classes might have different levels of their ease of extraction, and hence need different number of sprinkled words. The approach, called Adaptive Sprinkling (AS), first defines the maximum number of sprinkled columns, and then distributes these columns among the different document classes. The assignment of sprinkled column to each class is based on the observed confusion matrix on a validation dataset.

The algorithms do not propose a way to determine the number of columns to be added to the original matrix. This limitation also extends to our approach of weighting the augmented column. In practice, a cross-validation strategy can be used to estimate the number of artificial words (or in our case the weight of the artificial word),

though this might not be the optimal value. Unlike our approach, however, both the above approaches are still dependent on the sensitive nature of LSI to the number of lower-dimensions to which the matrix is projected. Furthermore, adding more columns to the original document-term matrix no longer guarantees the *k*-rank approximation on which the LSI method is based on.

Other Algorithms

We also tested our method against two other approaches – the first one is the well known *k*-nearest neighbors (*k*-NN) method which has been studied very well in pattern recognition and which has been applied to text categorization (Yiming Yang 1994). The *k*-NN method uses a simple technique of comparing a given test document with all training documents and chooses the *k* nearest neighbors based on similarity score. The category of the test document is chosen based on weighted score of its *k* nearest neighbors. We used the popular Cosine similarity measure with the *k*-NN approach. The second method we test against is the Support Vector Machine (SVM) introduced in the 1990s (Boser, Guyon, and Vapnik 1992). SVM is based on the concept of determining a decision surface that 'best' separates data points in two different classes. Both *k*-NN and SVM are well known techniques and many description of these methods can be found, such as in (Y. Yang and X. Liu 1999) and (Joachims 1998), therefore, we omit the details of these methods.

A linear kernel was used for SVM. This was chosen for two main reasons — a running implementation is readily available for multi-class classification[23], and it has been found to perform well in text categorization (Joachims 1998). As was the case with the unsupervised methods (section 4.5.4), the methods presented here for comparison are only a handful of a vast existing literature in the domain. Several other methods exists for text categorization including, for instance, many new kernels and techniques based on learning a similarity (or distance) metric, etc. that could also be used for a more rigorous comparison (see also section 4.7).

4.6.2. Datasets

We evaluate the proposed techniques on several real datasets. We used 3 popular text corpora which have been widely used in the literature for both classification and clustering tasks and created several datasets. The first corpus, the 20-Newsgroup (section 4.3.1) from which we created the following datasets:

- **HIERARCHICAL:** This is a dataset containing the *comp* and *rec* sub-trees with 5 and 4 classes respectively from the 20-Newsgroup corpus. This dataset was used in *(Chakraborti, Mukras, et al. 2007)*.
- **RELPOL**: The RELPOL dataset corresponds to two classes, Religion and Politics, from the Newsgroup dataset. This dataset was used in *(Chakraborti et al. 2006)*, *(Chakraborti, Wiratunga, et al. 2007)*. Each of the class consists of 1000 messages from that topic.

[23] http://svmlight.joachims.org/svm_multiclass.html

- **HARDWARE**: The HARDWARE dataset corresponds to two classes relating to hardware, Apple Mac and PC, from the 20-Newsgroup dataset. This dataset was used in *(Chakraborti et al. 2006)*, *(Chakraborti, Wiratunga, et al. 2007)*.

We also use two other text corpora the Reuters-21578 which is a collection of documents that appeared on Reuters (section 4.3.2) newswire in 1987, and the LINGSPAM (section 4.3.4) corpus which is a collection of legitimate and spam emails, typically used for the task of filtering spam emails. We formed the following datasets.

- **ORTHOGONAL:** This dataset consist of 3 classes (acq, crude and earn) from the Rueters-21578 collection. We selected 500 documents from each class such that each document belongs to only one category. This dataset was used in *(Chakraborti, Mukras, et al. 2007)*.
- **LINGSPAM**: This dataset contains 2893 email messages of which 83% are non-spam messages related to linguistics and the rest are spam. This dataset was also used in *(Chakraborti, Mukras, et al. 2007)*.

All datasets underwent similar pre-processing. After stop-words removal and stemming, we convert the resulting Document term matrix to binary values as in (Chakraborti et al. 2006), (Chakraborti, Mukras, et al. 2007), (Chakraborti, Wiratunga, et al. 2007). For the HIERARCHICAL and ORTHOGOCAL datasets, 500 documents were randomly selected from the dataset. For the other datasets, 20% documents were randomly selected. Equal sized training and test datasets were formed in all cases respecting the category distribution in the dataset. We selected the top 1000 words based on (Supervised) Mutual Information (SMI) in the training set and keep the same features for the test set. For repeated trials, 10 such train-test pair sets were randomly generated from each dataset and the average and standard deviation reported. We use MAP as a measure of effectiveness as this has been preferred in single labeled datasets of equal sizes (Gabrilovich and Markovitch 2005).

4.6.3. Methods

As seen in the previous chapter, we study two methods to classify the documents in the test dataset – using the popular *k*-NN method that associates a test document based on its *k* nearest neighbors and using Nearest Class (NC) approach which forms a class centroid for each class in the training dataset and assigns each document of the test dataset with the nearest class centroid (see details in section 3.7.5). We test both these approaches in our experimentation. Furthermore, we tested each variant of the χ-Sim algorithm (using *k*-NN or NC) with different values of the two parameters *w* and λ, introduced in section 3.7. A value of w=2 was empirically found to be a good compromise between under-influencing and over-emphasizing category information and hence we used a value of 2 for w in all the experiments. The value of *w* can also be determined by creating a validation set from the training dataset and using, say, 5-fold cross-validation approach to determine the value of *w*. We report experimental results for two values of λ — at λ=1 which corresponds to the unsupervised approach; and at λ=0 which discards similarity value between documents of different categories and was empirically found to give the best results.

We performed a comparison of our proposed approach with several other approaches cited in section 4.6.1, namely – the higher order word similarity based case retrieval network (HOCRN) (Chakraborti, Wiratunga, et al.

2007), Supervised LSI using sprinkling (SLSI) (Chakraborti et al. 2006), adaptive sprinkling based LSI (ASLSI) (Chakraborti, Mukras, et al. 2007), and Support Vector Machines (SVM) using a linear kernel (Joachims 1998). In addition to these algorithms, we implemented a weighted *k*-NN using the Cosine similarity as a baseline method for comparison. The results from HOCRN, SLSI, and ASLSI are quoted from (Chakraborti, Wiratunga, et al. 2007), (Chakraborti et al. 2006) and (Chakraborti, Mukras, et al. 2007) respectively since we conduct experiments on the same datasets and use similar pre-processing measures.

Usually, a nested cross-validation criterion is usually employed when evaluating supervised learning algorithms. However, to make our work comparable with the other algorithms mentioned above, we generated random training and test datasets. We adjusted the various parameters and report the best results. For example, when using *k*-NN to classify the labels, *k* was set to 3 for both χ-Sim and *k*-NN as this was empirically found to be a good value. Similarly, we adjusted the parameter for SVM and report the results corresponding to the best precision value. We found a fairly large plateau for the parameter 'c' of SVM and usually obtained good results for c=500. Moreover, we tested χ-Sim, K-NN with Cosine, and SVM using *number of occurrences*, *tf-idf* and Boolean indicators (section 2.1) for the document-term matrix, and report results corresponding to the *number of occurrences* as this was found to give the best result for each algorithm.

All results for χ-Sim, *k*-NN using Cosine, and SVM are averaged over 10 runs on each dataset each time randomly selecting documents to form the training and test sets.

4.6.4. Analysis

The results of the various datasets are shown in Table 4-13. We report results corresponding to 3 cases:

- Using sprinkled columns only (w=2, λ=1),
- Setting inter-category similarity values to 0 only (w=0, λ=0)
- Using both sprinkled columns and setting inter-similarity values to 0 (w=2, λ=0).

Since we used the (SMI) based feature selection, we found the effect of the pruning parameter parameter to be almost irrelevant, which is consistent with our previous observations in clustering (section 4.5.3). As such, the results reported in this section do not use pruning (i.e. ρ=0).

From in Table 4-13, we observe that setting the value of λ to 0 always increases the accuracy of the result. This improvement is quite significant especially in the case of the HIERARCHICAL dataset (using *k*-NN) which is much harder since several classes belong to the same sub-tree which makes them rather fuzzy. For example, the comp sub-tree contains the classes – *sys.ibm.pc.hardware*, *sys.mac.hardware*, *comp.windows.xp* which can contain many similar words hence making the boundaries rather difficult to detect. As such, many words that are not very discriminatory could end up having a high similarity value. Setting λ to 0 forces only word co-occurrences within a category to contribute to the similarity values. Intuitively, only words that are representative of a class will have such higher order co-occurrences with a category.

We further investigate this by grouping the sub-trees together as one category, thus ending up with only 2 categories corresponding to the two sub-trees. We re-run the χ-Sim algorithm on this data and report the findings in

Table 4-14. As expected, the impact on the accuracy values by setting λ to 0 or 1 is comparatively less significant in the transformed hierarchical dataset since the fuzzy sub-categories are now considered as the same.

Next we analyze the different classification techniques. Forming category vectors, NC is observed to give comparable or better results than traditional *k*-NN. For example, when λ is set to zero, NC can be seen to be comparable or outperform *k*-NN on all but the ORTHOGONAL dataset where the accuracy is slightly less. Moreover, using NC lead to less variations in the results between setting λ=0 and λ=1. One reason for this is that NC generalizes the category "concept" and is not susceptible to data sparseness as is the case in *k*-NN.

Table 4-13 Precision values with standard deviation on the various dataset

	HIERARCHICAL	HARDWARE	RELPOL	ORTHOGONAL	LINGSPAM
χ-Sim (w=2,λ = 1) + *k*-NN	0.6564 ± 0.01	0.8540 ± 0.01	0.9816 ± 0.00	0.9511 ± 0.00	0.8816 ± 0.06
χ-Sim (w=2,λ = 1) + NC	0.7666 ± 0.01	0.8617 ± 0.02	0.9783 ± 0.00	0.9318 ± 0.00	0.8778 ± 0.07
χ-Sim (w=2,λ = 0) + *k*-NN	0.7349 ± 0.01	0.8584 ± 0.01	**0.9899 ± 0.00**	0.9558 ± 0.00	0.9298 ± 0.06
χ-Sim (w=2,λ = 0) + NC	**0.7699 ± 0.01**	0.8713 ± 0.01	0.9848 ± 0.01	0.9378 ± 0.01	0.9803 ± 0.01
χ-Sim (w=0,λ = 0) + *k*-NN	0.4934 ± 0.03	0.7520 ± 0.02	0.9634 ± 0.01	0.9425 ± 0.01	0.9608 ± 0.02
χ-Sim (w=0,λ = 0) + NC	0.7556 ± 0.01	0.8698 ± 0.01	0.9752 ± 0.01	0.9459 ± 0.01	0.9559 ± 0.03
k-NN + Cosine	0.6534 ± 0.01	0.8422 ± 0.01	0.9805 ± 0.01	0.9507 ± 0.01	0.9726 ± 0.01
SLSI	-	0.8042	0.9389	-	**0.9832**
ASLSI	0.6040	-	-	0.9520	-
HOCRN	-	0.8044	0.9393	-	-
SVM	0.7692 ± 0.01	**0.8740 ± 0.07**	0.9671 ± 0.01	**0.9650 ± 0.01**	0.9513 ± 0.02

When compared to other algorithms, χ-Sim is seen to match or outperform other algorithms especially on the HIERARCHICAL and HARDWARE datasets (except for SVM) which can be seen as harder problems. However, on two of the datasets – LINGSPAM and RELPOL, χ-Sim outperforms SVM. In general, however, we see that the results are fairly comparable (except for the HIERARCHICAL dataset) for most of the algorithms, particularly in the case of SVM. One explanation for this result is that using supervised mutual information for feature selection results words that are both discriminating and not rare. This seems all the more credible when we compare the results from RELPOL and LINSPAM datasets where a simple Cosine based *k*-NN algorithm gives very good results. The Cosine measure only compares the existence of common words between documents which suggests the existence of discriminating groups of words in the datasets. In such cases, using higher-order co-occurrences might not necessarily improve an already (relatively) easy problem.

Table 4-15 reports the time to perform the classification task using the NC and *k*-NN algorithms as described in section 3.7.5. The running time of both variants corresponds to the average over all runs of that particular dataset. The algorithms were run using Matlab® on a PC using a 2.4GHz Core 2 Duo CPU and 4GB of memory running under Windows Vista™.

The advantage of using NC is, of course, the gain in performance as every test document is only compared to the category vectors instead of all training documents as in the case of *k*-NN. It can be observed that the NC variant

for categorizing documents in the test set is significantly faster than a traditional *k*-NN approach. The HIERARCHICAL dataset, for example contains 2250 each of training and test documents. The test time, which includes the time to create the prototype vectors for each class, is roughly 6 times less than when using a *k*-NN approach since there are only 9 categories to compare against. The gain in time can be expected to be much more significant when categorizing even larger sets.

Table 4-14 Result of merging the sub-trees of the hierarchical dataset

	Before Merging		After Merging	
	$\lambda = 1$	$\lambda = 0$	$\lambda = 1$	$\lambda = 0$
χ-Sim + *k*-NN	65.64 ± 0.01	73.49 ± 0.01	93.48 ± 0.03	95.98 ± 0.01
χ-Sim + NC	76.66 ± 0.01	76.99 ± 0.01	94.10 ± 0.01	95.71 ± 0.01

Table 4-15 A comparison of running time (in seconds) for on the different datasets.

Datasets	Documents	Train (*in secs*)	Test (*in secs*)	
			χ-Sim + *k*-NN	χ-Sim + NC
HIERARCHICAL	4500	2.14	0.61	0.11
HARDWARE	800	2.61	0.52	0.18
RELPOL	800	1.28	0.29	0.10
ORTHOGONAL	1500	0.92	0.19	0.09
LINGSPAM	570	0.85	0.15	0.11

4.7. Conclusion

In this chapter, we provide experimental results to explore the effectiveness of our proposed co-similarity based measure. We tested various parameters and variation of our proposed techniques on several synthetic and real datasets that have been widely used in the literature.

We first show that using direct co-occurrences (which corresponds to the first iteration in χ-Sim) does not yield accurate results mainly due to sparseness of the data and the fact that documents belonging to a given category do not necessarily share the same set of features. Our proposed measure takes into account the relationships between different words and documents by using weighted higher-order co-occurrence paths in a bipartite graph to obtain the similarity between documents. At the same time, our approach avoids the data-sparseness problem by performing implicit word clustering on the data set by bringing words that belong to the same cluster closer together. This is done by exploiting the dual relationship between words and documents where documents are considered similar when they share similar words and words are thought to be similar when they occur in similar documents.

Through experiments, we have also shown that using the pruning parameter in χ-Sim helps to avoid the adverse

effect of less discriminatory features in a dataset. This is usually the case when unsupervised feature selection methods are used to select a subset of features. Our experiments show that pruning can significantly enhance the accuracy of the resulting clustering in such cases. Although tuning the pruning parameter to obtain the best result was beyond the scope of our study and was not addressed in this thesis, we provide empirical evidence on both synthetic and real datasets to suggest that the pruning parameter displays a fairly large plateau (on the datasets tested) and even arbitrarily choosing the parameter within a certain range could lead to better performance as compared to the baseline approach. Of course this leaves a need for a more formal study of the effects of using higher order co-occurrences and their effects on similarity propagation. Moreover, the empirical evidence need also be improved by testing more datasets.

We also examine the extension of the χ-Sim algorithm for the supervised document categorization task. The inter-class similarity parameter, λ, can result in incorporating category knowledge into the **R** and **C** matrices by influencing their corresponding similarities. A value of $\lambda=0$ results in much better categorization task since by eliminating the similarity between documents that belong to different categories, only words that mainly occur in the same category can have any significant similarity value. Furthermore, we provide empirical evidence that by using a prototype vector for each category to find the Nearest category (NC) whereby test documents are compared by their word similarities to different category vectors instead of the *k* nearest neighbors, we can significantly reduce computations time for the test dataset while maintaining (or even improving) the accuracy of the categorization task.

We would like to stress here that, in general, evaluating an algorithm (even empirically) is not an easy task and one needs to use a variety of datasets and compare with different algorithms on varying tasks. Several different approaches to both clustering and categorization have been proposed in the literature. For instance, kernel and spectral based algorithms have been an area of much research in text clustering in the last decade and several algorithms proposed in this area such as (Dhillon, Y. Guan, and Kulis 2005; Filippone et al. 2008b) including those based on semantic kernels for text clustering(Farahat and Kamel 2009; AlSumait and Domeniconi 2007). Recently, (Yen et al. 2009) proposed an extension to the commute time kernel that takes both direct links (shared words) and indirect links (higher-order co-occurrences) into account and uses several kernel clustering algorithms to perform document clustering. It would be interesting to study χ-Sim as a possible kernel and perform a comparison with other kernel based methods. Similarly, several metric learning based techniques have been developed for document classification such as (Davis et al. 2007; Weinberger and Saul 2009; Y. Chen et al. 2009; Qamar and Gaussier 2009), among many others. As we saw in section 3.3.4, χ-Sim can be considered as a generalized similarity measure of several classical similarity measures. Thus, supervised χ-Sim can be considered as a metric learning approach and forms an interesting case study. We believe these studies can bring significant enhancements into the understanding of χ-Sim and should be an object of focus in future research.

In this chapter, we primarily studied one way clustering i.e. document clustering, and saw that using our co-similarity approach resulted in improving the accuracy values even when feature (word) clustering wasn't explicitly required. In the next chapter, we will present and evaluate our co-similarity approach for the co-clustering task on several biological datasets, where not only do we need to find subsets of samples that have a strong association with subsets of features, but also such co-clusters need to be automatically recovered in the dataset.

Chapter 5
Application to Bioinformatics

In the last chapter, we discussed the problem of document clustering, which is a one-way clustering problem and showed that our algorithm increased the accuracy of the results for various datasets. This chapter aims to evaluate the performance of the algorithm for the two way clustering problem. Gene expression dataset is a typical example where users are interested in clustering both the genes and the conditions under which the genes are examined. We start with a brief introduction of background of gene expression data as well as the processes involved in microarray data analysis. A technique is then presented to automatically extract biclusters from a given dataset after running the χ-Sim algorithm. Resistance to noise is an important factor when considering biclustering techniques for gene expression analysis, since most real world datasets contains noisy data. We evaluate our results on synthetic datasets and show that the algorithm performs satisfactory under different levels of noise. Finally, we perform biclustering on various real world gene expression datasets and evaluate our results based on their gene ontology enrichment.

5.1. Introduction

Two complementary advances over the last 15-20 years, one in the domain of knowledge engineering and management and the other in technology, have greatly stimulated the acquisition of the genome map of living organisms and the study of gene expression data and the analysis of the roles played by specific genes in the metabolism and development of diseases. This leads to the emergence of a new research domain known as *Bio-Informatics,* which is "the combination of biology and information technology, dealing with the computer-based analysis of large biological datasets" (Fogel and Corne 2003). One such area resulting from these two (and various other) technologies is the study of gene expression data. Biologists perform experiments using various microarrays and the resulting data is stored in databases and analyzed by scientists, using tools for data analysis to extract gene

expression data. FIGURE 5.1 shows the interaction of various technologies involved in the microarray analysis process.

The traditional approach to research in Molecular Biology has been an inherently local one, examining and collecting data on a single gene, a single protein or a single reaction at a time. As a result of the Human Genome Project (Lander et al. 2001; Venter et al. 2001), there has been an explosion in the amount of data available about the DNA sequence of the human genome (as well as that of other living organisms). This has resulted in new and better understanding of the various genes including identification of a large number of genes within these previously unknown sequences

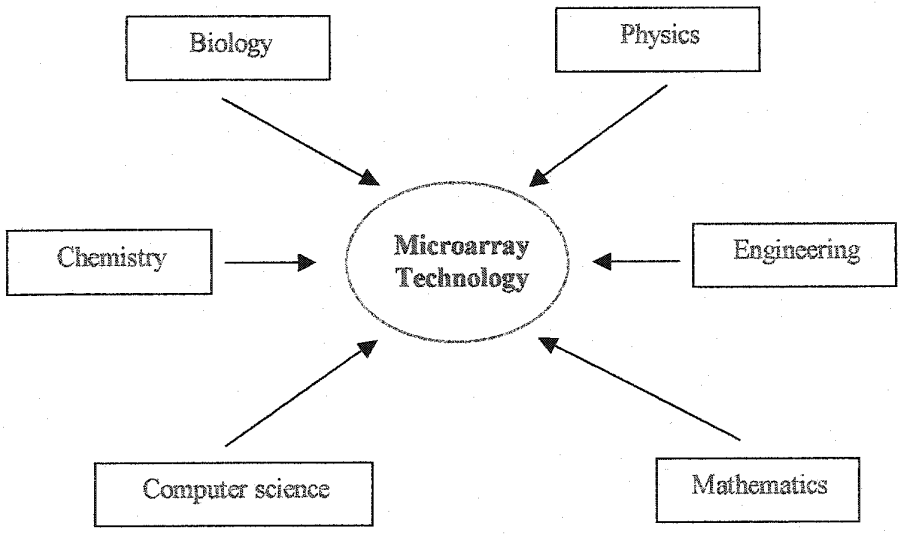


FIGURE 5.1: Microarray technology incorporating knowledge from various disciplines (Heejun Choi 2003)

To better understand the kind of data generated, we first give a brief introduction of the data generation process.

5.2. Overview of Microarray Process

The objective of this section is to provide an overview of the generation methodology of gene expression data. A good understanding of these methods and the biology behind the data is needed to have an intuitive idea of the nature of the problem. This in turn will help us better adapt the algorithm in an appropriate way for solving a particular problem. We will start this section by defining a few commonly used terminologies in molecular biology.

5.2.1. DNA

The DNA (or deoxyribonucleic acid) is a nucleic acid, contained in all cells of a living organism, which contains a set of genetic instructions that determines the development and functioning of that living being. It is a complex giant

molecule that contains, in chemically coded form, the information needed for a cell to make proteins. It determines the order in which amino acids are joined together to make a specific protein in a cell. A DNA molecule is made up of repeating entities called nucleotides (Saenger and Cantor 1984). Each nucleotide contains a sugar, a phosphate, and a base. There are four different nucleotides, or bases: adenosine (A), cytosine (C), guanine (G) and thymine (T).

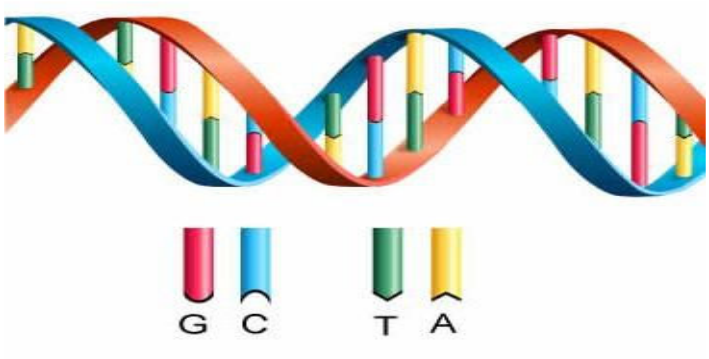


FIGURE 5.2 A double Helix DNA

In living organisms, DNA does not usually exist as a single molecule but as a pair of molecules that are held tightly together. The two halves are joined together by the bases forming pairs. The bases form into two specific base pairs: adenine with thymine and guanine with cytosine. The sequence of these base pairs in the DNA acts as a code, carrying genetic information about the sequence of amino acids in proteins. Such a DNA is referred to as a double stranded DNA, with the second strand being a reverse complement strand. Two complementary polynucleotide chains form a stable structure known as the DNA double helix (FIGURE 5.2), which is how DNA is normally found though other possibilities like single stranded are also possible.

5.2.2. Gene

A gene is the basic working unit of the DNA. Genes hold the information to build and maintain an organism's cells and pass genetic traits to offspring. The concept of genes, started when Gregory Mendel studied the biological inheritance of traits in living organisms from parents to offspring. However, it was not until much later when DNA was identified as a genetic material during the 1940s (Saenger and Cantor 1984).

Each gene contains a particular set of instructions, usually a coding for a particular protein. A modern working definition of a gene is "a locatable region of genomic sequence, corresponding to a unit of inheritance, which is associated with regulatory regions, transcribed regions, and of other functional sequence regions " (Pennisi 2007). They determine a particular characteristic in an organism such as hair color or eye color. Genes carry information that is passed on to future generations. Each human cell (except red blood cells) contains around 25,000 genes. At a given point of time, only a subset of these genes is active. Genes also undergo mutation when their DNA sequence changes.

5.2.3. RNA

RNA (Ribonucleic Acid) is a similar molecule to the DNA and is made up of a long chain of nucleotide units. RNA differs from DNA is that (1) it is usually found in a single strand whereas DNA is usually double stranded, (2) its nucleotides contains the sugar ribose (as opposed to deoxyribose), and (3) it has organic base uracile (U) rather than thymine.

Most RNA molecules act as cellular intermediaries in that they convert the genetic information stored in DNA into the proteins that provide cells with structure enabling them to carry out metabolism. RNA is transcribed from DNA by enzymes called RNA polymerases. Genetic information coming from DNA is transferred to 3 types of RNA:

- Ribosomal RNA (rRNA)
- Messenger RNA (mRNA)
- Transfer RNA (tRNA)

The rRNA is associated with small cellular structures known as ribosomes, the protein factories of the cells. The tRNA is part of the process of making proteins from instructions present in the genetic code by joining amino acids (a process called translation). The mRNA contains this genetic code copied from the DNA sequence to produce a complementary RNA (a process called transcription) (Crick 1970). Thus mRNA is transcribed from a gene and then translated by ribosomes in order to manufacture a protein. When a gene is active, the coding and non-coding sequences are copied producing an RNA copy of the gene's information, contained in mRNA. mRNA processing extracts the coded part of the gene (known as *Exons*) from the non-coded parts (known as *Introns*).

Thus by measuring the mRNA of a cell, one can estimate the genes that are active (amongst the protein producing ones). This has great biological significance since an excessive value (or recessive value) can indicate the abnormal behavior in the genetic code of the genes.

5.2.4. Complementary DNA (cDNA)

cDNA stands for complementary DNA. cDNA is synthesized from a mRNA template using an enzyme called reverse transcriptase. The resulting cDNA is single-stranded. This process is called reverse transcription (RT). If we want to express a protein in a cell which does not otherwise express that protein, we can transfer the cDNA that codes for the protein to the recipient cell. This is required in some experiments such as microarray. In this case a double-stranded cDNA is required which can be produced by another round of DNA synthesis following the first strand synthesis. The purpose of converting mRNA to cDNA is mainly for the analysis of the template mRNA because DNA is much stable than RNA. Once mRNA is converted to cDNA, the cDNA can be used for generating many copies of the DNA sequence in a process called amplification in a laboratory technique called RT-PCR (Reverse Transcription Polymerase Chain Reaction), as probe for expression analysis and for cloning of the mRNA sequence.

5.2.5. Microarray Chip

First developed at Stanford University, microarrays are glass slides on which cDNA has been deposited by high-speed robotic printing. Microarray technology incorporates sequence resources created by the genome projects and other such sequencing efforts to measure which genes are expressed in a particular cell type of an organism, at a particular time, and under a given condition (Mount 2004).

5.2.6. The Data Extraction Procedure

This subsection describes the procedure of using microarray chips to extract gene expression levels of different samples under given conditions, as will typically be the case in our datasets used in further experiments in this chapter.

Gene Expression data is generated by DNA chips and Microarray techniques involving DNA sequences from two different samples – one a normal (reference) sample and the other a test sample (other different conditions are also taken like different time points or different environmental conditions) (Madeira and Oliveira 2004). Messenger RNA (mRNA) is extracted from each of the samples, then reverse transcribed into a more stable complementary DNA (cDNA). These two samples are fluorescently labeled using two difference dyes. Infected samples are usually labeled red (cyanine 5) and unaffected samples green (cyanine 3). At this stage the two samples are combined and the resulting mixture is applied to the microarray chip. This step is called hybridization. The cDNA from the sample binds to its complementary sequences from the cDNA bases on the chip. Once the genes have been expressed, they are rinsed and fed to a laser.

The light from the laser excites the different dyes generating spots of different colors. Green spots indicate that the test substance has lower activity than the reference substance, red spots indicate that the test substance is more abundant than the reference substance; yellow spots mean that there is no change in the activity level between the two populations of test and reference substance. Black represents areas where neither the test nor the reference samples have control substance bound to the target DNA. The information is fed to a computer which calculates the red and green on each spot which indicates which genes are expressed or unexpressed in the test and reference samples. Measuring the quantity of label on each spot then yields an intensity value that should be correlated to the abundance of the corresponding RNA transcript in the sample. This step is known as the data quantification and the resulting data is then stored for further analysis. The process involved in gene expression analysis is shown in Figure 8.3.

So far we have discussed the experimental steps to obtain the gene expression data. Once the expression values are quantified in numerical form, they might be further transformed (normalization, filtering, etc) using various techniques and expressed as gene clusters. Sometimes, prior biological knowledge (feedback loop in Figure 8.3) is also utilized to determine the suitable number of clusters and to validate results obtained from the analysis. We now present in the next section the steps for analyzing the expression data and discuss the various techniques involved. For a detailed explanation of the process of extracting gene expression values, a discussion of the commercially available DNA chips, and various software tools, we refer the reader to (Mount 2004; Stekel 2003; Holloway et al.

2002) and the thesis report of Gundlapalli at the university of Louisville (Gundlapalli 2005) for a step by step process of obtaining gene expression data. A comparison of image analysis methods for quantification is reported in Yang et al. (Y. H Yang et al. 2002) and an overview of the process is described in Huber et al. (Huber, Von Heydebreck, and Vingron 2003).

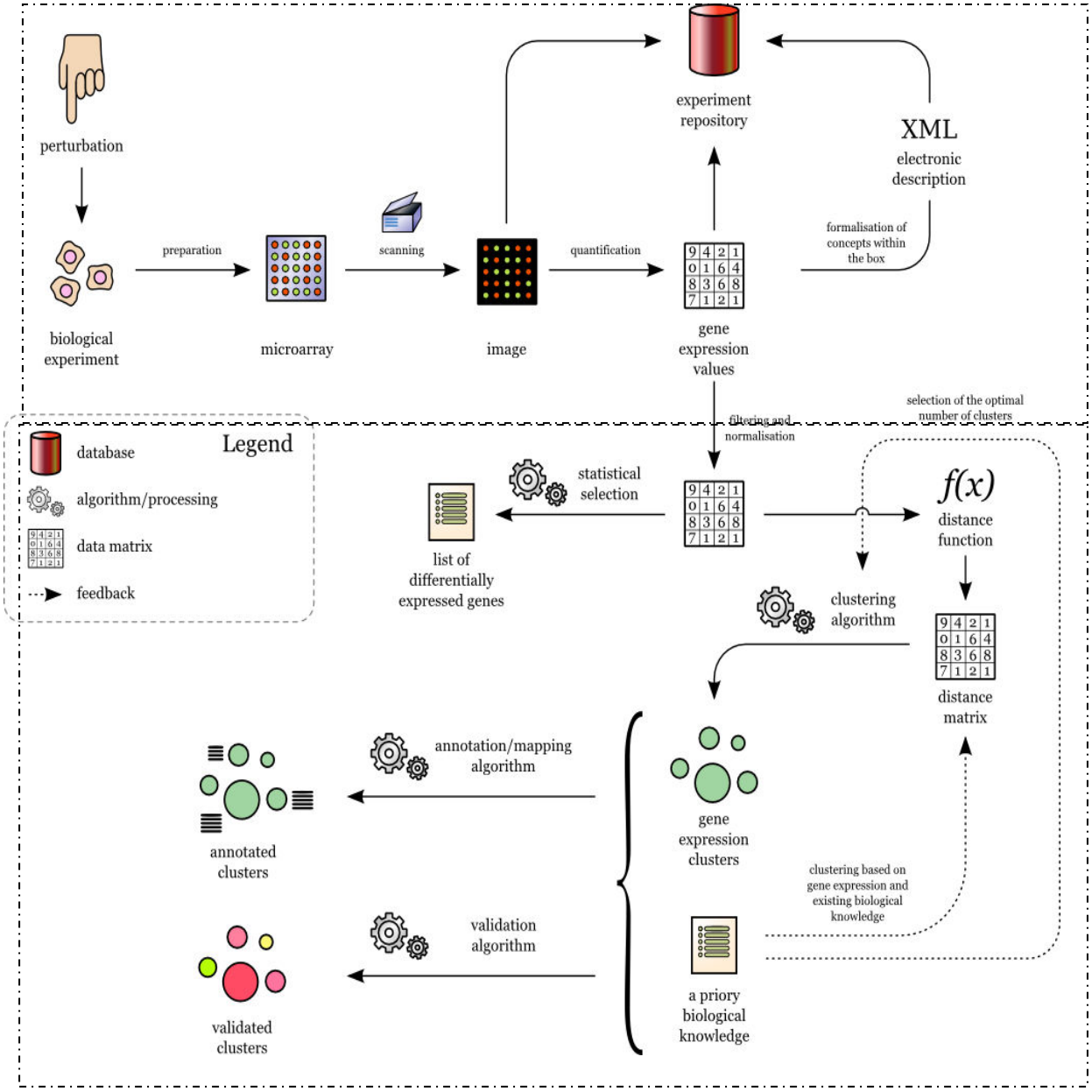


FIGURE 5.3 Overview of the process for gene expression analysis.

Source: http://www.stathis.co.uk/galleries/infographics/microarray_info_flow.png (reprinted with permission from the author)

5.3. Microarray Data Analysis

One of the usual goals in the analysis of microarray data is to identify which genes are involved in the process being studied. It involves the statistical analysis by various graphical and numerical means to identify genes or groups of genes whose expression profiles can reliably classify the different RNA sources into meaningful groups, or to group conditions based on to the expression of a number of genes. This is shown in the lower part of FIGURE 5.3.

5.3.1. Data Matrix Representation

A gene expression matrix is a matrix **A**, in which each row $\mathbf{a}_i$ represent a given gene, each column $\mathbf{a}^j$ represent a different experimental condition under which the expression value was obtained, and each cell A_{ij} usually represents the gene expression value for the given experiment. Table 5-1 below represents a typical gene expression data matrix (rounded to 4 digits after the decimal).

Table 5-1 Example of raw gene expression values from the colon cancer dataset **(Alon et al. 1999)**.

	Exp 1	Exp 2	Exp 3	Exp 4
Gene 1	8589.4163	5468.2409	4263.4075	4064.9357
Gene 2	9164.2537	6719.5295	4883.4487	3718.1589
Gene 3	3825.705	6970.3614	5369.9688	4705.65
Gene 4	6246.4487	7823.5341	5955.835	3975.5643
Gene 5	3230.3287	3694.45	3400.74	3463.5857

These expression values have usually been pre-processed in different ways for example during image analysis, quantification, screening out, etc. Micro-array technology is most commonly available in two forms i.e. GeneChip®, a trademark used by Affymetrix Inc, that relies on DNA oligonucleotide for chip manufacturing and Spotted Arrays developed at Stanford University, which uses a glass slide on which DNA is immobilized using a robot arm. Using these technologies, the expression levels of thousands of genes are simultaneously measured under a specific condition. The hypothesis underlying microarray analysis is that the measured intensities of each arrayed gene represent its relative expression level. Therefore, data analysis is based on the hypothesis that there are biologically relevant patterns to be discovered in the expression data.

The expression levels are usually compared between different states on a gene-to-gene basis. Statistical problems, however, arise as a consequence of the many technical peculiarities and their solution is, hence, a prerequisite to any meaningful subsequent interpretation of the data (Quackenbush 2002; Cuesta Sánchez, Lewi, and Massart 1994). Before levels can be compared appropriately, a number of transformations must be carried out to eliminate low-quality measurements and to adjust the different intensity levels. Moreover, a gene selection strategy is usually employed to select genes that have a significantly differential expression level between conditions.

We briefly discuss below the main sources of such noise or experimental errors that might occur when calculating expression levels in a microarray experiment and explain in the next section some of the most common pre-processing steps taken to transform data into a more statistically relevant form to facilitate comparison among expression levels.

5.3.2. Noisy Nature of Data

Real world data are rarely without "noise". Noise may be present in gene expression data due to 'experimental' (technical) noise or 'biological' noise (Klebanov and Yakovlev 2007).

Experimental Noise

Experimental noise can result at different stages of micro-array experiments described previously. For example, difference in intensity levels may be due to the processing of microarray and geometric transformation of images used for correcting and magnifying microarray images. A careful design and selection of the immobilized DNA and planning for the hybridization of array with samples is required. De-noising or filtering methods like linear filtering, mean filtering, Gaussian filtering, etc are usually used to minimize these effects (Heejun Choi 2003). Some of the sources of experimental noise are:

- Labeling efficiencies of the dyes
- Different amounts of Cy3 and Cy5 labeled mRNA
- Scanning parameters
- Spatial or plate effects, print tip effects
- Image translation, transformation and segmentation
- Unequal quantities of starting RNA, etc

Biological Noise

Biological variations are due to real differences, impurities or a misclassification between the different cell types and tissues being compared. Usually normalization techniques are used as a first transformation to adjust for the intensity values from hybridization to balance them appropriately so that meaningful statistical and biological comparisons can be performed.

Missing Values

Gene expression data sets often contain missing values due to various reasons, e.g. insufficient resolution, image corruption, dust or scratches on the slides, or experimental error during the laboratory process. Ideally, a repetition of identical experiments to validate downstream microarray analysis is carried out, but this is costly and time consuming. Another approach is to simply omit such genes or replace missing data with zero values. This

however, comes at a cost. Omitting an expression vector amounts to losing information about the experiment and replacing it with zeros might introduce unwanted noise into the data. Several statistical estimation techniques have been suggested such as (Oba et al. 2003; X. Wang et al. 2006; Kim, G. H Golub, and Park 2005; Brock et al. 2008). Details of such techniques are however, beyond the scope of this thesis and all datasets used in this thesis do not have missing values or such values have been previously pre-processed.

Irrelevant or Non-Discriminatory Genes

As described previously, a microarray experiment typically contains expression levels of up to several thousand genes. Most of these genes are not related to the condition under observations and hence are not significant to the data analysis. For example, in a DNA array sequence obtained from tumor and non-tumor tissues, we wish to find subsets of genes that are significantly over-expressed (or under-expressed) in tumor tissues. Usually, such a class (tumor or non-tumor) can be discriminated through only a small subset of genes whose expression levels strongly correlate with the class distinction. These genes are called informative genes. But the vast majority of genes may not be involved in cell division process and hence, do not show a differential behavior. Such genes are considered as noise (Cho et al. 2004; Quackenbush 2002). The combined effect of a large number of irrelevant genes can potentially overshadow the contribution of the relevant genes.

The effect of noise has been studied in the literature and is known to have an undesirable effect on clustering, particularly on gene expression data (Abdullah and A. Hussain 2005; Dave 1991; Sun and Nobel 2006). For example, Self-Organizing Maps (SOM), which is a topology-preserving neural network, will produce an output in which this type of data will populate the vast majority of clusters (Herrero, Valencia, and Dopazo 2001). In the presence of abundant irrelevant profiles, most of the interesting patterns may get diffused into only a few clusters and their identification is rather difficult and subjective. Similarly Hierarchical clustering, which is among the most commonly used methods for profiling gene expression data is also known to suffer in the presence of a large number of irrelevant data as is the case in gene expression (Tamayo et al. 1999). Usually feature (gene) selection techniques are employed on the dataset, either before or after other transformation(s) to select a subset of genes that are more discriminatory in identifying the sample classes.

As discussed above, data pre-processing is necessary to balance the values and allow for a meaningful biological comparison between genes. In addition, for biclustering algorithms, we also need to compare sample vectors and, hence, these also need to be appropriately normalized. We describe here some of the popular data transformation techniques that have been historically employed on gene expression data.

5.3.3. Gene Selection

Typically, gene expression datasets contain thousands of genes and relatively fewer samples, usually a few hundred or less. For example, the colon cancer dataset (Alon et al. 1999) contains a total of 6500 human genes and 62 samples while the Lung cancer dataset (Gordon et al. 2002) contains 12,533 human genes and 181 samples. Of these, only a subset of genes is discriminatory in nature between tumor and non-tumor tissues. Moreover, many of

these genes are noisy and redundant thus necessitating the need of feature selection. Such data can occur in two categories

- Where the sample label is known (for example healthy/non-healthy)
- Where the sample label is not known and the aim is to discover grouping of the samples

In the former case, we could utilize the prior class label on the samples and measure gene intensity on a gene-by-gene basis. This is referred to as supervised methods where sample labels are utilized to indentify discriminatory genes. However, we are concerned with an unsupervised approach here where such prior sample labels are not available. Here we consider two simple selection techniques popularly used for gene selection (Dudoit, Fridlyand, and Speed 2002; Alon et al. 1999; Cho et al. 2004; Dettling and Bühlmann 2002).

Relative Deviation

In this case, maximum and minimum gene expression values are considered for a particular gene across all the samples and only genes whose relative deviation is greater than a predefined threshold are retained. The formula for relative deviation is given by

(5.1) $$\delta_{low} \leq \left| \frac{\max(\mathbf{a}^i)}{\min(\mathbf{a}^i)} \right|,$$ where δ_{low} is the lower threshold value.

Hence, only genes whose relative deviation is greater than δ_{low} are considered for the clustering process. Genes whose relative deviation is smaller than δ_{low} are considered non-discriminatory and regarded as noise and discarded.

Absolute Deviation

Similar to relative deviation, absolute deviation is also used for gene selection process and considers the range of absolute values in a particular gene across all samples. Absolute deviation is calculated as,

(5.2) $$\theta_{low} \leq \left| \max(\mathbf{a}^i) - \min(\mathbf{a}^i) \right|,$$ where δ_{low} is the lower threshold value.

Again, the idea behind taking the absolute deviation is that only genes with a large enough deviation can be thought as capable of categorizing different samples. Usually a further pre-processing is used prior to applying either (or both) of relative and absolute deviation and a predefined value is used to threshold gene expression values using *ceil* and *floor* operations to reduce the effect of outliers.

5.3.4. Data Transformation

Data transformation, sometimes referred to as normalization or standardization, is necessary to adjust individual hybridization intensities of gene profiles such that intensity values between different genes are balanced and a comparison between them becomes meaningful (Dudoit et al. 2002; Cho et al. 2004). Transformation of raw data is considered an essential element of data mining since the variance of a variable determines the importance of that

feature.

Various authors (Quackenbush 2002; Cho et al. 2004), among other, have emphasized the importance of microarray data normalization and transformation. Cho et al. studied the effect of data pre-processing on gene data analysis on the minimum squared residue co-clustering algorithm. Their study has shown that appropriate data pre-processing has a profound effect on the resulting biclusters' quality. Wouters et al. (Wouters et al. 2003) and Kluger et al. (Kluger et al. 2003) have studied various data transformation schemes for various projection based techniques and spectral clustering respectively. For example, Kluger et al. have showed that simultaneous normalization of the rows and columns of a positive matrix, known as *bistochastization*, has an advantage for spectral clustering over simple scaling methods. Here we consider the following data transformation techniques used by Cho et al. (Cho et al. 2004) and Abdullah et al. (Abdullah and A. Hussain 2006) for their co-clustering algorithms.

Row/Column Standardization

This is a classical technique in data analysis usually refered to as *centering* or *scaling*. Column Standardization (CS) involves taking the difference between intensity values of a gene from the mean in units of standard deviation. Mathematically speaking, column standardization is defined as

$$A_{ij}^{C} = \frac{A_{ij} - \bar{\mu}_j}{\sigma_j}, \ \forall i \in 1..m, j = 1..n \tag{5.3}$$

Where $\bar{\mu}_j = \frac{1}{m}\sum_{i=1}^{m} A_{ij}$ and $\sigma_j = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(A_{ij} - \bar{\mu}_j)^2}$.

Similarly, Row Standardization (RS) is defined as

$$A_{ij}^{R} = \frac{A_{ij} - \bar{\mu}_i}{\sigma_i}, \ \forall i \in 1..m, j = 1..n \tag{5.4}$$

Where $\bar{\mu}_i = \frac{1}{n}\sum_{j=1}^{n} A_{ij}$ and $\sigma_i = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(A_{ij} - \bar{\mu}_i)}$.

Double Centering

The double-centering transformation is defined as

$$A'_{ij} = A_{ij} - \bar{\mu}_i - \bar{\mu}_j + \bar{\mu}_{ij}, \ \forall i \in 1..m, j = 1..n \tag{5.5}$$

Where $\bar{\mu}_i = \frac{1}{n}\sum_{j=1}^{n} A_{ij}$, $\bar{\mu}_j = \frac{1}{m}\sum_{i=1}^{m} A_{ij}$ and $\bar{\mu}_{ij} = \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n} A_{ij}$.

Double centering (DC) is symmetric with respect to the rows and columns of the data table. As a result of the double-centering, all absolute aspects of the data are removed. What remains are contrasts between the different

rows (genes) and contrasts between the different columns (samples) of the data table. Using this centering technique, the data matrix is projected onto a hyper plane that passes through the origin (Cho et al. 2004; Kluger et al. 2003).

Binormalization

Binormalization is a scaling of all rows and columns of a rectangular matrix. Various authors have presented binormalization algorithms (G. H. Golub and Varah 1974; Livne and G. H Golub 2004). Livne et al. (Livne and G. H Golub 2004) for example, present iterative algorithms to scale the rows and columns of a data matrix to have unit L2-norm. Thus for a given matrix **A**, we would like $\sum_j A_{ij}$ are equal for i=1...m and $\sum_i A_{ij}$ are equal for i=1...n. The resulting scaled matrix has the property that its rows and columns lie on a hyper sphere whose radii are given by $\sqrt{m}$ and $\sqrt{n}$ respectively.

Data Discretization

Hartigan (Hartigan 1972) defined a perfect constant bicluster as a sub-matrix $S(\hat{x}_i, \hat{y}_i)$ where all values within the bicluster are equal for $i \in \hat{x}_i$ and all $j \in \hat{y}_i$: $S_{ij} = \mu$, for any constant μ. In this scenario, noise within the bicluster can be thought of as an "additive noise" to a constant value making the bicluster non-constant. By using discretization, all such biclusters are hence once again set to a constant value. Discretization has been shown to benefit the data with respect to noise in certain cases (Abdullah and A. Hussain 2006). Several discretization techniques have been proposed, for example, the equal width technique divides the values of a gene into k equal intervals between the min and max values. Similarly, Equal frequency and threshold based techniques also exist. One such popular method is setting values below the mean to 0 and values above the mean to 1 to achieve a binary discretization.

Earlier works on data discretization in machine leaning can be found in (Chmielewski and Grzymala-Busse 1996; Dougherty, Kohavi, and Sahami 1995; Fayyad and Irani 1993). Data discretization may also help speed up the running time of various algorithms, for example, the crossing minimization paradigm of Abdullah et al. (Abdullah and A. Hussain 2006). It should, however, be noted that data discretization may also lead to loss of information where boundaries between over-expressed and under-expressed genes is narrow especially in the presence of noise (Y. S Choi and Moon 2007).

5.4. χ-Sim as a Biclustering Algorithm

Of particular interest in recent literature is the concept of bicluster[24], corresponding to the subset of genes and a subset of conditions with a high similarity score (Madeira and Oliveira 2004; Cho et al. 2004; Tanay, Sharon, and

[24] In the domain of bioinformatics, *co-clustering* is usually labeled as *biclustering* and correspondingly, *co-clusters* as *biclusters*.

Shamir 2002). As mentioned in Chapter (section 2 2.5), co-clustering is the simultaneously clustering of rows and columns to identify row clusters that have some correlation with a certain column clusters. In this section, we describe how χ-Sim can be used as an algorithm to discover biclusters in the underlying data. Particularly, we will focus on how to extract the biclusters automatically from gene expression datasets. As described in chapter 3, χ-Sim generates two similarity matrices – the row similarity matrix, **R**, and the column similarity matrix, **C**. Each of these similarity matrices can then be used as input to any standard clustering algorithm such as *k*-means, agglomerative hierarchical clustering, etc to achieve a hard clustering of the either rows (instances) or features (columns).

The biclustering effect is achieved since each the similarity matrices, **R** and **C,** is built on the basis of the other, thus implicitly taking into account a feature selection. For example, when clustering the genes (rows) in the data, each row similarity value between $\mathbf{a}^i$ and $\mathbf{a}^j$ given by R_{ij} is weighted by the feature similarity values C_{kl} for elements A_{ik} and A_{jl}. Hence, R_{ij} will be closer together (have a higher similarity value) if A_{ik} and A_{jk} are up-regulated in the gene expression matrix and the feature pair similarity C_{kl} have a high value. The argument is similar when computing values between conditions (columns), C_{ij}. Therefore, clustering the row similarity or column similarity matrices results in the biclustering of the gene expression dataset i.e. a set of genes that are up (or down) regulated under a subset of conditions. Observe that this is different from two-way clustering of the original data matrix where the row and column sets are independently clustered, because each clustering does not take into account the other.

Table 5-2 Biclustering of gene expression dataset

		Condition Clusters					
		$\hat{y}_1$	$\hat{y}_2$	…	$\hat{y}_l$	Max-Min	Index
Gene Clusters	$\hat{x}_1$	$\hat{x}_1\hat{y}_1$	$\hat{x}_1\hat{y}_2$	…	$\hat{x}_1\hat{y}_l$	$\max(\hat{x}_1\hat{y}_i) - \min(\hat{x}_1\hat{y}_j)$	$Ind_1 \in [1..l]$
	$\hat{x}_2$	$\hat{x}_2\hat{y}_1$	$\hat{x}_2\hat{y}_2$	…	$\hat{x}_2\hat{y}_l$	$\max(\hat{x}_2\hat{y}_i) - \min(\hat{x}_2\hat{y}_j)$	$Ind_2 \in [1..l]$
	…	…	…	…	…	…	…
	$\hat{x}_k$	$\hat{x}_k\hat{y}_1$	$\hat{x}_k\hat{y}_2$	…	$\hat{x}_k\hat{y}_l$	$\max(\hat{x}_k\hat{y}_i) - \min(\hat{x}_k\hat{y}_j)$	$Ind_k \in [1..l]$

Clustering the row and column similarity matrices result in a checkerboard structure corresponding to clustering the genes into *k* hard clusters and the conditions into *l* hard clusters. Each element of the checkerboard, $\hat{x}_i\hat{y}_j$, is the mean value of the set of genes in cluster $\hat{x}_i$ under condition $\hat{y}_j$. We are interested in extracting sets of genes that are up-regulated under one of the condition clusters but not under the rest of the condition clusters. Note that this transforms to the problem of optimal assignment. One way to do this could be to form a matrix of differences between each pair of values $\hat{x}_i\hat{y}_j$ and use a standard optimal assignment technique, such as the Hungarian algorithm. Such a method, however, may not accurately depict the behavior of the same cluster across the different gene clusters. Indeed, we would like to select biclusters where the gene clusters show greater variations among the different conditions.

We thus use a simpler technique and calculate the difference between the maximum value and the minimum value for each gene cluster. This gives an indication of the variation (in terms of mean values) shown by different gene clusters across the different condition clusters. We can now select a gene cluster showing maximum variation across the condition clusters for each condition cluster. This is shown in Table 5-2. The column Index stores the index of the condition cluster corresponding to the maximum value in that row. Observe that we could select more biclusters than the number of conditions by simply putting to zero, the "max-min" value corresponding to the bicluster already selecting and repeating the selection process. This approach is both straight-forward and suitable particularly since most real datasets in our experimentation corresponds to only two condition clusters, for example, gene expression values corresponding to *tumurous* and *non-tumorous* tissue samples. Note that if the dataset is not pre-processed (such as by row centering), one needs to normalize the column "max-min" as, typically, gene expression values differ substantially which may result in an uneven comparison.

5.5. Related Work

In chapter 2, we discussed several co-clustering algorithms (section 2.5) proposed in the literature. Here we briefly discuss a few other biclustering algorithms that were discussed in Prelic et *al*.(Prelic et al. 2006) which we shall be using for comparison purposes with our algorithm.

Tanay et al. (Tanay, Sharan, and Shamir 2002) proposed an algorithm, known as the Statistical-Algorithmic Method for Bicluster Analysis (SAMBA) to discover biclusters. The model is based on a data matrix corresponding to a bipartite graph and uses statistical models to solve the problem by identifying bicliques in the graph. Depending upon whether a gene is up-regulated or down-regulated, the corresponding edges are assigned a weight. An edge with weight 1 corresponds to an edge present, and an edge with weight 0 corresponds to an edge absent. Biclusters correspond to heavy sub-graphs in the bipartite graph, which are computed using the sum of the weights of gene-condition (row-column) pairs in it including edges and non-edges. A merit function is used to evaluate the quality of a computed bicluster using SAMBA by computing its weight. SAMBA performs simultaneous bicluster identification using exhaustive bicluster enumeration, and avoids exponential runtime by restricting the number of rows the biclusters may have by assuming that row vertices have d-bounded degree, where d is a user-specified parameter. The model is also able to search for non-complete bicliques in the bipartite graph in order to accommodate the effect of noise in the data.

The Order Preserving Sub Matrix (OPSM) technique proposed of Ben Dor et al. (Ben-Dor et al. 2003) assumes a probabilistic model of the data matrix. They define a bicluster as a group of rows such that the expression values in all features increase or decrease simultaneously. This is an NP-Hard, thus, the algorithm greedily searches for biclusters for which there exists a permutation of columns under which the sequence of values in every row is strictly increasing. OPSM results in finding large order preserving sub-matrices. The algorithm starts by evaluating all (1, 1) partial models (of which there are $m(m\text{-}1)$) and keeps the best ℓ of them where ℓ is the number of partial models. This is followed by expanding the bicluster size by increasing rows and columns until we get ℓ ($[s/2]$, $[s/2]$) models, which are complete models, and where s is the size of the model. The algorithm finally outputs the best

model.

Similar to the OPSM algorithm presented above, Murali and Kasif (Murali and Kasif 2002) have proposed a representation for gene expression data called conserved gene expression motifs or xMOTIFs (biclusters). They define an xMOTIF as a subset of rows whose values are conserved across a set of conditions if the gene is expressed with the same abundance in all the samples. Note also that the algorithm assumes the expression values can be in a fixed number of states, hence the concept of up-regulated genes and down-regulated genes. The expression value of a gene is conserved across a subset of conditions if the gene has the same state under each of the conditions in a subset. xMOTIF aims to find the largest bicluster containing the number of conserved genes, which forms the merit function. Certain other conditions about the number of conditions in which the gene should be up-regulated and the subset of genes not in the xMOTIF showing conservation among subsets of conditions in the bicluster are also taken into account when finding xMOTIFs in the data.

The Iterative Signature Algorithm (ISA) was proposed by (Bergmann, Ihmels, and Barkai 2003; Ihmels, Bergmann, and Barkai 2004) and considers a bicluster as a transcription module. Rather than alloting each gene to a single cluster, we assign both genes and conditions to context-dependent and potentially overlapping biclusters. ISA searches for the modules encoded in the data by iteratively refining sets of genes and conditions until they match a given definition. Each iteration involves a linear map, induced by the normalized expression matrix, followed by the application of a threshold function. This is achieved by selecting a seed vector r_o which is typically a sparese 0/1 vector across the conditions. This vector is then multiplied with the transpose of the gene expression matrix, **A**, and the results are thresholded (using with row/column centering).

The Bimax algorithm was used as a reference method in a comparative study of biclustering algorithms by (Prelic et al. 2006). The algorithm uses on 0's and 1's which can be obtained by discretization as a prior pre-processing step. A bicluster is then defined as a submatrix containing all 1's i.e. a set of genes that are up-regulated in a set of conditions. To this end, Bimax utilizes a divide and conquer approach for finding inclusion maximal biclusters. Bimax tries to identify areas in the matrix **A**, which contain only zero values and exclude them from further processing. The algorithm is then applied recursively until perfect biclusters can be found. If two (or more) resulting submatrices are found after removing biclusters of 0's that do not overlap, the submatrices can be processed independently.

Finally, we describe another co-clustering algorithm proposed by (Cho et al. 2004) that does not take into account a correspondence between row and column clusters as such, but consider sub-matrices formed by them with the overall aim to minimize the sum of squared residue within the sub-matrix. Cho et al. proposed an algorithm that is based on algebraic properties of a matrix. The algorithm runs in an iterative fashion and on each iteration, a current co-clustering is updated such that sum of squared residue is not increased. In other words, the algorithm monotonically decreases and converges towards a locally optimal solution. A survey of most commonly used biclustering algorithms can be found in (Madeira and Oliveira 2004).

5.6. Effect of Noise on χ-Sim

As discussed previously, real data is rarely perfect and it is important to validate the performance of the algorithm in

the presence of noise. We therefore used a synthetic dataset as a noise model to explore the robustness of the χ-Sim algorithm for detecting biclusters. A further aim of this experimentation is to test the best linkage strategy for use in the hierarchical clustering step of the algorithm.

5.6.1. Synthetic Dataset

Prelic et al. (Cho et al. 2004) has performed an analysis of five popular biclustering algorithms using a synthetic dataset to study the effect of noise on the algroithms. The construction of the dataset is based upon the model proposed by (Ihmels et al. 2004). In this model, biclusters are considered as transcription modules. Taking back our matrix **A**, with *m* genes and *n* conditions, let G be a set of genes with *k* gene clusters, G_1, …, G_k, and each of these genes are regulated by a set of common transcription factors. Similarly, let C be a set of conditions also with *k* condition clusters, C_1, …, C_k, in which these factors are active. They construct matrices with 10 biclusters (k=10) each containing 10 genes and 5 conditions, resulting in a dimension of 100x50 for the matrices. Moreover, they consider two types of bicluster concepts, non-overlapping and overlapping biclusters. For each of these concepts, two types of bicluster structures are considered based on Madeira – constant values and coherent values (additive biclusters). They introduce various noise levels in these matrices to study the performance of different algorithms on datasets with noise and 10 such matrices are generated for each noise level and each type. Since our algorithm does not generate overlapping clusters, we only considered non-overlapping hard clusters in our evaluation. These datasets are available from the authors' website along with supplementary material with detailed description of the generation procedure (see supplementary material of (Prelic et al. 2006)).

5.6.2. Validation

In the case of the synthetic dataset, since prior knowledge on both the gene and condition labels for the implanted biclusters is known a-priori, it is possible to measure the average bicluster relevance and bicluster recovery as proposed by the authors. Note that these labels are only used for validation purposes and not in the biclustering process in anyway. The average bicluster relevance is defined as

$$S_G^*(\mathrm{B},\mathrm{B}_{opt}) = \frac{1}{|\mathrm{B}|}\sum_{(G,C)\in \mathrm{B}} \max_{(G_{opt},C_{opt})\in \mathrm{B}_{opt}} \frac{|G \cap G_{opt}|}{|G \cup G_{opt}|} \tag{5.6}$$

Where B denotes the biclusters from a biclustering algorithm and B_{opt} is the set of optimal (implanted) biclusters. Note that each of these biclusters consists of a set of genes (a gene cluster G_i, $1 \le i \le k$) and a set of conditions (a condition cluster C_j, $1 \le j \le k$). The measure in Equation 5.6 is not usually symmetric since it reflects the average of the maximum match scores for all biclusters in B with respect to the biclusters in B_{opt} for the gene dimension. Similarly, the score given by $S_G^*(\mathrm{B}_{opt},\mathrm{B})$ quantifies how well each of the true biclusters is recovered by the biclustering algorithm under evaluation.

Note also that this is the gene relevance score and the condition relevance score $S_C^*(\mathrm{B},\mathrm{B}_{opt})$ can similarly be

computed by replacing G and G_{opt} by C and C_{opt} for the score matching in Equation 5.6. An overall match score can then be calculated as a geometric mean of the gene relevance score and condition relevance score. Each of these measures take a maximum value of 1 if all the implanted biclusters are exactly recovered by the biclustering algorithm ($B=B_{opt}$).

5.6.3. Results and Comparison using Synthetic Dataset

We present here the results of our experiments. For comparison purposes, we utilized the gene expression dataset with non-overlapping clusters and additive co-clusters implanted. Comparison is performed against seven other algorithms available from the 'Biclustering Analysis Toolbox (BicAT)' software by Barkow et al. (Barkow et al. 2006)[25]. The BicAT software implements the following algorithms: Iterative Signature Algorithms (ISA) (Ihmels et al. 2002); Cheng and Church's biclustering algorithm (CC) (Y. Cheng and Church 2000); Samba (Tanay et al. 2002); xMotif (Murali and Kasif 2002); Order Preserving Sub-matrix Algorithm (OPSM) (Ben-Dor et al. 2003). In addition, Prelic et al. present the results from Bimax and Hierarchical clustering (Single linkage using Euclidean distance) (Prelic et al. 2006).

The χ-Sim algorithm was implemented in MATLAB for computing both the row similarity and column similarity matrices and we use the MATLAB function 'linkage' and 'cluster' to generate and cut the resulting tree respectively. We set the number of gene and sample clusters to the original number (10 clusters each) to obtain 10 x 10 biclusters using the χ-Sim algorithm. As done in (Prelic et al. 2006), the experiment is run on 10 input datasets for each noise level and we report the average values respectively for that noise level.

The results of our experimentation are shown in FIGURE 5.4. The 7 algorithms from (Prelic et al. 2006) are shown in FIGURE 5.4 (a) and (c). For χ-Sim, we report the tests for its various configurations as follows:

- χ-Sim + S: using AHC with Single linkage on the output SR matrix
- χ-Sim + A: using AHC with Average linkage on the output SR matrix
- χ-Sim + C: using AHC with Centroid linkage on the output SR matrix
- χ-Sim + W: using AHC with Ward's linkage on the output SR matrix

This test fundamentally serves as a basis to assess the baseline performance of the different co-clustering algorithms to discover the implanted biclusters in the dataset. Additionally, the systematic addition of noise to the datasets enables us to study the sensitivity of the various algorithms to different levels of noise. Notice that the size of the datasets does not nullify the generality of the results since the main focus of the study is the structure of the matrix.

The tests were carried out on a non-overlapping dataset with an additive model while noise was imitated by adding random values drawn from a normal distribution. As pointed out by (Cheng and Church and Prelic et al.), for constant biclusters, the problem is much simpler and ISA, Samba and Bimax are able to recover most of the implanted biclusters (accuracy>90%) in the absence of noise. This is also true for the additive bicluster scenario studied here. However, both CC and xMotif performed particularly badly even with no noise. This can be explained

[25] available from http://www.tik.ethz.ch/sop/bicat/

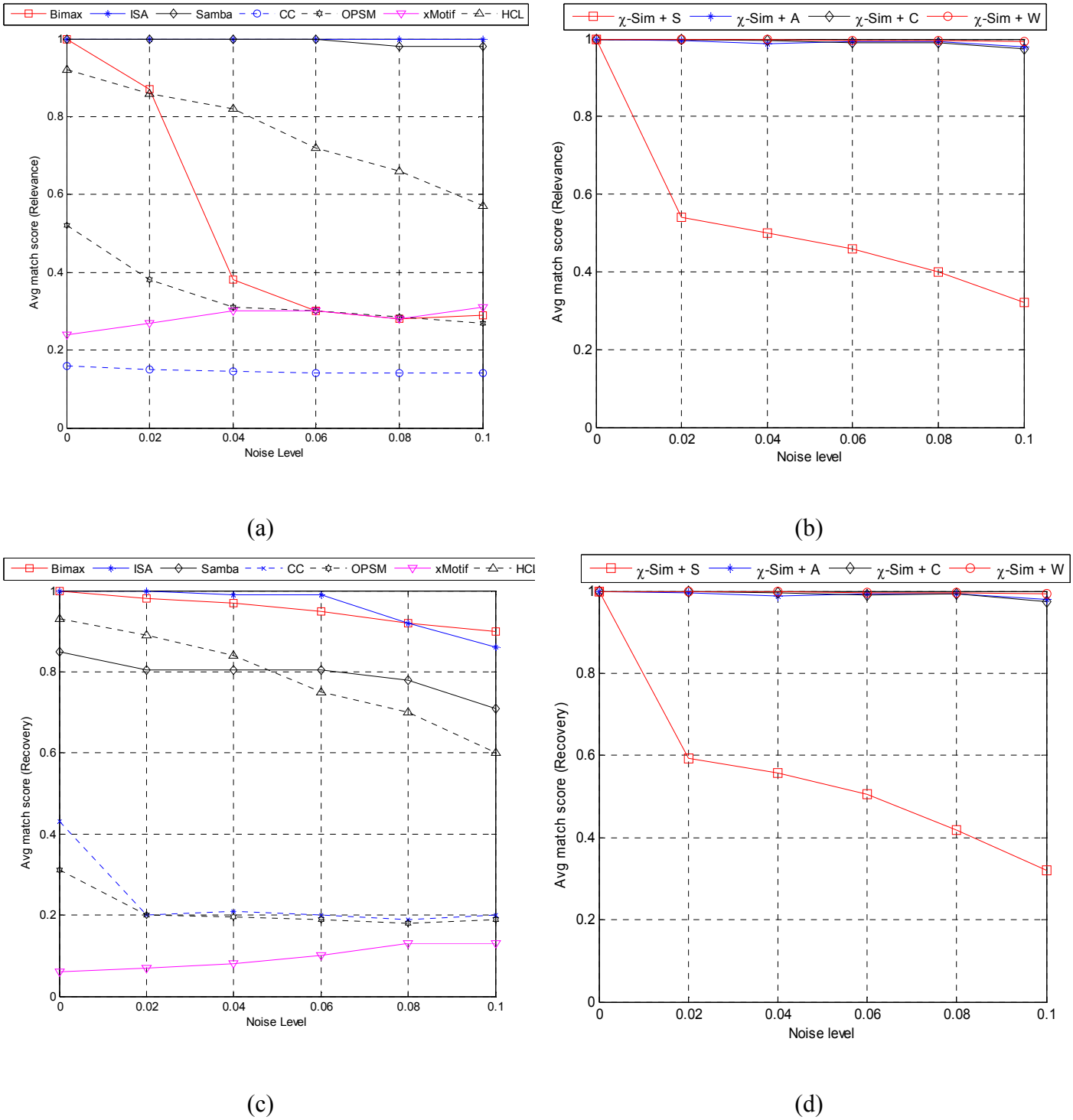
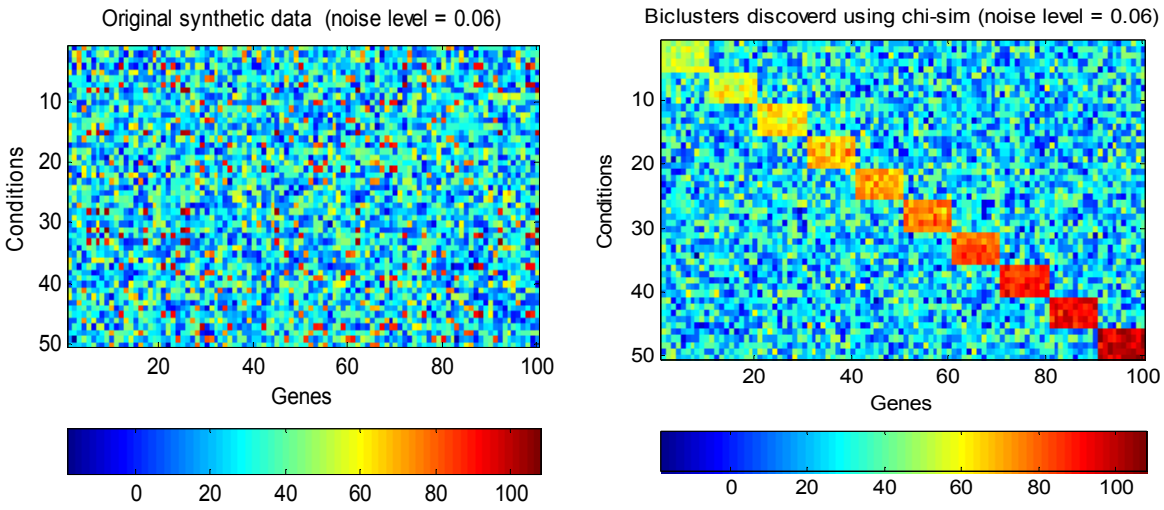


FIGURE 5.4 (a) and (c) show the average co-cluster relevance and average co-cluster recovery scores respectively for the methods compared, while (b) and (d) show the corresponding results for χ-Sim with different linkage methods

by the greedy nature of CC which can get stuck in local optima. Similarly, xMotif might not be suitable for finding additive biclusters, since it is primarily designed for coherent values. A somewhat surprise is the OPSM algorithm, since it looks for evidence of up-regulation and down-regulation. However, it is observed that OPSM doesn't give a good score on either bicluster relevance or recovery. The Bimax algorithm, which incorporates a discretization phase in the beginning, shows a clear behavior when noise is added to the data. Without noise, the implanted biclusters are readily identifiable and thus, generates a good discretization. However, with the addition of noise into the data, the frontiers between the up-regulated genes and background noise starts to become fuzzier and both the average co-cluster relevance and recovery scores are significantly decreased.

In the absence of noise, all variants of the χ-Sim algorithm with different linkage methods are able to recover the implanted biclusters. As noise is added to the data, Wards, Complete and Centroid linkages still manage to retrieve most of the implanted biclusters resulting in average bicluster relevance and recovery scores of more than 0.97. The single linkage algorithm, however, exhibits substantial degradation even with a small amount of noise (0.02) as shown in FIGURE 5.4(b) and (c). One explanation could be that with the addition of noise, some genes (condition) that lie farther away from their cluster centroids as a result of the noise and might have their closest gene (condition) pair might be in a different cluster. Since the single linkage algorithm is based on merging clusters on the basis of the closest similarity value of any member of a cluster with any other member of another cluster, this may degrade the accuracy of the retrieved biclusters.



(a) (b)

FIGURE 5.5 Initial and clustered data (heat map) using χ-Sim and Ward's linkage for Noise=0.06

Finally, we describe the effectiveness of χ-Sim in extracting biclusters from a given dataset. We use the so called *heat maps* to evaluate the results of the extracted biclusters. Heat maps draws a matrix based on its contents and assigns a color for each value or range of values. Thus, the color homogeneity of an extracted bicluster serves as a measure of the quality of the bicluster. We present here, the biclustering results from one of the datasets containing

a noise of 0.06 to demonstrate the ability of our proposed method to extract biclusters, even from noisy data. FIGURE 5.5(a) presents the original data whose rows and columns have been randomly permuted while FIGURE 5.5(b) shows the output matrix after extracting the biclusters. Since in the synthetic dataset we know both the number of row and column clusters, we set the values of k and l – the predicted number of row and column clusters – to the number of clusters in the dataset. Note that the extracted biclusters do not have to form the diagonal of the input matrix. In FIGURE 5.5, we sorted the rows and columns of the input matrix with respect to their predicted cluster labels for aesthetic reasons, as they are easy to recognize and interpret.

5.7. Results on Real Gene Expression Datasets

5.7.1. Gene Expression Datasets

We also validate the algorithm using real world datasets to analyze the capability of the algorithm to extract biologically meaningful biclusters. We used four popular microarray expression datasets used in (Cho et al. 2004). All these datasets were originally generated using the Affymetrix technology and are publicly available. We provide here a brief description of each of the datasets.

Colon Cancer This dataset contains expression levels for 6500 human genes across 62 samples used by Alon et al. (Alon et al. 1999). The dataset corresponds to Colon Adenocarcinoma specimen collected from several patients, while normal tissues were also obtained from some of these patients. We selected the top 2000 genes with highest intensity across the samples. The resulting dataset contains 2000 genes and across 40 tumorous and 20 normal colon tissues. Note that this dataset does not contain negative values and only 1909 of the 2000 genes are unique. We further pre-processed the data by removing genes with $|\max/\min| < 15$ and $|\max - \min| < 500$ leaving a total of 1096 genes.

Leukemia This dataset was used by Golub et al. (T. R. Golub et al. 1999) and contains 7129 genes across 72 samples. The dataset corresponds to RNA extracted from bone marrow samples of patients with leukemia at the time of diagnosis. About 47 samples were suffering from Acute Lymphoblastic Leukemia (ALL) while 25 samples were suffering from Acute Myeloid Leukemia (AML). We first used a floor value of 100 and a ceil value of 16000. Only genes with $|\max/\min| < 5$ and $|\max - \min| < 500$ were selected leaving a total of 3571 genes.

Lung Cancer Gordon et al. (Gordon et al. 2002) used a dataset of human tissue samples consisting of 31 samples of Malignant Pleural Mesothelioma (MPM) and 150 samples of Adenocarcinoma of the lung. All the MPM samples in this dataset contain relatively pure tumor. The dataset corresponds to surgical specimens from patients that underwent surgery at BWH (Brigham and Women's Hospital, Harvard Medical School, Boston, Massachusetts) from 1993 to 2001. We screened out genes having $|\max/\min| < 5$ and $|\max - \min| < 600$ from the original 12533 genes leaving a dataset with 2402 genes and 181 samples.

Mixed Lineage Leukemia (MLL) This dataset was used by Armstrong et al (Armstrong et al. 2002) having gene expression values of 12582 genes from 72 patients that had been diagnosed by pathologists as suffering from acute

lympoblastic leukemia (ALL, 28 samples), acute myeloid leukemia (AML, 28 samples) or mixed lineage leukemia (MLL, 20 samples). Several of the MLL genes are also expressed in either (or both) of ALL and AML. We screened out genes with |max/min| < 5 and |max - min| < 5500 leaving a total of 2474 genes across 72 samples.

Table 5-3: Description of the Micro array datasets used in this thesis

Dataset	Colon	Leukemia	Lung	MLL
No. of original genes	2000	7129	12533	12582
No. of samples	62	72	181	72
No. of sample classes	2	2	2	3
Sample class names	Normal (20) Tumor(42)	ALL(47) AML(25)	ADCA(150) MPM(31)	ALL(24) AML(28) MLL(20)
Relative deviation threshold, $\|max/min\|$	15	5	5	5
Absolute deviation threshold, $\|max - min\|$	500	500	600	5500
No. of selected genes	1096	3571	2401	2474

The pre-processing steps applied here have been used to enable direct comparison with the results of Cho et al. (Cho et al. 2004). It should be noted that more sophisticated techniques could be applied, for example, using previously identified genes shown to have discriminatory capabilities, to obtain a relatively more coherent data with less potential noise. However, we chose to employ rather simplistic pre-processing steps to emphasize the unsupervised nature of the task. A summary of the datasets used in this thesis is given in Table 5-3.

5.7.2. Analysis of Sample (Gene) Clustering

In this section, we evaluate the results on the condition clusters obtained from the co-clustering of real expression datasets. We utilize all the reduced gene subsets given in Table 5-3 and analyze the best transformation explained in section 5.3.3.

To validate the clustering of the samples, we use the category labels (e.g. tumor or normal) to evaluate the quality of the result using the widely used accuracy measure given by,

$$\text{Accuracy} = \frac{1}{n}\left(\sum_{i=1}^{l} t_i\right) \tag{5.7}$$

Where n denotes the total number of conditions, l denotes the number of condition clusters, and t_i denotes the number of samples correctly clustered into the sample class i. The accuracy is computed by first creating a confusion matrix whose entries (i,j) denotes the number of samples predicted to be in class i that belongs to the true category j. The class assignment is selected such that it maximizes the sum of t_i, for example using the Kuhn-Munkres

algorithm (Munkres 1957).

As mentioned earlier, different genes exhibit different range of values under given conditions and many of the values in real gene expression data could be noisy in nature. Therefore, we perform a pre-processing step such as those described in section 5.3.4. In order to determine the best pre-processing strategy, we test the different data transformation techniques for the given real datasets using the ward's linkage as it was found to be the best linkage strategy for both the synthetic (section 5.6.3) and real datasets. More precisely, we analyze the following transformation techniques: NT- No transformation, RS- Row scaling, CS- Column Scaling, DC- Double Centering, NBIN – Binormalization, and DISC- Discretization;

Additionally, we test another transformation which is formed by a combination of both RS and CS. Row scaling is performed to transform the values in the rows so as to enable a better comparison between two pair of rows while minimizing the effect of outliers. Similarly, column scaling is performed to scale the different values in each column for better comparison between pairs of columns. Since χ-Sim involves comparing both row pairs and column pairs, we use two different transformations of the original matrix, **A**, depending on whether we are computing the **R** or **C** similarity matrices. Denote the matrix **A** whose rows have been transformed based on row scaling as $\mathbf{A}_R$ and the matrix **A** whose columns have been transformed using column scaling as $\mathbf{A}_C$. Then, equations (3.22) and (3.23) at each iteration t could be re-written as

$$\mathbf{R}_{ij}^{(t)} = \begin{cases} 1 & \text{if } i = j \\ \left[(\mathbf{A}_R \mathbf{C}^{(t-1)} \mathbf{A}_R^T) \otimes \mathbf{NR} \right]_{ij} & \text{otherwise} \end{cases}$$

$$\mathbf{C}_{ij}^{(t)} = \begin{cases} 1 & \text{if } i = j \\ \left[(\mathbf{A}_C^T \mathbf{R}^{(t-1)}) \mathbf{A}_C \otimes \mathbf{NC} \right]_{ij} & \text{otherwise} \end{cases}$$

Thus, in total 7 data transformation techniques were studied and the results are presented in FIGURE 5.6. From the Figure, it is evident that row centering generally provides the lowest accuracy amongst all the data transformation techniques providing an accuracy of 0.74, 0.85, 0.99 and 0.65 for the colon caner, leukemia, lung and MLL datasets respectively. The lower score of 9 of the datasets could be explained by the observation that since the rows denotes the genes, using row transformation do not scale the different scaling factor of the different genes when comparing column pairs. For example, one gene might exhibit a value of 1.0 and 2.0 for tumorous and non-tumorous datasets while a second gene might exhibit the values 25 and 50 respectively. Even though both the genes are down-regulated in the tumorous tissues by the same factor, the scale of the values is different. Scaling by the row may distort this even further thus resulting in a worse accuracy value.

On the other hand, using the combination of both row scaling and column scaling depending on whether we are comparing row pairs or column pairs yield the best results yielding accuracy values of 0.9032, 0.9722, 1.0 and 0.85 respectively for the colon, leukemia, lung and MLL datasets. As mentioned previously, using a combination of both RS and CS leads to a better scaling when comparing row pairs or column pairs and hence, better results. Amongst the datasets, the lung cancer dataset is found to be the easiest to cluster resulting in a an average accuracy value of 0.95 among all the different transformation methods, followed by leukemia with an average accuracy of

0.91, MLL with 0.85 and colon cancer with 0.78.

In order to specify the best strategy, we consider the average accuracy results of each technique across all the datasets. Using the combination of RS and CS yields an average accuracy of 0.93 followed by NT with 0.917, while using RS alone results in the worst average accuracy value of just 0.8091. Therefore, using RS + CS is the only data transformation technique that actually increases the average accuracy rate of the different datasets.

Finally, we make a comparison of the clustering accuracy of the results obtained by χ-Sim with those obtained by Cho. et *al*. They proposed the Minimum Sum Squared Residue Co-Clustering (MSSRCC) algorithm and test their method on the 'reduced gene subset', which is the same dataset used in our experiments as well as using a supervised method for selecting highly discriminative genes that have been reported in the literature (which they call the 'selected gene subset'). We report the result from our method using RS+CS and using Ward's linkage algorithm against the best performing strategy of MSSRCC on both the 'reduced gene subset' and the 'selected gene subset' as reported in Cho. et *al*. Note that since we perform our experiments on the same datasets using the same pre-processing (only for the 'reduced gene subset'), a direct comparison between the results is possible.

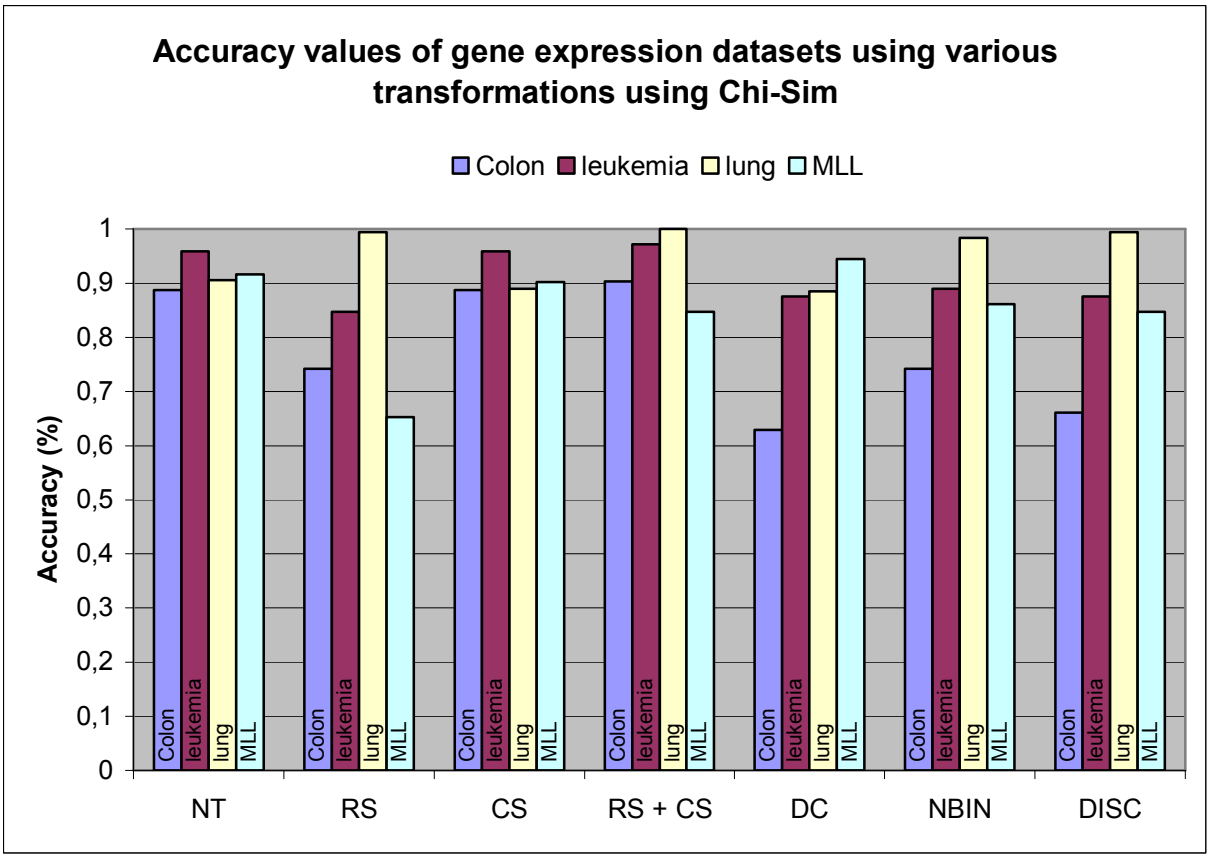


FIGURE 5.6 Accuracy values on the reduced gene expression datasets in Table 5-2 using various pre-processing schemes. Abbreviations: NT- No transformation; RS- Row scaling; CS- Column Scaling; DC- Double Centering; NBIN – Binormalization; and DISC- Discretization.

Table 5-4 Comparison of Sample clustering of χ-Sim with 'RS+CS' and the best performing results from Cho. et al.

Datasets	Accuracy (%)		
	Reduced Gene subsets		Selected Gene subset
	χ-Sim	MSSRCC	MSSRCC
Colon	0.9032	0.8573	0.9274
Leukemia	0.9722	0.9314	0.9702
Lung	1	0.9969	-
MLL	0.8472	0.9340	0.9958

The results are summarized in Table 5-4. As can be seen from the results, χ-Sim using RS+CS performs better than MSSRCC on the Colon, Leukemia and Lung cancer datasets but gives a lower accuracy on the MLL dataset. In fact, on the Leukemia dataset, χ-Sim is able to outperform MSSRCC using a highly discriminating subset and for the Lung cancer dataset, χ-Sim is able to cluster the cancerous and non-cancerous tissues with 100% accuracy. Note that the results corresponding to MSSRCC were described by the authors as those resulting with their best strategy, while the results of χ-Sim are fixed on just one strategy. Indeed for the MLL dataset, χ-Sim using DC gives an accuracy of 0.9444 which outperforms MSSRCC for the reduced gene dataset. Moreover, the results obtained using χ-Sim are deterministic as opposed to many of the variations of MSSRCC which starts with a random initialization and converges towards a locally optimum solution.

5.7.3. Analysis of Gene Clusters

Unlike the synthetic data, however, we do not have a priori knowledge of the gene clusters. Hence, we use both visual heat maps and a biological knowledge background to asses the quality of the gene clusters. Coherence in gene clusters can be visualized using the checkerboard structures in the so-called *heat maps*. Genes that are biologically meaningfully related are either up-regulated or down-regulated under certain conditions and hence can be visualized using such maps. Error bars have been used to visually analyze the quality of the resulting gene clusters. Furthermore, gene clusters are also evaluated by verifying which genes previously found in the literature to be high discriminatory are grouped together. We used biological databases such as DAVID (Dennis Jr et al. 2003) and two ontology databanks[26] as well as analysis by biological experts to assess the enrichment of Gene Ontology biological processes in the discovered clusters[27].

We evaluated the quality of some of the gene clusters generated by the χ-Sim biclustering algorithm corresponding to the sample accuracy values as listed in Table 5-4. Since we do not know before hand the exact number or nature of the gene clusters, we generate a set of 100 gene clusters and select the top 3 gene clusters as

[26] http://www.uniprot.org/ and http://www.ncbi.nlm.nih.gov/gene

[27] We are greatful to Mr. Muhammad Ramzan of the Institut Nationale de la Santé et de la Recherche Médicale (INSERM), Unité 823, Grenoble for his help in analyzing GO biological processes.

described in section 5.4. These biclusters correspond to those gene clusters whose expression levels are clearly distinguished from other clusters (for example between tumor and non-tumor). A list of genes corresponding to each of these clusters can be found in appendix VI.

Gene Clusters of the Colon Cancer Dataset

As described above, we cluster the genes into 100 clusters and select the top 3 discriminating clusters (CO-42, CO-62,CO-63) across the two condition clusters (tumor, non-tumor) and plot the heat map corresponding these 6 biclusters in FIGURE 5.7(a). The gene clusters (CO-42), (CO-62) and (CO-63) contains 4, 17 and 13 genes respectively.

FIGURE 5.7(b) shows the mean expression level of each of the 3 gene clusters across the two condition clusters. As can be seen, the Figure clearly illustrates that χ-Sim captured homogenous gene expression patterns in the gene clusters. As mentioned in (Alon et al. 1999) and (Cho et al. 2004), the ribosomal genes are partly up-regulated in tumor samples and comparatively down-regulated in non-tumor (normal) samples. In fact, most of the ribosomal protein genes given in (Alon et al. 1999) can be found in two clusters (CO-5) and (CO-6) (see appendix VI). Most of these genes cluster together—as expected for genes that are regulated coordinately.

Furthermore, we perform gene ontology enrichment of the biological processes of each of the genes captured in the gene clusters by χ-Sim. (Alon et al. 1999) also describe a subset of genes, within the high intensity genes that are related to cellular metabolism. Indeed, by analyzing the cluster (CO-42), which is up-regulated for tumor tissues, we find that 7 of the genes contribute to *Cell proliferation & Differentiation* biological process while 5 of the genes are related to cell signaling and adhesion. This signifies that the genes grouped together by our χ-Sim algorithm produce genes that are enriched in biological processes related to cellular metabolism as would be expected in tumurous tissues.

Gene Clusters of the Leukemia Dataset

Similar to above, we cluster the genes into 100 clusters and select the top 3 discriminating clusters (Le-18, Le-29, Le-56) across the two condition clusters (ALL, AML) and plot the heat map corresponding these 6 biclusters in FIGURE 5.8(a). The gene clusters (Le-18), (Le-31) and (Le-56) contains 37, 29 and 27 genes respectively. FIGURE 5.8 (b) shows the mean expression level of each of the 3 gene clusters across the two condition clusters (ALL and MLL).

The clustering of the genes reveals a strong correlation between gene types within a cluster. The cluster (Le-18), for example, contains genes that are partly but highly expressed in tissues of AML and almost never expressed in tissues of MLL. Similarly, the genes in (Le-56) are highly expressed in MLL tissues. Each of these gene clusters contains highly discriminative genes, many of which have also been identified previously (for example in (Cho et al. 2004; T. R. Golub et al. 1999). Indeed the clusters (Le-18) and (Le-29), both of which are highly expressive in AML and down-regulated in ALL, contains 9 genes that were clustered together in two clusters (cluster-99 and cluster-33)

identified by (Cho et al. 2004). (See their supplementary materials[28] and Appendix VI).

Finally, we present the gene ontology enrichment of biological process for each of the identified gene clusters in Table 5-5. We focus on the same gene clusters that were identified automatically and presented in Figure 5.7 and Figure 5.8 since they were discovered on the basis of high discrimination among the different tissue conditions. The first column in Table 5-5 represents the identified gene cluster and the second column provides the main biological processes associated with genes of that cluster. The 'count' column gives the number of genes participating in that biological process. As can be seen from Table 5-5, the gene clusters contains high percentages of genes related to major biological processes. For example, (CO-42) is highly enriched in Biosynthesis & Respiration; (CO-63) in Cell differentiation and signaling; (Le-56) in Biosynthesis and regulation and Cell signaling and adhesion.

The results for Lung cancer dataset and MLL datasets, along with their heat maps, gene expression levels and gene ontology biological expression are given in appendix IV, while the list of all the genes of the extracted clusters is given in appendix V.

[28] Available at http://userweb.cs.utexas.edu/users/dml/Data/cocluster/geneNameClustered_leukemia.txt

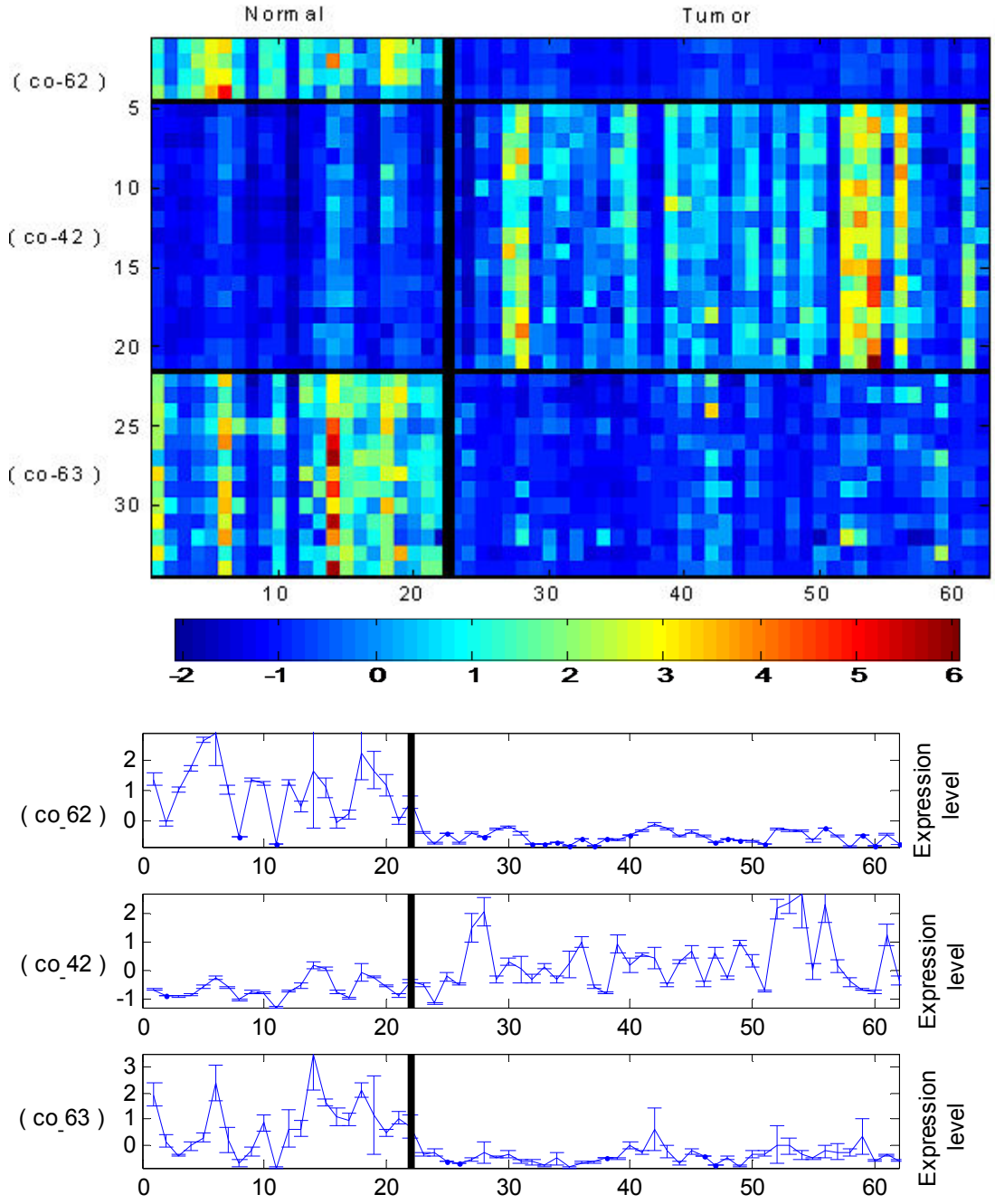


FIGURE 5.7 (a) Six highly discriminating biclusters in the Colon cancer dataset, and (b) the average gene expression levels corresponding to each of the biclusters.

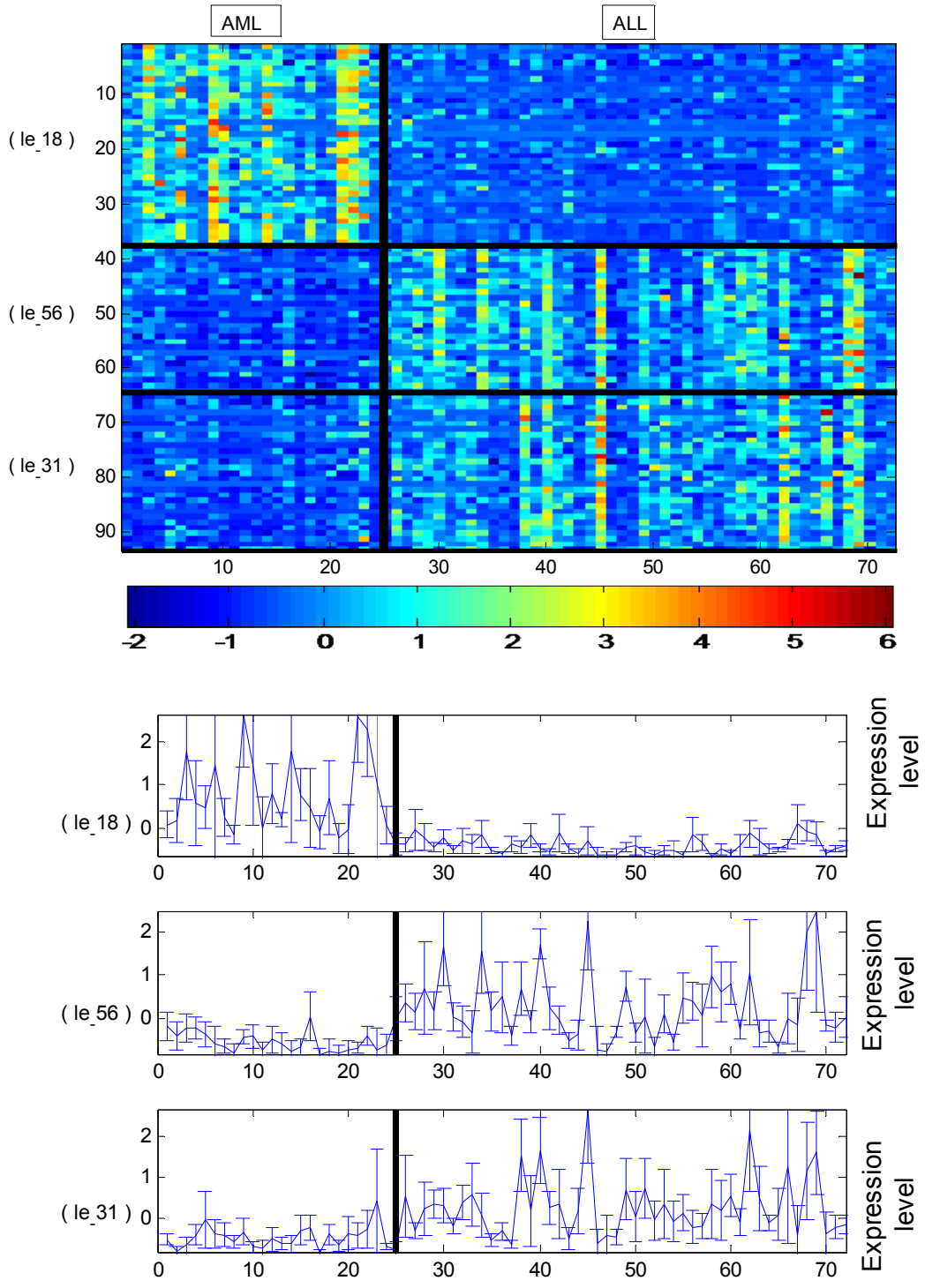


FIGURE 5.8 (a) Six highly discriminating biclusters in the Leukemia dataset, and (b) the average gene expression levels corresponding to each of the biclusters.

Table 5-5 Enrichment of GO biological processes in gene clusters

Cluster	Biological Process	Count	Percentage
(CO-42)	Biosynthesis & Respiration	8	47.1
	Homeostasis	5	29.4
	Cell proliferation & Differentiation	5	29.4
	Immune response	2	11.8
	Cell signaling & Adhesion	7	41.2
(CO-62)	Biosynthesis & Homeostasis	2	50.0
	Immunity & Signaling	2	50.0
(CO-63)	Immune responses	6	46.2
	Cell differentiation & Signaling	7	53.8
	Biosynthesis & Metabolism	6	46.2
(LE-18)	Biosynthesis & Regulation	13	35.1
	Metabolism & Homeostasis	17	46.0
	Immune responses	10	27.0
	Cell proliferation & Differentiation	10	27.0
	Cell signaling	10	27.0
(LE-56)	Biosynthesis & Regulation	14	51.5
	Metabolism & Homeostasis	12	44.4
	Immune responses	6	22.2
	Cell proliferation & Differentiation	8	29.6
	Cell signaling & Adhesion	14	51.5
(LE-31)	Biosynthesis & Regulation	11	37.9
	Metabolism & Homeostasis	11	37.9
	Immune responses	10	34.5
	Cell proliferation & Differentiation	8	27.6
	Cell signaling & Adhesion	6	20.7

5.8. Conclusion of the Chapter

In this chapter it was successfully demonstrated that the proposed χ-Sim algorithm does perform biclustering and its results are better than other contemporary traditional techniques. We described the micro array analysis process and several data transformation techniques were tested with the proposed technique. One of the challenges of biclustering is the amount of noise present in the data. We successfully demonstrated that χ-Sim was able to recover the implanted clusters in the presence of noise in the gene expression data using simulated dataset at different noise levels. We proposed a variant of the proposed algorithm that combines two pre-processing techniques that resulted in an improvement of the accuracy of our results. Furthermore, we developed a technique to automatically extract discriminating biclusters in the gene expression dataset. The quality of the results was verified by biclustering several publicly cancer dataset and validating the results by measuring the accuracy of the clusters recovered and analyzing the biologicial ontology enrichment of the gene clusters.

Chapter 6
Conclusion and Future Perspectives

This chapter presents the main contributions of this work. We also discuss various limitation of the current work as well as a description of possible developments that can be pursued as future work. Clustering is an unsupervised learning technique, so the cluster membership is determined only by the spatial and statistical characteristics of the data. The data is usually of high dimensionality. In such cases, traditional similarity measures, which assume orthogonoality between features, do not always perform best. This thesis incorporates several ideas coming from cognitive science, information Retrieval, and graph theory to propose a new measure of calculating similarity based on the structure and statistics of the data. The aim of this thesis was to propose a similarity based approach that incorporates the idea of co-clustering, while taking advantage of individual (feature or object) rather than group (cluster) relationships.

6.1. Summary and Contributions of the Thesis

The main contribution of the thesis is summarized as follows.

In chapter 2, the necessary background material and a short review of the current state-of-the-art is given. We started by introducing different kinds of data representation used in the literature and vocabulary that was used in the rest of the thesis. We also discussed the limitation of traditional similarity measures (such as Cosine, Tanimoto, etc) when applied to high dimensional and sparse data and discussed the various alternative approaches that have been proposed in order to deal with this limitation. An overview of the various families of co-clustering algorithms was also given.

In chapter 3, we proposed a co-similarity based measure was introduced. The proposed technique takes into

account the structure of the resulting bipartite graph of a document-by-word matrix. When comparing the similarity between documents, not only are the words contained in both documents considered but we also consider *similar* words that may result as a consequence of various linguistic phenomenon such as synonymy and an individual writer's style and vocabulary usage. In this regard, we proposed a co-similarity measure than incorporates the underlying statistical similarity between words when computing the similarity between documents and vice versa. As a result, each of our document-document similarity matrix, **R**, and word-word similarity matrix, **C**, in itself contains information coming from the other.

We further examined the algorithm from a theoretical point of view based on the concept of higher order co-occurrences in graph theory. It was observed that iteratively computing the two similarity matrices results in taking redundant paths into account which do not contribute any new information between the document or word pairs being considered. We proposed a way to compute only elementary (non-redundant) paths of up to order-3. Pruning was also introduced as a measure to reduce similarity generation from random (or low) co-occurrences. Such occurrences usually make the boundaries between clusters rather fuzzy as they mostly occur because of less informative words.

An inherent characteristic of the proposed method is that we can easily incorporate external knowledge (such as category labels of documents) from a training data to better influence the evolution of similarity measures. We introduced two ways to take advantage of such data – by adding dummy features to enhance links within nodes of a cluster and by diminishing the similarity values between elements (objects or features) that belong to different categories. In this manner, we extend the similarity measure into a supervised one.

In chapter 4, we provide experimental evidence to verify the advantages of our proposed co-similarity measure. The results of clustering are usually verified by measuring the accuracy of different clustering algorithms on a dataset as compared to an existing (usually manual) classification of the data. The different parameters of the proposed measure were explored and it was shown that χ-Sim performed much better than other clustering and co-clustering algorithms tested on the Newsgroup and SMART datasets. Similarly, the supervised version of the measure was found to compare or outperform several state-of-the-art methods on the Newsgroup, Reuters and Lingspam dataset. An important result here was that pruning was shown to significantly enhance the clustering accuracy, particularly when tested on dataset generated using an unsupervised feature selection mechanism, thus re-enforcing the notion that pruning can be an effective way of dealing with data where boundaries are rather fuzzy and only elements with a strong connectedness need be considered for generating similarities.

In chapter 5, we apply the χ-Sim algorithm to perform the task of biclustering gene expression data. As opposed to document-by-term matrices, the matrices resulting from gene expression data are usually full. Moreover, experimental errors usually make such data prone to noise and any effective algorithm must be able to extract biclusters in the presence of such noise. We performed clustering of both genes and conditions and proposed a way to automatically extract the k most significant biclusters that shows greater homogeneity within the bicluster and more variations when compared to other biclusters. In this regard, χ-Sim was able to recover the implanted biclusters even in the presence of a significant amount of noise. The results of biclustering on real cancer datasets are also verified from a biological process point of view.

6.2. Limitations of the Proposed Similarity Measure

The proposed co-similarity measure has been shown to work relatively well on the datasets tested. However, we would like to mention a few limitations of the approach

- The Algorithm is rather expensive in space and time for significantly large datasets. A complexity of $O(t.\max(m^2n,mn^2))$ will usually prohibit its usage on datasets such as large corpora containing several million documents and words, or the world wide web. Similarly, the corresponding similarity matrices are difficult to fit in the main memory of a computer. In Chapter 3 (section 3.6.3), we established a relationship between χ-Sim and the SimRank algorithm, and showed that χ-Sim can be considered as a generalization of the SimRank algorithm. In this regard, one could consider the various improvements, for instance (P. Li et al. 2010; Jia et al. 2010) that have been proposed for the efficiency of the SimRank algorithm and try to incorporate them into our χ-Sim measure.
- The algorithm is limited to homogenous features using the definition of the similarity function $f_s(,)$ as described in section 3.3.3. One may perform feature transformation prior to calculating the similarity values, as was done in Chapter 5 (section5.3.4) in order to make the features comparable since different genes have express with different intensity levels across the various conditions. However, the (present definition of the) algorithm remains inapplicable on datasets containing heterogeneous features such as age, income, etc.
- Calculating similarity values in the current way (section3.3.6) necessitates that all similarity values be recomputed with the addition of new documents (objects) and words (features). This is an expensive operation and demerits the use of the proposed measure where incremental updates are necessary. A possible line of exploration in this case could be to consider representative vectors of each row and column cluster (similar to the NC method proposed in section3.7.5). The row and column similarity matrices can then be computed on the basis of these representation vectors while similarity values between documents (or words), whenever necessary, can be computed based on a *cluster membership* value that represent the strength with which a vector represents the distribution of the clusters and the cluster similarity values of the two documents (words).

6.3. Future Work

The work described in this dissertation opens up a series of interesting lines of research for the future. Some of the interesting possible developments for this work include, but are not limited to, the following:

- The output of the co-similarity measure is two proximity matrices **R** and **C**. These matrices were used in both document clustering and categorization problems. It would be interesting to further study the properties of these matrices, for instance, whether they are (or can easily be modified to become) positive definite matrices. Studying these properties could further enhance the method to be used as kernel function

, such as for kernel based clustering and in Support Vector Machines.

- In this thesis, the L1 normalization was considered and was shown to perform well for the document clustering task. Additionally, a technique was demonstrated to calculate non-redundant paths of order-1,2 and 3 in chapter 3 (section 3.5). However, the similarity values generated for each length of paths are not directly comparable since the values at successive length of paths are usually increasing but the normalization factor remains the same. As a result, the combination of these similarity values needs a weighted combination of similarities. One possible solution could be that at each similarity values for $\mathbf{R}^{(1)}$, $\mathbf{R}^{(2)}$ and $\mathbf{R}^{(3)}$, the normalization step takes into account the number of paths of that length generated by each vector. If we can make the different similarity matrices comparable, a simple linear combination the matrices should suffice to find the overall similarity measure due to non-redundant paths between elements.

- One of the applications of the co-similarity measure, in this thesis, was for the task of document clustering. A closely related task could be that of information retrieval. Query matching is usually performed on the basis of keywords that are matched with different document vectors to retrieve a sorted list of documents that are similar to the query vector. Using the word similarity measure, we could enhance the query matching process to not only include the words in the query but also words that can be considered as similar to the words given in the query as dictated by the word similarity matrix. The task is similar to performing a *k*-nearest neighbor matching as described in section 3.7.5. One may also chose to cluster the data and use the NC approach first to select the relevant cluster followed by *k*-NN to select a list of documents matching the given query and sorted on the basis of their similarity values.

- Since co-clustering algorithms cluster both objects and features, one application of co-clustering algorithm could be that of feature selection. The co-similarity measure presented in this thesis takes into account the co-occurrence statistics in the data. One possible use of this concept, for example, could be to perform an initial grouping of the features and select representative feature(s) that combines a high degree of information for a given category and provides good connectedness (direct or higher order co-occurrences) to other features that span the same category but few links to features of other categories. One could reduce the dimension by initially selecting features based on some criteria (such as mutual information of features and object categories) to select only discriminatory features and also to speed up the process.

Résumé et Perspectives

La notion de similarité est centrale dans la fouille de texte en général mais plus particulièrement pour le classification et classement des documents. Le but est de trouver les documents similaires et de les grouper dans des classes ou « cluster ». Classiquement, on utilise des mesures de similarité pour classifier les documents et la représentation des données se base sur le modèle vectorielle. La similarité est évaluée à partir du nombre de termes communs entre les documents. L'hypothèse faite dans cette thèse est que l'utilisation seule des termes communs est insuffisante quand on travaille dans un espace de dimensions élevé. Ainsi, deux document peuvent être similaire sans partager pourtant des mots en communs. Nous avons introduit une nouvelle approche pour le co-clustering qui est basée sur la définition de la mesure de co-similarité χ-Sim. En prenant en compte de façon itérative la similarité entre termes dans le calcul de la similarité entre documents et réciproquement celles des documents dans celle des termes, nous avons montré que cette méthode est capable de mettre automatiquement en relation des documents et des termes ne partagent pas explicitement des informations grâce à la découverte automatique de co-occurrences d'ordre supérieur. Cette approche basée sur la similarité permet de choisir librement l'algorithme de classification final tout en bénéficiant des avantages, en termes de qualité des classes produites, de la co-classification.

Résume et Contributions de la Thèse

La contribution principale de la thèse peut-être résumée comme suit.

Dans le chapitre 2, nous avons décrit les concepts de base et fournit un résumé de l'état de l'art. Nous avons commencé par l'introduction de différents types de représentation des données utilisés dans la littérature ainsi que par la description du vocabulaire qui a été utilisé dans le reste de la thèse. Nous avons également discuté la limitation des mesures de similarité traditionnelles (tels que Cosinus, Tanimoto, etc) lorsqu'ils sont appliquées à des données dans des espace de haute dimension. Nous avons discuté les diverses approches possibles qui ont été proposées afin de faire face à cette limitation. Un aperçu des différentes familles d'algorithmes de co-classification a également été donné.

Dans le chapitre 3, nous avons proposé notre mesure de co-classification. Cette méthode tient compte de la structure du graphe biparti résultant d'une matrice documents-mots. Lorsque l'on calcule la similarité entre les documents, non seulement les mots contenus dans les deux documents sont examinés, mais nous considérons aussi des mots similaires qui peuvent être une conséquence d'un phénomène linguistique tels que la synonymie, le style propre d'un auteur et du vocabulaire. Pour prendre en compte ces mots similaires, nous avons proposé une mesure de similarité qui permet de calculer en parallèle deux matrices contenant respectivement les indices de similarité entre documents et entre termes, chacune de ces matrices étant construite à l'aide de la seconde. En conséquence, chacun de nos matrices R de similarité document-document, et C de similarité mot-mot, contient des informations provenant de l'autre matrice.

Nous avons étudié l'algorithme à partir d'un point de vue théorique basée sur le concept de chemins dans un graphe bipartite d'ordre supérieur. Ainsi nous avons pu observer que le calcul itératif des deux matrices de similarité lorsqu'il prend en compte les chemins redondants ne contribuent pas à apporter de nouvelles informations entre les paires de documents ou de. Nous avons donc proposé un moyen de calculer seulement les chemins élémentaires (c-à-d non-redondant) jusqu'à l'ordre-3. Nous avons en outre introduit un paramètre, appelé le pruning, comme une mesure visant à réduire la similarité résultant de co-occurrences fortuites.

L'une des caractéristiques inhérentes de notre méthode est que nous pouvons facilement intégrer les connaissances externes (telles que les étiquettes de catégorie de documents) à partir de données d'apprentissage afin d' influencer l'évolution des mesures de similarité. Nous avons présenté deux façons pour profiter de ces information afin de renforcer les liens au sein d'une classes et, au contraire, pour diminuer les valeurs de similarité entre les éléments (objets ou des caractéristiques) qui appartiennent à des catégories différentes. Grâce à ces propositions, nous avons pu étudier notre mesure de similarité dans le contexte du classement supervisé des document.

Dans le chapitre 4, nous fournissons les résultats expérimentaux pour illustrer et mesurer les avantages de notre mesure de similarité. Pour évaluer le résultat d'un clusteringon compare généralement l'adéquation entre les classes produites par l'algorithmes avec un ensemble de classes, supposées parfaites, qui ont été construites manuellement. Les différents paramètres de la mesure proposée ont été explorés et nous avons pu montrer que χ-Sim est sensiblement plus performant que les autres algorithmes de co-clustering sur les ensembles de données Newsgroup et SMART. De même, nous avons comparé la version supervisée de notre mesure avec plusieurs méthodes sur les données Newsgroup, Reuters et Lingspam.

Dans le chapitre 5, nous avons appliqué notre mesure pour accomplir la tâche de biclustering sur des données génomiques. Par apport aux matrices de documents / termes, qui sont très creuses, les matrices de données d'expression génique sont le plus souvent pleines. En outre, les erreurs expérimentales tendent à rendre ces données plus sensibles au bruit et des algorithmes efficaces doit être capable d'extraire les biclusters en présence d'un tel bruit. Nous avons effectué la bi-classification des gènes et des conditions simultanément et nous avons proposé un moyen d'extraire automatiquement les k biclusters les plus significatif en sélectionnant ceux qui montre la plus grande homogénéité au sein du bicluster et la plus grandes variations lorsqu'ils sont comparés à d'autres. Nous avons observé que le χ-Sim réussite a retrouver les biclusters initiaux, même en présence de bruit. Les résultats du

biclustering sur des données réelles du cancer ont également été utilisées d'un point de vue des processus biologiques.

Limites de la Mesure de Similarité Proposée

Nous avons pu montrer que, sur les jeux de données testés, notre mesure de co-classification permet de construire des classes pertinentes. Cependant, il faut mentionner quelques limites à l'approche.

- L'algorithme est assez coûteux en temps et l'espace lorsque les données sont grande. Une complexité de $O(\max(m^2n,mn^2))$ est généralement trop élevée pour que l'on puisse penser l'appliquer directement sur des ensembles de données tels que des corpus de grande taille contenant plusieurs millions de documents et de mots. De même, les matrices de similarité correspondantes ne peuvent pas tenir dans la mémoire vivre d'un ordinateur. Dans le chapitre 3 nous avons établi une relation entre χ-Sim et l'algorithme SimRank, et nous avons montré que χ-Sim pouvait être considérée comme une généralisation cet algorithme. À cet égard, on pourrait donc envisager diverses améliorations, notamment celles proposées par (P. Li et al 2010; Jia et al 2010.) qui ont été proposées pour améliorer l'efficacité de l'algorithme SimRank.
- Notre mesure de similarité est limitée aux attributs de type homogènes en utilisant la définition de la fonction de similarité fs (,) tel que décrit dans sous-section 3.3.3. On peut effectuer un recodage des données avant le calcul des valeurs de similarité, comme nous l'avons fait dans le chapitre 5 (section 5.3.4) afin de rendre les caractéristiques comparables entres elles. Dans ce cas il s'agissait d'homogénéiser les différents niveaux d'intensité obtenues dans les diverses conditions expérimentales. Toutefois, la définition actuelle de l'algorithme reste inapplicable sur des ensembles de données contenant des caractéristiques hétérogènes par exemple le niveau socio-culturel, revenu, etc dans la mesure évidemment ou le calcul de similarité reste sémantiquement fondé.
- Le calcul des valeurs de similarité tel qu'il est effectué actuellement (section 3.3.6) exige que toutes les valeurs de similarité soient recalculées lorsqu'on ajoute de nouveaux documents (objets) ou des mots (attributs). C'est une opération et il serait préférable de faire des mises à jour incrémentales lorsque c'est réalisable. Une piste possible serait d'utiliser des vecteurs « représentant » de chaque ligne et colonne de cluster (dans une idée similaire à la méthode proposée dans la section NC, (section 3.7.5). Les matrices de similarité entre ligne et entre colonne pourraient alors être calculées sur la base de ces vecteurs et les valeurs de similarité entre documents (ou mots), pourraient alors être exprimée comme une valeur (ou une probabilité) d'appartenance avec ces différents représentants. On aurait ainsi une représentation beaucoup plus compact et facile à mettre à jour.

Perspectives

Les travaux décrits dans cette thèse sont susceptibles de fournir des pistes de recherche intéressantes et diverses. De manière non exhaustive, on peut proposer les éléments suivants:

- La mesure de co-similarité produit deux matrices de proximité **R** et **C**. Ces matrices ont été utilisées pour le et les problèmes de catégorisation de documents. Il serait intéressant d'étudier plus formellement les propriétés de ces matrices, par exemple, pour voir si l'on pourrait les rendre définies positives. L'étude de ces propriétés pourrait de surcroit améliorer la méthode et permettre de l'utiliser comme une fonction noyau pour faire de la classification basée en utilisant un Séparateur à Vaste Marge (SVM).
- Dans cette thèse, la normalisation L1 a été utilisée et a montré de bons résultats dans le clustering et le classementde documents. En outre, une technique a été proposée permettant calculer les chemins non-redondants d'ordre-1, 2 et 3 au chapitre 3 (section 3.5). Toutefois, les valeurs générées pour chaque longueur des chemins ne sont pas directement comparables puisque les valeurs correspondant à une longueur croissante de chemins sont également croissantes, sans que le facteur de normalisation ne change. Par conséquent, la combinaison de ces valeurs de similarité aurit besoin d'utiliser une combinaison pondérée. Une solution possible pourrait être que, pour chaque valeur de $\mathbf{R}^{(1)}$, $\mathbf{R}^{(2)}$ et $\mathbf{R}^{(3)}$, l'étape de normalisation prennent en compte le nombre de chemins de cette longueur généré par chaque nœud. Si nous pouvons construire des matrices de similarité comparable, une simple combinaison linéaire des matrices devrait alors suffire à trouver la mesure de similarité globale sans prendre en compte les chemins redondants entre les éléments.
- La Recherche d'Information (RI) est une application très proche de classification et categorisation de documents. En RI, un utilisateur donne une requête pour récupérer une liste triée de documents qui sont semblables au mots de la requête. En utilisant une mesure de similarité entre mot, nous pourrions améliorer ce processus en incluant (c'est classiquement ce que l'on appelle faire de « l'expansion de requête ») non seulement les mots de la requête, mais aussi des mots qui peuvent être considérés comme similaires du point de vue de la matrice de similarité. La tâche est similaire à l'exécution d'un approache *k*-NN tel que décrite dans la section 3.7.5. On peut aussi choisir de regrouper les données et utiliser les NC afin de sélectionner les groupes concernés puis utiliser le *k*-NN pour sélectionner une liste de documents correspondant à la requête donnée et la trier sur la base de leurs valeurs de similarité.
- Enfin, comme les algorithmes de co-clustering font la classification à la fois des objets et des attributs, une application de notre measure de co-similarité pourrait être la sélection d' attributs.. On pourrait, par exemple, effectuer un regroupement initial des attributs et sélectionnez ceux qui ont les plus représentatifs dans chaque cluster. Cette notion de représentativité serait fonction du degré de connexions (directe ou d'ordre supérieur) avec les autres attributs qui couvrent la même catégorie de documents. Cette étape pourrait évidemment être couplée en amont avec des approches plus classiques comme la sélection basée sur l'information mutuelle afin de diminuer le nombre d'attributs candidats et ainsi accélérer le processus.

Publication (en française)

Co-classification : méthode et validation[29]

Gilles Bisson, Fawad Hussain

Laboratoire TIMC-IMAG, CNRS / UJF 5525
Université de Grenoble - Domaine de la Merci
38710 La Tronche, France
{gilles.bisson,fawad.hussain}@imag.fr

Résumé. La classification simultanée des *individus* et des *variables* qui composent un jeu de données, encore appelée *co-classification*, a été largement étudiée c'est dernières années. A l'inverse de la plupart des méthodes de co-classification existantes qui se focalisent sur l'obtention directe de classes d'individus et de variables, nous proposons ici *une mesure de co-similarité*, appelée χ-Sim, qui vise à construire simultanément des indices de similarité entre ces éléments. De la sorte, l'utilisateur a tout loisir de mettre ultérieurement en œuvre l'algorithme de classification (k-means, CAH, classification par densité, …) le mieux adapté à ses besoins. Les expérimentations effectuées sur la classification de corpus montrent que notre approche permet de construire des classes de qualité significativement supérieure à celles habituellement produites.

Mots-clés : Co-classification, Co-similarité, χ-Sim, Recherche d'information.

1 Introduction

La classification vise à organiser un ensemble de données issues d'une collection d'observations. Lorsque ces données sont « non-structurées », on peut les représenter sous la forme d'une table dans laquelle les lignes correspondent aux *individus* et les colonnes aux *variables* décrivant ces individus. Or, dans de nombreux problèmes, les variables sont suffisamment homogènes pour que l'on puisse envisager de les catégoriser au même titre que les individus. Ainsi, en *Recherche d'Information* (RI), dans le *modèle vectoriel* popularisé par Salton (1970), les corpus s'expriment comme une matrice dont les lignes représentent les « *vecteurs documents* » et les colonnes les « *vecteurs termes* » ; les valeurs indiquant la fréquence des termes dans les documents. Dans ce contexte, la similarité entre un couple de documents est fonction du degré de similarité entre les termes qui les composent et réciproquement deux termes (ou mots) auront « possiblement » des sens proches s'ils apparaissent dans des documents similaires. Cette double dépendance n'est pas toujours exploitée en RI, la ressemblance entre deux documents n'étant souvent fonction que du nombre de termes communs. La co-classification permet donc de prendre en compte et de tirer parti des informations qu'apportent cette dualité entre termes et documents.

[29] dans ***11ème Conférence Francophone sur l'apprentissage automatique (CAp 2009)***, *Plate-forme AFIA, Hammamet, Tunisie, 26-29 Mai, 2009. Editions Cépaduès*

Si la notion de co-classification, encore appelée bi-classification, a été introduite il y presque quarante ans par Hartigan (1972) puis par Govaert (1984), ce n'est que plus récemment que ce domaine de recherche a été intensivement exploré à la fois, comme nous venons de l'évoquer, dans le domaine de la recherche documentaire (citons entre autres les travaux de : Dhillon, 2001 ; Liu *et al.,* 2002 ; Dhillon *et al.,* 2003 ; Long *et al.,* 2005 ; Nadif & Govaert, 2005 ; Manjeet *et al.*, 2008), mais aussi dans le domaine de la bioinformatique afin, notamment, d'analyser les données d'expression génique (citons entre autres les travaux de : Cheng & Church, 2000 ; Madeira & Oliviera, 2004 ; Speer *et al.*, 2004)). Dans ces deux disciplines, en effet, les données se présentent sous la forme de matrices de co-occurrences caractérisant respectivement les relations entre termes/documents et entre gènes/expressions. Le présent travail se focalise sur la classification de documents même si notre approche reste, a priori, utilisable pour travailler sur des données biologiques.

Dans le domaine de la recherche d'information, la co-classification apporte un double bénéfice. D'une part, comme le souligne Long *et al.*, (2005), elle améliore la qualité des classes lorsqu'on travaille avec des matrices creuses et de grandes dimensions. D'autres part, cette approche permet de mettre en évidence des ressemblances entre des documents ne partageant, a priori, aucun terme comme l'illustre l'exemple de la table 1. Ces ressemblances reposent sur l'hypothèse que deux documents discutant d'une même thématique, et donc appartenant à la même catégorie, peuvent certes contenir des termes différents, mais que ces termes doivent être a priori fréquents dans la collection des documents relatifs à cette thématique.

Alors que la plupart des auteurs proposent des approches directes pour construire conjointement les classes de termes et de documents, dans cet article nous nous focaliserons sur la notion de *co-similarité*. Ainsi, nous avons défini une mesure, nommée χ-Sim, qui permet de calculer en parallèle deux matrices contenant respectivement les indices de similarité entre les documents et entre les termes, chacune de ces matrices étant construite à l'aide de la seconde. Après cette étape, l'utilisateur est libre d'utiliser les matrices avec l'algorithme de classification qui lui convient le mieux (K-means, Classification ascendante, Classification par densité, …) pour élaborer les classes de termes et/ou de documents.

Table 1 – Dans cet exemple « jouet », les documents d_1 et d_2 n'ont aucun terme en commun ; dès lors, avec une distance de Minkowski ou une mesure du Cosinus, leur similarité serait nulle (réciproquement, la distance serait maximale). Cependant, comme d_1 et d_2 partagent des termes avec d_3, les termes t^2 et t^3 ont donc une certaine ressemblance dans *l'espace des documents*. La co-classification permet ainsi de mettre en évidence une ressemblance entre d_1 et d_2 qui, bien que plus faible que celle existant entre d_3 et d_1 ou d_2 n'est pas nulle.

M	t^1	t^2	t^3	t^4
d_1	1	1	0	0
d_2	0	0	1	1
d_3	0	1	1	0

L'article est organisé ainsi : dans la deuxième partie, nous expliquons les principes de notre mesure de similarité χ-Sim ; dans la troisième partie, nous décrivons l'algorithme de calcul ; dans la quatrième partie, nous comparons notre mesure avec des méthodes courantes de (co)-classification ainsi qu'avec l'approche LSA (Latent Semantic Analysis) qui lui est conceptuellement proche.

2 Principe de la mesure χ-Sim

2.1 Notations utilisées

De façon classique, les matrices sont notées en caractères majuscules gras, les vecteurs en minuscules gras et les autres variables en minuscules italiques.

Matrice de données : soit **M** la matrice contenant les informations du corpus. Elle comporte *r* lignes (documents) et *c* colonnes (termes) ; chaque valeur m_{ij} caractérise *la nature de la relation*[30] entre le $j^{ème}$ terme et le $i^{ème}$ document ;

[30] Pour un corpus donné, m_{ij} correspond souvent au nombre d'occurrences (ou au tf-idf) du $j^{ème}$ terme dans le $i^{ème}$ document. En génomique cette valeur est plutôt un entier relatif indiquant le sens de variation de l'expression d'un gène par rapport à la valeur de base du métabolisme.

$\mathbf{m_i} = [m_{i1} \; .. \; m_{ic}]$ est le vecteur ligne correspondant au document i et $\mathbf{m^j} = [m_{1j} \; .. \; m_{rj}]$ est le vecteur colonne correspondant au terme j.

Matrices de similarité : **SR** (de taille *rxr*) et **SC** (de taille *cxc*) représentent les matrices carrées et symétriques contenant respectivement les valeurs de similarité entre documents et entre termes avec $sr_{ij} \in [0,1]$, $1 \leq i,j \leq r$ et $sc_{ij} \in [0,1]$, $1 \leq i,j \leq c$. Le vecteur $\mathbf{SC_i} = [sc_{i1} \; ... \; sc_{ic}]$ (respectivement $\mathbf{SR_i} = [sr_{j1} \; ... \; sr_{jc}]$) indique la similarité entre le terme i et l'ensemble des autres termes (respectivement entre le document j et l'ensemble des autres documents).

Fonction de similarité : la fonction générique $F_s(.,.)$ prend en entrée deux valeurs m_{ij} et m_{kl} de **M** et retourne la similarité entre ces valeurs avec $F_s(m_{ij}, m_{kl}) \in [0,1]$; en outre, on pose l'hypothèse que $F_s(m_{ij}, m_{kl}) = 0$ lorsque $m_{ij} = 0$ ou $m_{kl} = 0$. Dans la suite, nous utiliserons également la notation $\vec{F}(m_{ij,} \; \mathbf{m_k})$ (respectivement $\vec{F}(m_{ij,} \; \mathbf{m^k})$) pour caractériser la fonction qui calcule le vecteur des similarités entre la valeur $m_{i,j}$ et chacun des éléments du vecteur ligne $\mathbf{m_k}$ (respectivement du vecteur colonne $\mathbf{m^k}$).

2.2 Calcul de la co-similarité

Tout d'abord, nous allons décrire comment est effectué le calcul de la co-similarité pour la matrice **SR** entre les documents. Dans la littérature, la similarité (ou la distance) entre deux documents $\mathbf{m_i}$ et $\mathbf{m_j}$ est calculée par une fonction, que nous noterons ici $Sim(\mathbf{m_i}, \mathbf{m_j})$, qui prend en compte le nombre des termes partagés.

$$Sim(\mathbf{m_i}, \mathbf{m_j}) = F_s(m_{i1},m_{j1}) + F_s(m_{i2},m_{j2}) \; \ldots + F_s(m_{ic},m_{jc}) \qquad (1)$$

Le calcul est souvent plus complexe : ainsi, dans le cas de la distance Euclidienne, $Sim(\mathbf{m_i}, \mathbf{m_j})$ est la racine de la somme des carrés des $F_s(m_{ik},m_{jk})$, mais fondamenta-lement, on ne considère que des éléments ayant le même indice de colonne dans **M** : le terme k du document $\mathbf{m_i}$ n'est comparé qu'avec le terme k du document $\mathbf{m_j}$.. Supposons maintenant que nous disposions d'une matrice **SC** dont les valeurs caractérisent les similarités existantes entre les termes du corpus. Lorsque $sc_{i,i}=1$, l'équation (1) peut-être réécrite de la manière suivante sans en modifier de sens :

$$Sim(\mathbf{m_i}, \mathbf{m_j}) = F_s(m_{i1},m_{j1}).sc_{11} + F_s(m_{i2},m_{j2}).sc_{22} + \ldots + F_s(m_{ic,},m_{jc}).sc_{cc} \qquad (2)$$

Notre idée est maintenant de généraliser l'équation (2) de manière à prendre en compte dans ce calcul *l'ensemble de toutes les paires de termes* qui apparaissent pour les documents $\mathbf{m_i}$ et $\mathbf{m_j}$. De cette manière, nous allons « capturer » non seulement les similarités des termes partagés entre $\mathbf{m_i}$ et $\mathbf{m_j}$, mais également les similarités des termes *qui ne sont pas simultanément présents* dans les deux documents. Bien évidemment, la similarité entre une paire quelconques de valeurs m_{ik} et m_{jl} sera pondérée par la ressemblance sc_{kl} (issue de la matrice **SC**) entre les termes k et l correspondants. Le calcul de la similarité ainsi défini entre deux documents $\mathbf{m_i}$ et $\mathbf{m_j}$ est donné par l'équation (3) dans laquelle les éléments entourés désignent les éléments qui étaient déjà présents dans l'équation (2) :

$$\begin{aligned} Sim(\mathbf{m_i}, \mathbf{m_j}) = \; & \boxed{F_s(m_{i1},m_{j1}).sc_{11}} + F_s(m_{i1},m_{j2}).sc_{12} + \ldots + F_s(m_{i1},m_{jc}).sc_{1c} + \\ & F_s(m_{i2},m_{j1}).sc_{21} + \boxed{F_s(m_{i2},m_{j2}).sc_{22}} + \ldots + F_s(m_{i2},m_{jc}).sc_{2c} + \\ & \bullet\bullet\bullet \\ & F_s(m_{ic},m_{j1}).sc_{c1} + F_s(m_{ic},m_{j2}).sc_{c2} + \ldots + \boxed{F_s(m_{ic},m_{jc}).sc_{cc}} \end{aligned} \qquad (3)$$

En utilisant la notion plus concise introduite dans la partie 2.1 pour représenter les vecteurs de similarité entre documents, on peut réécrire l'équation (3) de la manière suivante. Le signe «x» indiquant le produit scalaire :

$$Sim(\mathbf{m_i}, \mathbf{m_j}) = \vec{F}(m_{i1},\mathbf{m_j}) \text{x} \mathbf{sc_1} + \vec{F}(m_{i2},\mathbf{m_j}) \text{x} \mathbf{sc_2} + \ldots + \vec{F}(m_{ic},\mathbf{m_j}) \text{x} \mathbf{sc_c} \qquad (4)$$

De manière symétrique, pour la matrice **SR** on peut exprimer la similarité entre deux termes (ou variables) $\mathbf{m^i}$ et $\mathbf{m^j}$ de **M** en utilisant la même approche. Ainsi :

$$Sim(\mathbf{m^i}, \mathbf{m^j}) = \vec{F}(m_{1i}, \mathbf{m^j}) \text{x} \mathbf{sr_1} + \vec{F}(m_{2i}, \mathbf{m^j}) \text{x} \mathbf{sr_2} + \ldots + \vec{F}(m_{ri}, \mathbf{m^j}) \text{x} \mathbf{sr_r} \qquad (5)$$

De la sorte, on voit que la similarité entre une paire quelconque de documents dépend de la similarité entre *tous les termes* qui apparaissent dans ces documents (pondérée par les éléments de la matrice **SC**) et que, de manière duale, la similarité de deux termes dépend de la similarité de tous les documents dans lesquels ils apparaissent (avec les pondérations de **SR**).

Toutefois, les valeurs calculées dans les équations (4) et (5) ne peuvent pas être directement utilisées pour affecter la valeur des éléments sr_{ij} de **SR** et sc_{ij} de **SC**. En effet, comme nous l'avons indiqué les éléments de **SR** et **SC** doivent appartenir à l'intervalle [0,1], or ni $Sim(\mathbf{m_i}, \mathbf{m_j})$ ni $Sim(\mathbf{m^i}, \mathbf{m^j})$ ne vérifient cette propriété ce qui nécessite une étape de *normalisation* préalable de ces valeurs.

Comme par définition (cf 2.1) la valeur maximum renvoyée par la fonction $F_s(.,.)$ vaut 1 et que cette valeur est nulle si m_{ij}=0 ou m_{kl}=0, en regardant l'équation (3), on voit aisément que la valeur maximale de $(\mathbf{m_i}, \mathbf{m_j})$, avec $1 \leq i,j \leq r$, correspond au produit du nombre des éléments non nuls de $\mathbf{m_i}$ et $\mathbf{m_j}$ (soit en pratique le nombre de termes qui apparaissent dans les $i^{ème}$ et $j^{ème}$ documents). Nous noterons ce produit : $|\mathbf{m_i}| \,.\, |\mathbf{m_j}|$. Le raisonnement est exactement le même pour les éléments de **SC**. Dès lors, les valeurs sr_{ij} de **SR** et sc_{ij} de **SC** sont calculées ainsi :

$$\forall i,j \in 1..r,\ sr_{i,j} = \frac{Sim(\mathbf{m_i},\mathbf{m_j})}{|\mathbf{m_i}|.|\mathbf{m_j}|} \qquad \forall i,j \in 1..c,\ sc_{i,j} = \frac{Sim(\mathbf{m^i},\mathbf{m^j})}{|\mathbf{m^i}|.|\mathbf{m^j}|} \qquad (6)\ \&\ (7)$$

Soulignons que cette normalisation est pertinente dans le cas de corpus car elle ne fait que prendre en compte la longueur des documents et des vecteurs de termes. Comme les équations décrites dans (6) et (7) ne sont pas indépendantes, pour calculer le contenu des matrices **SR** et **SC** nous allons utiliser une approche itérative (méthode de Jacobi) pour évaluer successivement (6) et (7) et mettre à jour les valeurs.

3 Algorithme de calcul de χ-Sim

La complexité algorithmique pour calculer les valeurs d'une matrice de similarité (ou de distance) entre N documents est généralement de l'ordre de $O(N^3)$ en terme du nombres d'appels à la fonction de comparaison des occurrences $F_s(.,.)$. C'est le cas pour la similarité du cosinus ou la distance de Minkowski. Toutefois, dans le cadre du calcul de **SR** et **SC**, tous les couples de documents et *tous les couples de termes* doivent être comparés ce qui conduit, a priori, à une complexité en $O(N^4)$ interdisant l'exploitation de corpus de taille « réelle » par notre approche. Il existe cependant une manière[31] de réduire la complexité de notre algorithme en $O(N^3)$.
Lorsque la fonction $F_s(m_{ij}, m_{kl})$ est définie comme *le produit* des deux valeurs m_{ij} *et* m_{kl} (comme dans la mesure du cosinus, par exemple), le système d'équations permettant de calculer **SR** et **SC** peut être reformulé, sans perte de généralité, comme le produit de 3 matrices. L'algorithme est le suivant :

1. On initialise les deux matrices **SR** (documents) et **SC** (termes) avec la matrice identité **I**. En effet, lors de la première itération la similarité entre un document (respectivement un terme) et lui-même est égal à 1 et cette valeur vaut 0 dans tous les autres cas car les indices des documents et des termes sont différents. On appellera ces deux matrices initiales $\mathbf{SC_0}$ et $\mathbf{SR_0}$.
2. A chaque itération *t* (initialement t=1), la matrice de similarité entre documents $\mathbf{SR_t}$ est calculée en utilisant la matrice de similarité entre termes $\mathbf{SC_{t-1}}$ qui a été construite à l'itération précédente (initialement $\mathbf{SC_0}$). Pour normaliser les nouvelles valeurs (comme on le faisait dans l'équation (6)), il suffit de faire un produit de Hadamard (notée ci-dessous par l'opérateur « • » avec pour tout $\mathbf{C}=\mathbf{A}\bullet\mathbf{B}$; $c_{i,j}=a_{i,j}.b_{i,j}$) entre la matrice $\mathbf{SR_t}$ et une matrice de normalisation **NR** dont les éléments sont définis ainsi : $\forall i,j \in [1,r]$, $nr_{i,j} = 1/|\mathbf{m_i}|.|\mathbf{m_j}|$. Le processus est le même pour calculer la matrice de similarité entre termes $\mathbf{SC_t}$. On a donc :

$$\mathbf{SR_t} = (\mathbf{M}.\ \mathbf{SC_{(t-1)}}.\ \mathbf{M^T}) \bullet \mathbf{NR} \quad avec\ nr_{i,j} = \frac{1}{|\mathbf{m_i}|.|\mathbf{m_j}|} \qquad (8)$$

$$\mathbf{SC_t} = (\mathbf{M^T}.\ \mathbf{M}\ .\ \mathbf{SR_{(t-1)}}) \bullet \mathbf{NC} \quad avec\ nc_{i,j} = \frac{1}{|\mathbf{m^i}|.|\mathbf{m^j}|} \qquad (9)$$

[31] Une seconde approche existe qui utilise le fait que lorsque les valeurs dans M sont *discrètes*, c'est-à-dire lorsqu'elles appartiennent à un type énuméré E contenant |E| éléments, une grande partie des calculs de similarités peut être préalablement évaluée et *mise en mémoire* réduisant ainsi la complexité à $O(|E|.N^3)$. Nous ne détaillons pas cette méthode par manque de place, mais elle présente l'avantage de ne faire aucune supposition sur la fonction $F_s(.,.)$.

3. On force les diagonales de $\mathbf{SR_t}$ et $\mathbf{SC_t}$ à la valeur 1 afin de contraindre la similarité entre un élément et lui-même à l'identité.
4. On passe à l'itération suivante (t=t+1) et l'on recommence à l'étape 2 du calcul des matrices $\mathbf{SR_t}$ et $\mathbf{SC_t}$. Cette itération est répétée *n* fois.

Il faut s'interroger sur le nombre d'itérations nécessaires pour calculer les valeurs finales de **SR** et **SC**. Dans l'algorithme de calcul du système d'équations (8) et (9), la notion d'itération a une signification forte : itérer *n* fois le calcul correspond à prendre en compte *le* $n^{ième}$ *degré de co-occurrences* entre les documents et les termes. Pour s'en convaincre il suffit de se représenter le corpus **M** sous la forme d'un graphe biparti formé d'un ensemble de documents et d'un ensemble de termes, graphe dans lequel chaque co-occurrence entre un document et un terme forme une arête. Lors de l'initialisation (t=0) on prend en compte les ressemblances directes (c'est-à-dire les termes ou les documents en communs), puis chaque itération (t=t+1) du calcul de $\mathbf{SR_t}$ et $\mathbf{SC_t}$ permet de prendre en compte des chemins de longueur croissante entre les documents et entre les termes. Or, comme le souligne Kontostathis & Pottenger (2006), il est généralement inutile de prendre en compte les co-occurrences entre documents et entre termes ayant un degré plus élevé que 3 ou 4. C'est la solution que nous adopterons ici et dans la cinquième partie de cet article, le nombre d'itérations sera fixé à 4 pour toutes les expérimentations effectuées avec χ-Sim.

Du point de vue de la complexité algorithmique, pour calculer $\mathbf{SR_t}$ (ou $\mathbf{SC_t}$) nous n'avons qu'à effectuer le produit de trois matrices. Comme la complexité de ces multiplications est dans le pire des cas en $O(N^3)$ et qu'à chaque itération il faut effectuer 4 produits, la complexité générale est en $O(k.N^3)$ avec la constante $k \approx 16$ lorsqu'on effectue 4 itérations pour la recherche de co-occurrences du $4^{ième}$ degré.

4 Expérimentations et résultats

L'évaluation de notre algorithme a été réalisée à partir d'une série d'expérimenta-tions sur des corpus dont les documents sont déjà catégorisés. Le critère d'évaluation consiste alors à *mesurer l'adéquation* entre les classes de documents engendrées par un algorithme de classification utilisant la matrice de similarité **SR** produite par χ-Sim et les classes existantes. Nous évaluons donc ici l'apport de la co-similarité dans la classification des documents comme cela est classiquement testé.

4.1 Bases expérimentales

Nous avons utilisé trois bases[32] documentaires classiques : les « Newsgroups » (notée NG20), la collection SMART et la collection TASA. Ces bases ont été sélectionnées car elles sont très souvent utilisées pour évaluer la classification et la co-classification de documents ; on peut citer entre autres : (Dhillon *et al.,* 2003 ; Long, *et al.,* 2005 ; Nadif & Govaert, 2005 ; Zhang *et al.,* 2007 ; Long *et al.,* 2006). Ainsi, il est plus simple de comparer les résultats obtenus. Les caractéristiques des bases d'expérimentations sont récapitulées dans la table 2.

La base NG20 contient 20 000 messages de longueurs très variables qui ont été échantillonnés de manière équi-répartie sur 20 groupes de conversations Usenet (soit 1000 messages par groupe). Les thèmes de discussion évoqués dans ces groupes ne sont pas disjoints et environ 4,5% des documents ont été postés dans au moins deux groupes. Afin de pouvoir comparer nos résultats avec ceux reportés par les différents auteurs, nous avons construit, par échantillonnage sur NG20, trois problèmes notés M2, M5 et M10 qui sont semblables en termes de nombres de documents et de thèmes sélectionnés, à ceux utilisés par Dhillon *et al.*, (2003) et Long *et al.*, (2005), ainsi que trois problèmes, NG1, NG2, NG3 semblables à ceux de Long *et al.*, (2006). Chacun de ces problèmes étant un sous-ensemble de la base NG20, nous avons pu construire 10 échantillons aléatoires ce qui a permis de calculer un écart-type sur les résultats et de vérifier la robustesse des approches. Ainsi, le problème M5 a été engendré dix fois en sélectionnant à chaque fois 100 documents parmi les 1000 des 5 groupes. Nous avons exclu des échantillons les messages ayant moins de 2 termes.

La collection SMART contient un ensemble de : 1033 résumés issus de la base médicale MEDLINE, 1460 résumés de la base CISI sur la recherche d'informations et enfin 1400 résumés de la base CRANFIELD concernant

[32] NG20 : http://people.csail.mit.edu/jrennie/20Newsgroups/, TASA est obtenue sur demande : http://ics.colorado.edu/ et enfin SMART : ftp://ftp.cs.cornell.edu/pub/smart

l'aérodynamique.

Table 2 – Ensemble des problèmes utilisés pour effectuer les expérimentations.

Problème	*Thèmes présents dans les documents*	*#classes*	*#docs/classes*	*#docs*
M2	talk.politics.mideast, talk.politics.misc	2	250	500
M5	comp.graphics, rec.motorcycles, sci.space, rec.sport.baseball, talk.politics.mideast.	5	100	500
M10	alt.atheism, sci.crypt, comp.sys.mac.hardware, misc.forsale, rec.autos, sci.electronics, sci.med, sci.space, talk.politics.gun, rec.sport.hockey.	10	50	500
NG1	rec.sports.baseball, rec.sports.hockey	2	200	400
NG2	comp.os.ms-windows.misc, comp.windows.x, rec.motorcycles, sci.crypt,sci.space.	5	200	1000
NG3	comp.windows.x, misc.forsale, rec.motorcycles, sci.space, talk.politics.mideast, sci.crypt, talk.religion.misc, comp.os.ms-windows.misc.	8	200	1600
SMART	MEDLINE, CRANFIELD, CISI	3	1033,1460,1400	3893
TASA5	LanguageArts, Health, Science, SocialStudies, Business.	5	500	2500

Enfin la collection TASA produite par la « Touchstone Applied Science Associates, Inc. » est constituée d'environ 45000 textes de taille assez homogène (250-300 termes) qui sont d'origines diverses : romans, journaux,…. A chaque texte est associé un degré de lisibilité (DRP-Degrees of Reading Power Scale) qui est entier caractérisant sa complexité en termes de vocabulaire et de concept. Pour l'expérimentation, nous avons sélectionné pour les 5 classes, des textes ayant une complexité homogène correspondant à des documents compréhensibles par des élèves de niveau collège et lycée (les textes trop simples ou trop spécifiques ont été exclus).

De façon similaire à Dillhon *et al.*, (2003) nous avons effectué les prétraitements suivants sur les corpus correspond aux 8 problèmes :

- Les lignes contenant de manière explicite les thèmes des articles ou des messages (titre, liste de mots-clef, …) ont été évidemment retirées.
- Les termes « vides » (*stop words*) ont été supprimés.
- Dans la représentation vectorielle des documents, nous n'avons conservé que les 2000 mots les plus informatifs du corpus en nous basant sur la mesure d'information mutuelle.

4.2 Algorithmes utilisés

Les huit ensembles de tests ont été utilisés pour comparer χ-Sim avec 5 autres approches qui sont séparables en deux familles : premièrement, les méthodes de co-classification reposant sur l'utilisation d'une mesure de (co-) similarité, comme c'est le cas pour χ-Sim, et deuxièmement les algorithmes effectuant directement une co-classification des documents et des termes.

Dans le cas des approches basées sur une mesure de (co-)similarité, il est évidemment nécessaire pour construire des classes de documents d'utiliser conjointement à cette mesure un algorithme de classification. Dans tous les cas nous avons retenu[33] l'algorithme de Classification Ascendante Hiérarchique (CAH) avec l'indice de Ward. Nous avons étudié les mesures de similarité suivantes :

- *La similarité du cosinus*. Bien que cette mesure ne soit aucunement liée à la notion de co-classification, nous l'avons retenue car elle est intensivement utilisée en recherche d'information. Elle permet donc de définir une « ligne de base » pour les résultats et de mieux estimer l'apport d'une analyse conjointe des

[33] D'autres approches ont également été testées : les k-means et la CAH avec d'autres indices d'agrégation. Toutefois, les résultats obtenus avec ces approches ont été systématiquement moins bons pour l'ensemble des ensemble de tests et l'ensemble des mesures de similarité.

documents et des termes, même si, comme dans cette expérimentation, on ne s'intéresse qu'à la seule classification des documents.

- *La mesure de similarité induite par LSA* (Latent Semantic Analysis) (Deerwester *et al.,* 1990 ; Dumais, 2007) qui est très souvent utilisée en recherche d'information, en TAL et en sciences cognitives. Dans leur analyse Kontostathis & Pottenger (2006) montrent que LSA permet d'établir des ressemblances entre des documents (respectivement des termes) ne partageant pas explicitement de termes (respectivement des termes) ; il est donc pertinent de comparer une co-classification basée sur cette approche avec χ-Sim d'autant plus que les mécanismes sous-jacents sont différents : LSA est basée sur la décomposition en valeurs singulières de la matrice **M**.

En ce qui concerne les algorithmes de co-classification proprement dit, nous avons retenu les trois approches suivantes qui cherchent à réordonner les lignes et les colonnes de **M** de façon à mettre en évidence des sous-matrices dans lesquelles les documents et les termes sont corrélés :

- ITCC (Information Theoretic Co-Clustering) de Dhillon *et al.*, (2003).
- BVD (Block Value Decomposition) de Long *et al.*, (2005).
- RSN (k-partite graph partitioning algorithm) de Long *et al.*, (2006).

4.3 Implémentation et critère d'évaluation

Pour effectuer les tests, χ-Sim a été implémenté en C++, et nous avons utilisé MatLab pour calculer la similarité du Cosinus et implémenter LSA. Les classifications ascendantes hiérarchiques ont été réalisées à l'aide de MathLab. Le code utilisé pour ITCC est celui fourni par ses auteurs. Pour les BVD et RSN nous n'avions pas d'implémentations disponibles, aussi avons-nous reporté directement les *meilleurs résultats publiés* dans (Long *et al.*, 2005 ; Long *et al.,* 2006) qui utilisent également des échantillons du corpus NG20 (cf. Table 2 et 3).
De nombreuses mesures ont été proposées pour quantifier la ressemblance entre les classes initiales et les classes produites par un algorithme de classification. Elles sont souvent basées sur l'analyse de la matrice de confusion **C** dans laquelle chaque élément $c_{i,j}$ indique le nombre de documents de la classe réelle *i* qui ont été affectés à la classes apprise *j* (idéalement on devrait obtenir une matrice diagonale). Pour évaluer les résultats et permettre la comparaison avec les autres travaux nous avons utilisé deux indices classiques :

- La « précision micro-moyennée » ou micro-averaged Precision (Pr)
- L'Information Mutuelle Normalisée (NMI) utilisée par Banerjee *et al.,* (2002)

4.4 Résultats expérimentaux

Pour tous les algorithmes disponibles, les problèmes issus de NG20 et TASA5 ont été testés sur 10 échantillons aléatoires (Cf. 4.1) ce qui nous a permis d'obtenir la valeur moyenne et l'écart type des indices Pr et NMI. Pour SMART le test a porté sur l'intégralité de la base, il n'y a donc pas d'écart type. Dans le cas où les algorithmes étaient disponibles, le nombre de classes de documents demandés (ou obtenu en tronquant l'arbre construit par la classification ascendante hiérarchique) était égal au nombre de classe attendues (2 dans le cas de M2, 3 pour SMART, etc). Par ailleurs, afin de déterminer les meilleures conditions expérimentales pour LSA et ITCC, nous avons effectués les expérimentations supplémentaires suivantes :

- Pour LSA, nous avons systématiquement effectué les calculs en représentant les valeurs de co-occurrences dans **M** soit comme une fréquence d'apparition des termes, soit en calculant le « tf-idf». En outre, nous avons fait varier le nombre *k* des valeurs singulières conservées dans l'intervalle [10..200] par pas de 10 afin de trouver le nombre optimal de dimensions. La valeur rapportée pour chaque problème est celle correspondant à la représentation des données et la valeur de *k* qui maximise la valeur moyenne de Pr.
- Pour ITCC, nous avons fait varier pour chaque ensemble de tests le nombre de classes de mots dans la plage de valeur suggérée par Dhillon *et al.,* 2003). Par ailleurs, comme ITCC repose sur une initialisation aléatoire des classes, chaque problème a été soumis 3 fois. Comme pour LSA les résultats rapportés correspondent aux paramètres qui maximisent la valeur moyenne de Pr.

Les résultats obtenus sont regroupés dans le tableau 3. La colonne « temps » indique pour chaque problème le temps moyen qui a été nécessaire à χ-Sim pour calculer les deux matrices **SR** et **SC** avec 4 itérations. Le calcul a été

effectué sur une machine dotée d'un processeur Intel Core2 à 2Ghz.
Dans cette série d'expériences, on constate que les classifications de documents basées sur la mesure de similarité de χ-Sim sont, sur l'ensemble des problèmes, significativement meilleures à toutes les autres approches. En utilisant LSA on obtient cependant aussi de bons scores en moyenne sauf dans M10 et TASA5. Le résultat le plus surprenant est celui obtenu avec ITCC puisque, dans un certain nombre de problèmes, son score est inférieur[34] à celui obtenu à l'aide de la mesure du cosinus.

Il est intéressant d'analyser l'évolution du comportement de χ-Sim, sur les bases M2 et M10, lorsque l'on garde constant le nombre total de documents (500) tout en augmentant le nombre de classes (cf. table 2). Comme on a simultanément à résoudre un problème de plus en plus complexe tout en ayant un nombre d'exemples d'apprentissage par classes qui diminue, on peut s'attendre à une dégradation des performances (indice Pr) des algorithmes. C'est bien ce que l'on constate, toutefois cette dégradation est beaucoup plus faible pour χ-Sim (-0,18) que pour LSA (-0,24), ITCC (-0,43) et BVD (-0,28) pour lequel on n'a toutefois que le meilleur résultat.
Enfin, on peut remarquer que χ-Sim a généralement un écart-type faible, à la fois pour les valeurs de Pr et de NMI, sur les 10 tirages aléatoires de chaque problème. La méthode semble donc posséder une bonne stabilité.

Table 3 – Résultats des expérimentations en termes de précision micro-moyennée (Pr) et d'information mutuelle normalisée (NMI) pour les tests issus de NG20, SMART et TASA5.

Problème	*Temps*	*Critère*	*χ-Sim*	*LSA*	*ITCC*	*Cosinus*	*BVD*	*RSN*
M2	*12s*	Pr_{max}	**0.96**	0.94	0.92	0.72	0.95	-
		Pr	**0.93 ± 0.01**	0.78 ± 0.15	0.71 ± 0.15	0.62 ± 0.04		
		NMI	**0.65 ± 0.05**	0.37 ± 0.23	0.21 ± 0.21	0.15 ± 0.05		
M5	*10.9s*	Pr_{max}	**0.98**	0.95	0.56	0.76	0.93	-
		Pr	**0.94 ± 0.04**	0.82 ± 0.09	0.48 ± 0.06	0.6 ± 0.08		
		NMI	**0.85 ± 0.07**	0.65 ± 0.14	0.27 ± 0.09	0.55 ± 0.11		
M10	*13.5s*	Pr_{max}	**0.80**	0.67	0.33	0.55	0.67	
		Pr	**0.75 ± 0.04**	0.54 ± 0.09	0.28 ± 0.04	0.45 ± 0.07		-
		NMI	**0.65 + 0.05**	0.45 ± 0.08	0.16 ± 0.04	0.43 ± 0.07		
NG1	*6.6s*	Pr_{max}	**1**	0.98	0.83	0.97		
		Pr	**1 ± 0**	0.95 ± 0.02	0.67 ± 0.12	0.90 ± 0.11		
		NMI	**1 ± 0**	0.75 ± 0.08	0.13 ± 0.13	0.62 ± 0.20	-	0.64 ± 0.16
NG2	*14.6s*	Pr_{max}	**0.94**	0.87	0.72	0.72		
		Pr	**0.93 ± 0.01**	0.82 ± 0.03	0.57 ± 0.08	0.60 ± 0.08		
		NMI	**0.81 ± 0.04**	0.64 ± 0.03	0.35 ± 0.09	0.51 ± 0.07	-	0.75 ± 0.07
NG3	*20.2s*	Pr_{max}	**0.91**	0.80	0.65	0.62		
		Pr	**0.85 ± 0.06**	0.72 ± 0.05	0.55 ± 0.06	0.60 ± 0.04		
		NMI	**0.77 + 0.02**	0.59 ± 0.03	0.48 ± 0.05	0.55 ± 0.03	-	0.70 ± 0.04
SMART	*164.4s*	Pr	**0.99**	0.98	0.97	0.89	-	-
		NMI	**0.96**	0.93	0.90	0.63		
TASA5	*95,7s*	Pr_{max}	**0.80**	0.51	0.31	0.26	-	-
		Pr	**0.74 ± 0.04**	0.50 ± 0.02	0.27 ± 0.025	0.25 ± 0.02		
		NMI	**0.59 ± 0.02**	0.40 ± 0.05	0.028 ± 0.018	0.14 ± 0.03		

[34] De manière générale, nous ne sommes pas parvenus à reproduire les résultats publiés dans (Dhillon et al. 2003). Il ne nous a pas été possible de déterminer si cela provenait de l'implémentation fournie pas les auteurs ou si, plus simplement, du fait que nous ne travaillons pas avec les mêmes données, l'échantillonnage dans NG20 étant aléatoire.

6. Conclusion

Dans cet article, nous avons introduit une nouvelle approche pour la co-classification qui est basée sur la définition de la mesure de co-similarité χ-Sim. En prenant en compte de façon itérative la similarité entre les termes dans le calcul de la similarité entre les documents et réciproquement des documents dans celle des termes, nous avons montré que cette méthode est capable de mettre automatiquement en relation des documents et des termes ne partagent pas explicitement des informations. Cette approche basée sur la similarité permet de choisir librement l'algorithme de classification final tout en bénéficiant des avantages, en terme de qualité des classes produites, de la co-classification. Elle ne nécessite en outre aucune ressource linguistique (thésaurus, ...) particulière.

Nous avons pu montrer expérimentalement que notre approche permettait d'obtenir de meilleurs résultats que les méthodes de co-classification récentes. De surcroît, contrairement aux approches proposées par (Dhillon *et al.,* 2003 ; Long *et al.*, 2005), il n'est pas nécessaire de classer les termes pour classer les documents ce qui évite à l'utilisateur de devoir fournir le nombre de classes de termes attendu (l'avantage est le même lorsqu'on ne s'intéresse qu'aux seules classes de termes).
Il est intéressant de noter que notre approche obtient, sur les bases testées, de meilleurs résultats que LSA qui est aussi une mesure largement utilisée en RI et qui permet aussi de mettre en évidence des co-occurrences d'ordre supérieur entre termes et documents. Toutefois, dans nos expérimentations, afin de pouvoir comparer nos résultats avec ceux de la littérature nous nous sommes limités à des jeux de données de très petites taille (2000 mots et moins de 4000 documents au maximum) alors que LSA est généralement mis en œuvre sur des corpus beaucoup plus importants.

Dans des travaux en cours, non détaillés dans cet article, nous avons pu observer que lorsque le nombre de documents augmentait, l'écart entre LSA et χ-Sim avait tendance à se réduire : lorsqu'on soumet le problème M10 sur l'ensemble des 20 000 documents issus de la collection NG20, la valeur de Pr est respectivement de 75% et 82% pour LSA et χ-Sim. On peut imaginer que cet écart diminue sur de plus larges collections. Notre approche semble donc être plus à même que LSA de capturer les co-occurrences d'ordre supérieur sur les corpus de petite et moyenne dimensions.

Références

BANERJEE A. & GHOSH J. (2002). Frequency sensitive competitive learning for clustering on high dimensional hyperspheres. IEEE joint conf. on neural networks. p. 1590-1595. Hawaii.

CHENG Y. & CHURCH G. M (2000). Bi-clustering of expression data. Proceedings International Conference on Intelligent System for Molecular Biology. p. 93-103. Boston.

DEERWESTER S.C., DUMAIS S.T., LANDAEUR T.K., FURNAS G.W. & HARSHMAN R.A. (1990). Indexing by Latent Semantic Analysis. Journal of the American Society of Information Science. Volume 41(6):391-407.

DHILLON I.S. (2001). Co-clustering documents and words using bipartite spectral graph partitioning. Proceedings of the Seventh ACM SIGKDD. San Francisco.

DHILLON I.S., MALLELA S. & MODHA D.S. (2003). Information Theoretic Co-clustering. Proceedings of The Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Aug 24-27. 2003. p. 89-98. Washington.

DUMAIS S.T. (2007). LSA and Information Retrieval: Getting back to Basics. Handbook of Latent Semantic Analysis. Lawrench Erlabaum Associates. ISBN: 978-0-8058-5418-3.

GOVAERT G. (1984). Classification simultanée de tableaux binaires. In E. Diday. M. Jambu. L. Lebart. J. Pages, and R. Tomassone editors. Data analysis and informatics III. p. 233-236.

HARTIGAN J.A. (1972). Direct clustering of a data matrix. Journal of American Statistical Association. Volume 67(337):123-129.

KONTOSTATHIS A. & POTTENGER W.M. (2006). A framework for understanding LSI performance. Information Processing and Management. Volume 42. number 1. p. 56-73.

LIU, X., GONG, Y., XU, W., & ZHU, S. (2002). Document clustering with cluster refinement and model selection capabilities. In Proceeding of ACM SIGIR Conference on Research and Development in information retrieval.

LONG B., ZHANG Z. & YU P.S. (2005). Co-clustering by Block Value Decomposition. Proceedings of the eleventh ACM SIGKDD. August 21-24. p. 635-640. Chicago.

LONG B., WU X., ZHANG Z. & YU P.S. (2006). Unsupervised Learning on K-partite Graphs, Proceedings of the 12th ACM SIGKDD. August 20-23. p. 317-326. Philadelphia.

MADEIRA S.C. & OLIVEIRA A.L. (2004). Biclustering algorithms for biological data analysis: a survey. IEEE/ACM Transactions on Computational Biology and Bioinformatics. 1(1):24-45.

MANJEET R., MING D. & FARSHAD. F. (2008). Bipartite isoperimetric graph partitioning for data co-clustering. Journal of Data Mining and Knowledge Discovery. Volume 16(3):276-312.

NADIF M. & GOVAERT G. (2005). Block Clustering of Contingency Table and Mixture Model. IDA. LNCS : Advances in Intelligent Data Analysis VI. p. 249-259. Madrid

SALTON G. (1971). The SMART Retrieval System – Experiments in Automatic Document Processing. Prentice Hall. Inc.. Englewood Cliffs. NJ.

SPEER N., SPIETH C. & ZELL A. (2004). A Memetic Clustering Algorithm for the Functional Partition of Genes Based on the Gene Ontology. Proceedings of the IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology. p. 252-259.

TISHBY N., PEREIRA F. & BIALEK W. (1999). THE INFORMATION BOTTLENECK METHOD. PROCEEDINGS OF THE 37TH ANNUAL CONFERENCE ON COMMUNICATION CONTROL AND COMPUTING. P. 368-377.

ZHANG D., ZHOU Z-H & SONGCAN C. (2007). SEMI-SUPERVISED DIMENSIONALITY REDUCTION. PROCEEDINGS OF THE SEVENTH SIAM INTERNATIONAL CONFERENCE ON DATA MINING. MINNEAPOLIS.

Appendix I

Lance and Williams (Lance and Williams 1967) have shown that there exists a general combinatorial equation that can be used to describe the update step used in different agglomerative hierarchical clustering methods to compute the similarity matrix after a fusion of any two objects. The equation is:

$$s_{hk} = \alpha_i s_{hi} + \alpha_j s_{jh} + \beta s_{ij} + \gamma \left| s_{hi} - s_{hj} \right|$$

In this equation, s_{ij} refers to the similarity between the documents *i* and *j* that are to be merged to create a new cluster *k*. The new similarity between this newly created cluster *k* and any document *h* is given by s_{hk}, and the parameters α_i, α_j, β, and γ are specified by the respective hierarchic agglomerative procedure. Table A1.1 below gives the values of these parameters for various clustering methods. For the formulas in this table, n_r corresponds to the number of documents contained in cluster *r*, where *r* = I, j, h.

Table A1-1 Values of the parameters corresponding to the Lance and Williams's equation for re-calculating similarity values.

	α_i	α_j	β	γ
Single Link	1/2	1/2	0	-1/2
Complete Link	1/2	1/2	0	1/2
Group Average	$\frac{n_i}{n_i + n_j}$	$\frac{n_j}{n_i + n_j}$	0	0
Ward's Method	$\frac{n_h + n_i}{n_h + n_i + n_j}$	$\frac{n_h + n_j}{n_h + n_i + n_j}$	$\frac{-n_j}{n_h + n_i + n_j}$	0

Using these updating parameters and the Lance and Williams's equation, it is possible to efficiently implement the various linkage-based agglomerative hierarchical clustering methods. One such implement was provided by (Wishart 1969).

Appendix II

We denote discrete random variables as before by capitals (X,Y,..) and their realizations (values taken by the random variable) by lower case ($x, y, \ldots$). The probability distribution function of X is denoted by $p(x) = p(X=x)$, and the conditional probability distribution as $p(x|y)= p(X=x|Y=y)$.

Entropy

Entropy is the measure of the uncertainty of a random variable. The entropy of a discrete random variable X, having a probability distribution $p(x)$ is given by

$$H(X) = -\sum_{x \in X} p(x) \log p(x)$$

Similarly, let $p(X,Y)$ denote the joint probability distribution between X and Y. We can consider $p(X,Y)$ is an $m \times n$ matrix, which can be estimated from our matrix **A** using a statistical estimate called a two-dimensional contingency table or a two way frequency table (R. A. Fisher 1922). Let $p(x)$ and $p(y)$ be the marginal distributions of X and Y, then the conditional entropy is given by

$$H(X \mid Y = y) = -\sum_{x \in X} \frac{p(x,y)}{p(y)} \log \frac{p(x,y)}{p(y)}$$

and

$$H(\mathrm{X}|\mathrm{Y}) = -\sum_{y \in Y} p(y) H(\mathrm{X} \mid \mathrm{Y} = y).$$

Entropy is a measure of uncertainty – the higher the entropy, the more uncertain one is about a random variable.

Mutual Information

Mutual information is one of many quantities that measures how much one random variable tells us about another. The mutual information between two random variables X and Y with a joint probability distribution $p(x,y)$ is given by

$$I(X;Y) = \sum_{x \in X} \sum_{y \in Y} p(x,y) \log_2 \frac{p(x,y)}{p(x)p(y)}$$

Mutual information can also be expressed in terms of entropy as

$$I(X;Y) = H(X) - H(X \mid Y) = H(Y) - H(Y \mid X)$$

The mutual information is a natural measure of the dependence between random variables. From the above definition, it follows that mutual information is the reduction of uncertainty of X after observing Y. Since we define X and Y and the documents and terms respectively, the mutual information between X and Y thus provides us with a reduction of uncertainly regarding a document when the term (word) is observed. It is this relationship that all information theoretic algorithms tend to exploit to group documents and words together as shall be seen in the rest of this section. It is always nonnegative, and zero if and only if the variables are statistically independent. We also define below some of the properties of mutual information that shall be used.

- I(X;Y)=I(Y;X)
- $I(X;Y) \geq 0$
- $I(X;Y) = H(X) + H(Y) - H(X,Y)$
- Mutual information can be expressed in terms of the Kullback Leibler distance D_{KL}, as

  $$I(X;Y) = D_{KL}(p(X,Y) \,||\, p(X)p(Y))$$

Appendix III

Proof of Convergence of Similarity Matrices

Here we show that the matrices $\mathbf{R}^{(t)}$ and $\mathbf{C}^{(t)}$ converges when $A_{ij} \geq 0 \ \forall \ i \in 1..m, j \in 1..n$. In order to show the convergence, we need to satisfy the following two properties

- The matrices **R** and **C** are monotonically non-decreasing i.e. $\mathbf{R}^{(t+1)} \geq \mathbf{R}^{(t)}$ and $\mathbf{C}^{(t+1)} \geq \mathbf{C}^{(t)}$ for all t
- The matrices **R** and **C** are bounded

Proof:

1. *Matrices* **R** *and* **C** *are monotonically non-decreasing*

   Suppose that $\mathbf{C}^{(t+1)} \geq \mathbf{C}^{(t)}$ i.e. $\mathbf{C}^{(t+1)} - \mathbf{C}^{(t)} \geq 0$.

   To show that $\mathbf{R}^{(t+1)} \geq \mathbf{R}^{(t)}$, we need to show that $\mathbf{R}^{(t+1)} - \mathbf{R}^{(t)} \geq 0$. This is given by:

$$R_{ij}^{(t+1)} - \mathbf{R}_{ij}^{(t)} = \frac{1}{\sum_{k=1..n} A_{ik} \sum_{k=1..n} A_{jk}} \sum_{k=1}^{n} \sum_{l=1}^{n} A_{ik} \times A_{jl} \times C_{kl}^{(t)} - \frac{1}{\sum_{k=1..n} A_{ik} \sum_{k=1..n} A_{jk}} \sum_{k=1}^{n} \sum_{l=1}^{n} A_{ik} \times A_{jl} \times C_{kl}^{(t-1)}$$

$$= \frac{1}{\sum_{k=1..n} A_{ik} \sum_{k=1..n} A_{jk}} \sum_{k=1}^{n} \sum_{l=1}^{n} A_{ik} \times A_{jl} \times (C_{kl}^{(t)} - C_{kl}^{(t-1)})$$

   By our hypothesis $\mathbf{C}^{(t)} - \mathbf{C}^{(t-I)} \geq 0$. Moreover, $A_{ij} \geq 0$ implies that

(AIII.1) $$\frac{1}{\sum_{k=1..n} A_{ik} \sum_{k=1..n} A_{jk}} \sum_{k=1}^{n} \sum_{l=1}^{n} A_{ik} \times A_{jl} \geq 0 \, .$$

   Therefore, if $\mathbf{C}^{(t+1)} \geq \mathbf{C}^{(t)}$, it follows that $\mathbf{R}^{(t+1)} \geq \mathbf{R}^{(t)}$. Using a similar manner we can show that, if $\mathbf{R}^{(t+1)} \geq \mathbf{R}^{(t)} \Rightarrow \mathbf{C}^{(t+1)} \geq \mathbf{C}^{(t)}$. It is evident that $\mathbf{R}^{(1)} \geq \mathbf{R}^{(0)}$ since $\forall i = j, R_{ij}^{(1)} = R_{ij}^{(0)} = 1$ and $\forall i \neq j, R_{ij}^{(1)} \geq R_{ij}^{(0)}$ from (AIII.1) above given that $\mathbf{C}_{kl} \geq 0$.

   Therefore, we conclude that matrices $\mathbf{R}^{(t)}$ and $\mathbf{C}^{(t)}$ are monotonically non-decreasing.

2. Matrices **R** *and* **C** *are bounded*

Suppose that $0 \le C_{ij} \le 1$, we show that $R_{ij} \le 1$. R_{ij} is given by

(AIII.2) $$R_{ij} = \frac{1}{\sum_{k=1..n} A_{ik} \sum_{k=1..n} A_{jk}} \sum_{k=1}^{n} \sum_{l=1}^{n} A_{ik} \times A_{jl} \times C_{kl}$$

If $0 \le C_{kl} \le 1$, then

(AIII.3) $$R_{ij} \le \frac{1}{\sum_{k=1..n} A_{ik} \sum_{k=1..n} A_{jk}} \sum_{k=1}^{n} \sum_{l=1}^{n} A_{ik} \times A_{jl}$$

Since $\sum_{k=1..n} A_{ik} \sum_{k=1..n} A_{jk} = \sum_{k=1}^{n} \sum_{l=1}^{n} A_{ik} \times A_{jl}$, it follows that $R_{ij} \le 1$.

Since $\mathbf{C}^{(0)}=\mathbf{I}$, we satisfy our supposition $0 \le C_{ij} \le 1$. In a similar manner, if $0 \le R_{ij} \le 1$ it implies that $C_{ij} \le 1$. Since $\mathbf{R}^{(0)}=\mathbf{I}$, therefore we have $C_{ij} \le 1$.

We have shown that if $A_{ij}>0$, then matrices $\mathbf{R}^{(t)}$ and $\mathbf{C}^{(t)}$ are non-decreasing and bounded, therefore they will converge, which completes our proof.

Appendix IV

This appendix provides additional results to those provided in Chapter 4.

Table AIV.1 Values corresponding to FIGURE 4.2 (a)

			Iterations									
Overlap			1,00	2,00	3,00	4,00	5,00	6,00	7,00	8,00	9,00	10,00
	30,00	avg	0,35	0,35	0,36	0,36	0,36	0,36	0,36	0,36	0,36	0,36
		Std	± 0,01	± 0,01	± 0,01	± 0,01	± 0,02	± 0,01	± 0,02	± 0,01	± 0,02	± 0,02
	40,00	avg	0,35	0,38	0,37	0,38	0,38	0,38	0,38	0,38	0,38	0,38
		Std	± 0,01	± 0,02	± 0,02	± 0,02	± 0,01	± 0,02	± 0,01	± 0,01	± 0,01	± 0,02
	50,00	avg	0,37	0,38	0,44	0,45	0,46	0,45	0,45	0,44	0,44	0,44
		Std	± 0,02	± 0,01	± 0,03	± 0,03	± 0,02	± 0,01	± 0,01	± 0,01	± 0,01	± 0,02
	60,00	avg	0,36	0,55	0,58	0,59	0,63	0,63	0,62	0,61	0,61	0,61
		Std	± 0,04	± 0,03	± 0,03	± 0,02	± 0,04	± 0,03	± 0,02	± 0,02	± 0,03	± 0,02
	70,00	avg	0,56	0,81	0,83	0,85	0,86	0,85	0,85	0,85	0,85	0,84
		Std	± 0,03	± 0,04	± 0,02	± 0,03	± 0,04	± 0,02	± 0,03	± 0,04	± 0,03	± 0,03
	80,00	avg	0,85	0,91	0,92	0,93	0,94	0,94	0,94	0,94	0,94	0,94
		Std	± 0,03	± 0,02	± 0,03	± 0,03	± 0,03	± 0,02	± 0,02	± 0,03	± 0,03	± 0,03
	90,00	avg	0,96	0,97	0,98	0,98	0,98	0,98	0,98	0,98	0,98	0,98
		Std	± 0,01	± 0,01	± 0,01	± 0,01	± 0,01	± 0,01	± 0,01	± 0,01	± 0,01	± 0,01
	100,00	avg	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99
		Std	± 0,00	± 0,00	± 0,00	± 0,00	± 0,00	± 0,00	± 0,00	± 0,00	± 0,00	± 0,00

Table AIV.2 Values corresponding to

FIGURE 4.2 (b)

datasets		iterations									
		1,00	2,00	3,00	4,00	5,00	6,00	7,00	8,00	9,00	10,00
	m2	0,59 ± 0,05	0,88 ± 0,11	0,89 ± 0,10	0,93 ± 0,07	0,91 ± 0, 07	0,91 ± 0,01	0,91 ± 0,07	0,91 ± 0,07	0,91 ± 0,07	0,91 ± 0,08
	m5	0,57 ± 0,15	0,90 ± 0,09	0,92 ± 0,05	0,94 ± 0,04	0,94 ± 0,04	0,94 ± 0,04	0,94 ± 0,04	0,93 ± 0,04	0,94 ± 0,04	0,94 ± 0,04
	m10	0,32 ± 0,05	0,59 ± 0,10	0,63 ± 0,08	0,63 ± 0,10	0,63 ± 0,11	0,63 ± 0,13	0,64 ± 0,10	0,64 ± 0,10	0,63 ± 0,12	0,64 ± 0,11

Additional results on the effects of pruning at different levels of density

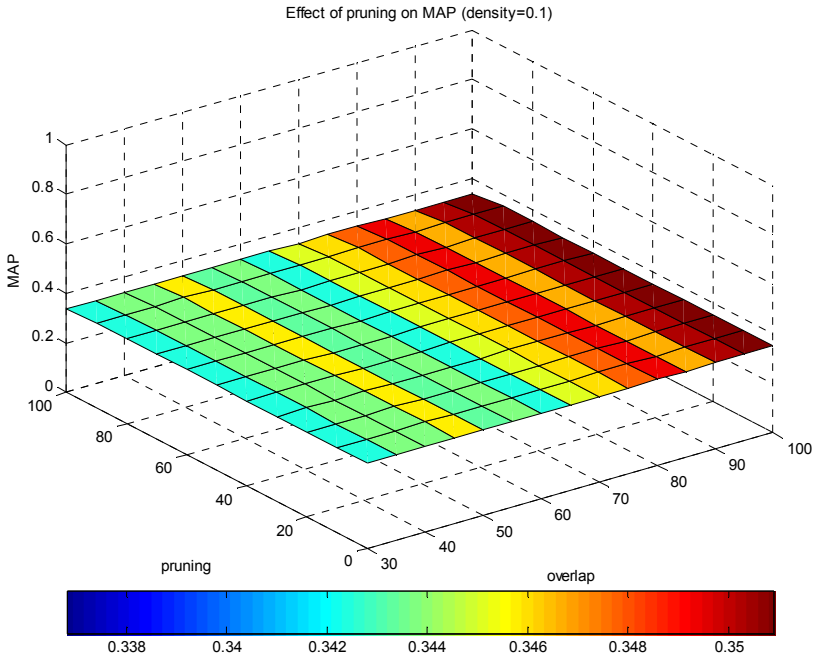


Figure AIV.1 Effect of pruning on MAP when density=0.1

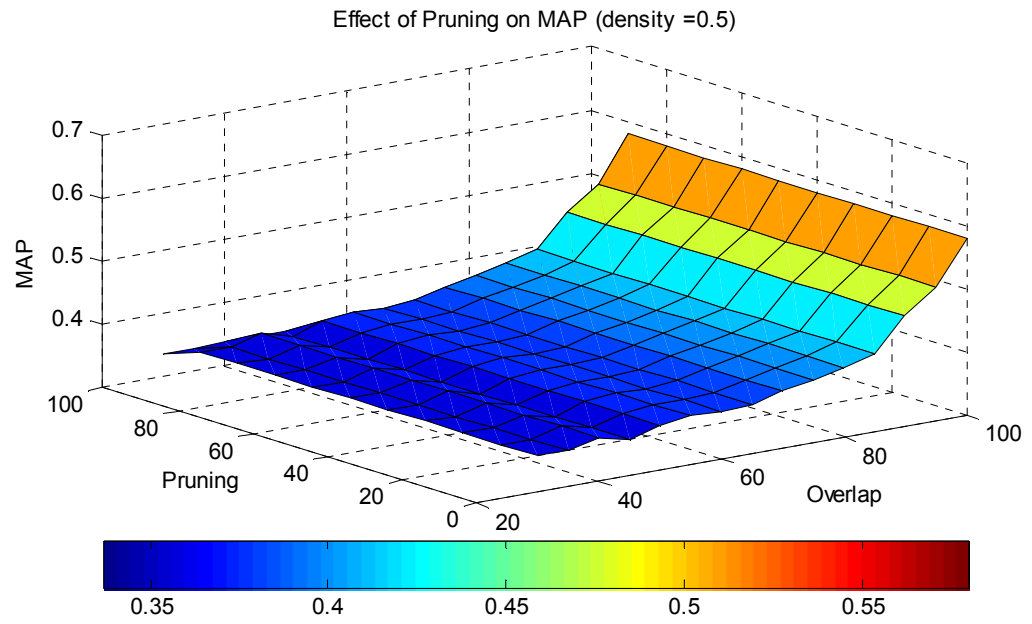


Figure AIV.2 Effect of pruning on MAP when density=0.5

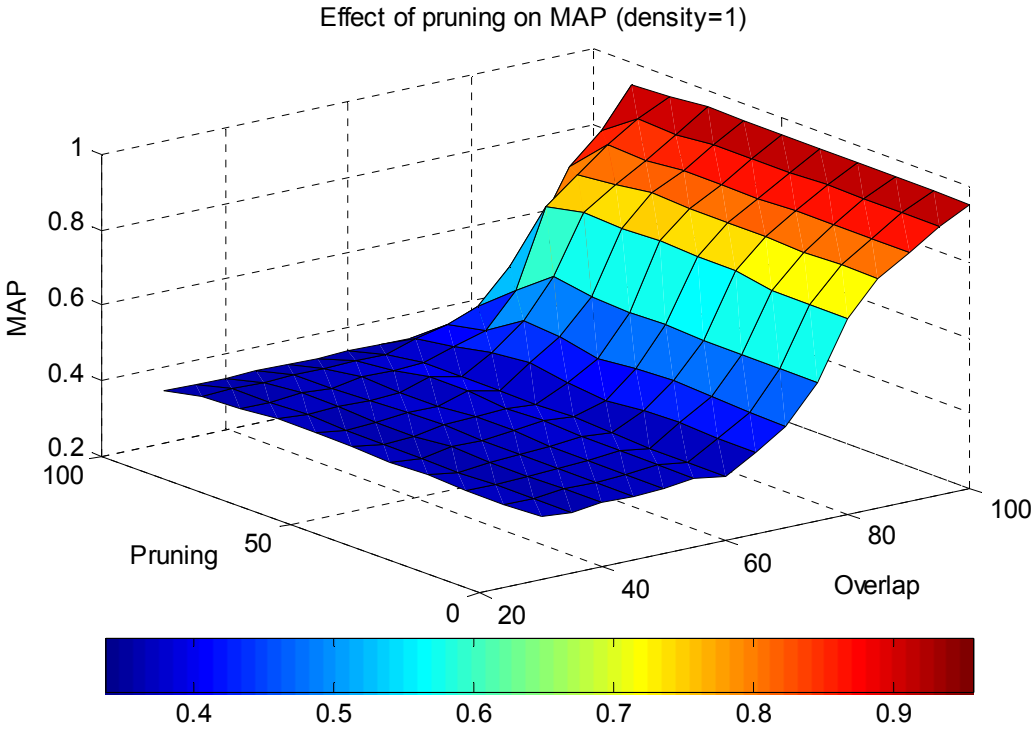
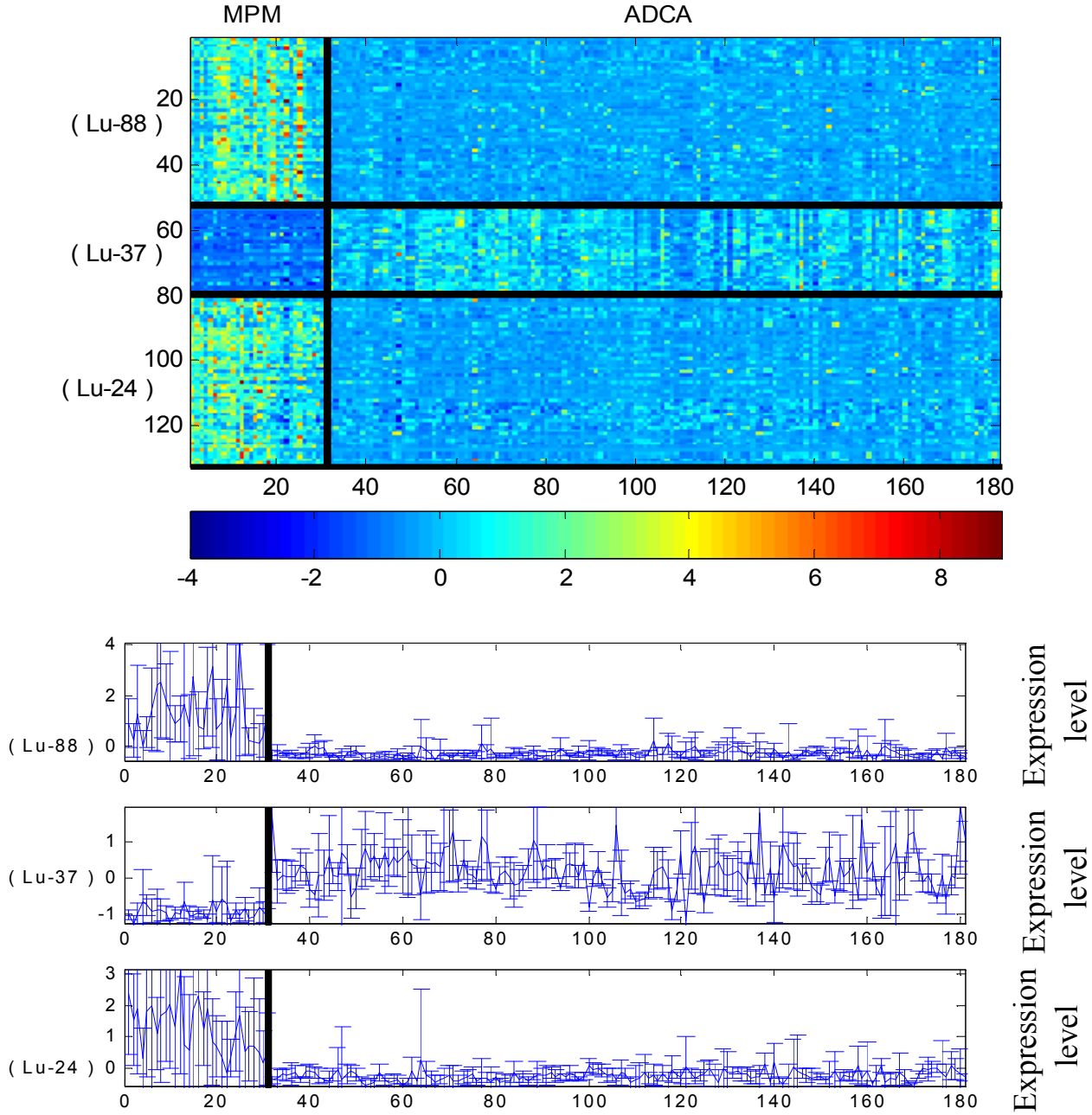


Figure AIII.3 Effect of pruning on MAP when density=1

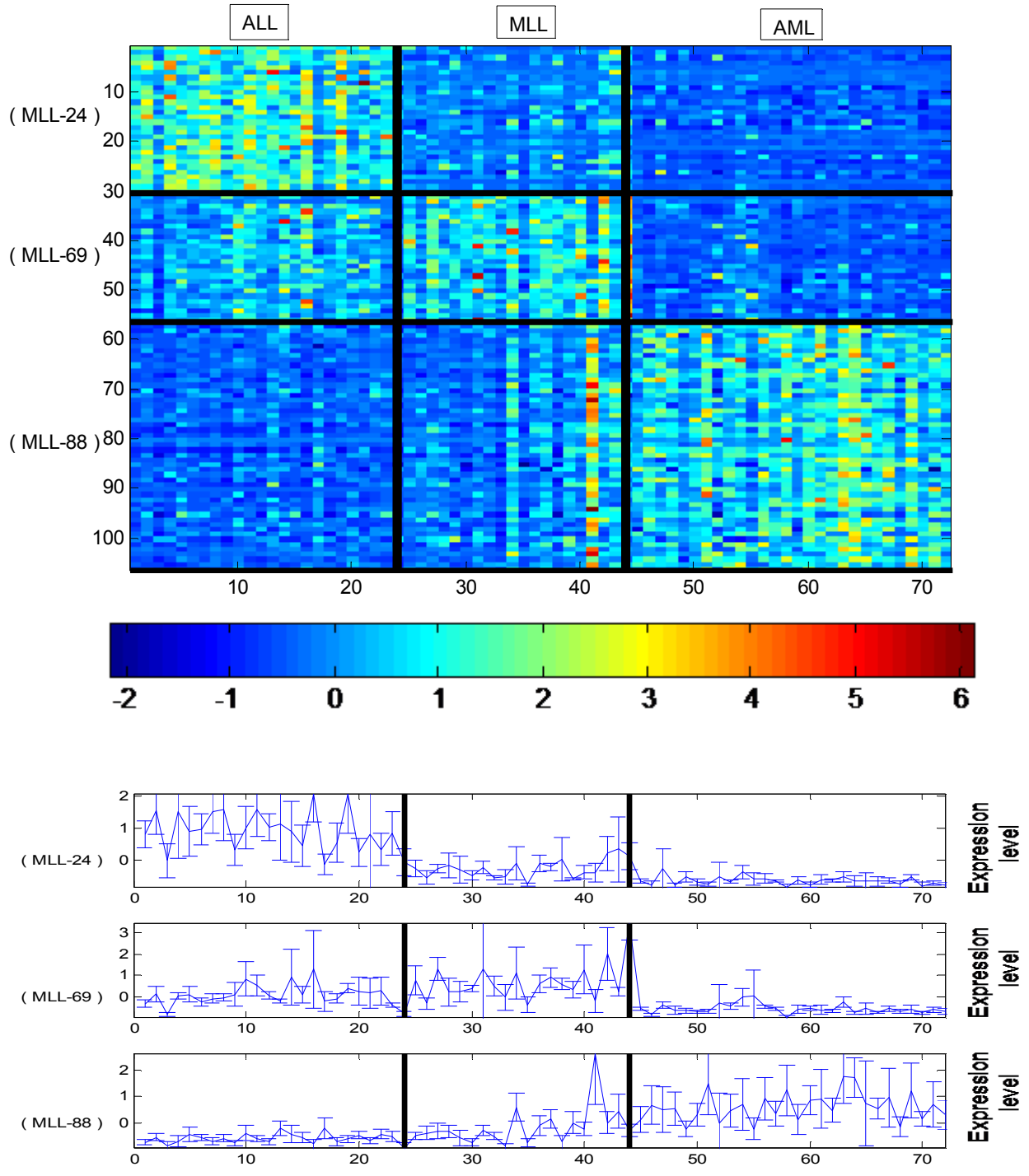
Table AIV.3 Comparison of MAP and NMI scores on the 20-Newsgroup datasets when using a PAM based feature selection (χ-Sim$_p$ denotes the result of χ-Sim with pruning)

	$M2_{PAM}$	$M5_{PAM}$	$M10_{PAM}$	$NG1_{PAM}$	$NG2_{PAM}$	$NG3_{PAM}$
Cosine	0.61 ± 0.04	0.54 ± 0.08	0.39 ± 0.03	0.52 ± 0.01	0.60 ± 0.02	0.49 ± 0.02
LSA	0.79 ± 0.09	0.66 ± 0.05	0.44 ± 0.04	0.56 ± 0.05	0.61 ± 0.06	0.52 ± 0.03
ITCC	0.50 ± 0.05	0.54 ± 0.05	0.29 ± 0.05	0.51 ± 0.06	0.44 ± 0.08	0.49 ± 0.07
SNOS	0.51 ± 0.01	0.26 ± 0.04	0.20 ± 0.02	0.51 ± 0.00	0.24 ± 0.01	0.22 ± 0.02
X-Sim	0.58 ± 0.07	0.62 ± 0.12	0.43 ± 0.04	0.54 ± 0.03	0.60 ± 0.12	0.47 ± 0.05
X-Sim$_p$	**0.65 ± 0.09**	**0.68 ± 0.06**	**0.47 ± 0.04**	**0.62 ± 0.12**	**0.63 ± 0.14**	**0.57 ± 0.04**

Appendix V



Lung Cancer dataset



Mixed Lineage Leukemia

Additional Enrichment of GO biological processes in gene clusters

Cluster	Biological Process	Count	Percentage
	Biosynthesis & Regulation	11	36.7
	Metabolism & Homeostasis	10	33.3
(ML-24)	Immune responses	12	40.0
	Cell proliferation & Differentiation	6	20.0
	Cell signaling	9	30.0
	Biosynthesis & Regulation	9	34.6
	Metabolism & Homeostasis	6	23.1
(ML-69)	Immune responses	9	34.6
	Cell proliferation & Differentiation	7	26.9
	Cell signaling	9	34.6
	Biosynthesis & Regulation	15	30.0
	Metabolism & Homeostasis	18	36.0
(ML-88)	Immune responses	9	18.0
	Cell proliferation & Differentiation	22	44.0
	Cell signaling & Adhesion	14	28.0

Appendix VI

List of genes in each of the selected clusters for Colon cancer, Leukemia, Lung cancer and Mixed Lineage Leukemia.

Colon cancer

CLUSTER ID = 62
==========

No. of genes = 17

[1]	Hsa.28914	Z48541	gene	1		H.sapiens mRNA for protein tyrosine phosphatase
[2]	Hsa.36010	H27771	3' UTR	2a	162849	G1/S-SPECIFIC CYCLIN D1 (Homo sapiens)
[3]	Hsa.3194	T60318	3' UTR	1	78472	A42811 INITIATION FACTOR EIF-4A HOMOLOG - ;.
[4]	Hsa.9691	T49703	3' UTR	2a	67944	60S ACIDIC RIBOSOMAL PROTEIN P1 (Polyorchis penicillatus)
[5]	Hsa.45499	H87344	3' UTR	2a	252396	SERUM ALBUMIN PRECURSOR (Homo sapiens)
[6]	Hsa.465	U07695	gene	1		"Human tyrosine kinase (HTK) mRNA, complete cds.
[7]	Hsa.1515	M85289	gene	1		"Human heparan sulfate proteoglycan (HSPG2) mRNA, complete cds.
[8]	Hsa.22762	H17434	3' UTR	1	50609	NUCLEOLIN (HUMAN);.
[9]	Hsa.318	M60706	gene	1		"Homo sapiens type I DNA topoisomerase gene, exon 21.
[10]	Hsa.5142	H82719	3' UTR	2a	249109	BETA-ADAPTIN (Homo sapiens)
[11]	Hsa.27592	H42477	3' UTR	2a	182692	RAS-RELATED C3 BOTULINUM TOXIN SUBSTRATE 1 (Homo sapiens)
[12]	Hsa.2007	X02744	gene	1		Human mRNA for c-sis gene (clone pSM-1).
[13]	Hsa.168	U02493	gene	1		"Human 54 kDa protein mRNA, complete cds.
[14]	Hsa.11765	R54854	3' UTR	2a	154475	VON WILLEBRAND FACTOR PRECURSOR (Homo sapiens)
[15]	Hsa.3228	U29171	gene	1		"Human casein kinase I delta mRNA, complete cds.
[16]	Hsa.7395	R10066	3' UTR	2a	128808	PROHIBITIN (Homo sapiens)
[17]	Hsa.1485	T62864	3' UTR	1	79590	"GUANINE NUCLEOTIDE-BINDING PROTEIN G(S), ALPHA SUBUNIT (HUMAN);

CLUSTER ID = 62
==========

No. of genes = 4

[1]	Hsa.41123	J00277	gene	1		"Human (genomic clones lambda-[SK2-T2, HS578T]; cDNA clones RS-[3,4, 6]) c-Ha-ras1 proto-oncogene, complete coding sequence.
[2]	Hsa.1205	R08183	3' UTR	1	127228	"Q04984 10 KD HEAT SHOCK PROTEIN, MITOCHONDRIAL ;.
[3]	Hsa.25748	R60883	3' UTR	2a	42504	"HLA CLASS II HISTOCOMPATIBILITY ANTIGEN, DQ(W1.1) BETA CHAIN PRECURSOR (Homo sapiens) "
[4]	Hsa.45732	H89477	3' UTR	2a	240171	G1/S-SPECIFIC CYCLIN D3 (Homo sapiens)

CLUSTER ID = 63
==========

No. of genes = 13

[1]	Hsa.1288	T53889	3' UTR	1	78017	COMPLEMENT C1R COMPONENT PRECURSOR (HUMAN).
[2]	Hsa.84	D11086	gene	1		Human mRNA for interleukin 2 receptor gamma chain.
[3]	Hsa.146	J04813	gene	1		"Human cytochrome P450 PCN3 gene, complete cds.
[4]	Hsa.11628	T60855	3' UTR	2a	76721	CALCINEURIN B SUBUNIT ISOFORM 1 (Homo sapiens)
[5]	Hsa.275	L10413	gene	1		"Human farnesyltransferase alpha-subunit mRNA, complete cds.
[6]	Hsa.33368	H88250	3' UTR	2a	252792	TRANSFORMER-2 SEX-DETERMINING PROTEIN (Drosophila melanogaster)
[7]	Hsa.4937	R43914	3' UTR	2a	33065	CREB-BINDING PROTEIN (Mus musculus)
[8]	Hsa.1855	M14764	gene	1		LOW-AFFINITY NERVE GROWTH FACTOR RECEPTOR PRECURSOR (HUMAN);.
[9]	Hsa.16742	R38513	3' UTR	2a	26871	FIBROBLAST GROWTH FACTOR RECEPTOR 2 PRECURSOR (Homo sapiens)

[10] Hsa.31801 R65697 3' UTR 2a 139435 ATP SYNTHASE A CHAIN (Trypanosoma brucei brucei)
[11] Hsa.3215 T74896 3' UTR 1 84893 SERUM AMYLOID A PROTEIN PRECURSOR (HUMAN);.
[12] Hsa.3198 Z47087 gene 1 H.sapiens mRNA for RNA polymerase II elongation factor-like protein.
[13] Hsa.114 L10717 gene 1 TYROSINE-PROTEIN KINASE LYK (HUMAN);contains Alu repetitive element;.

CLUSTER ID = 2
==========
No. of genes = 5
[1] HSAC07 control
[2] Hsa.8068 T57619 3' UTR 2a 75437 40S RIBOSOMAL PROTEIN S6 (Nicotiana tabacum)
[3] Hsa.1139 T88723 3' UTR 1 109876 UBIQUITIN (HUMAN);.
[4] Hsa.3006 T61602 3' UTR 1 78084 40S RIBOSOMAL PROTEIN S11 (HUMAN);.
[5] Hsa.5398 T58861 3' UTR 2a 77563 60S RIBOSOMAL PROTEIN L30E (Kluyveromyces lactis)

CLUSTER ID = 5
==========
No. of genes = 7
[1] Hsa.361 T63499 3' UTR 1 81466 "HLA CLASS I HISTOCOMPATIBILITY ANTIGEN, A-26(A-10) A*2601 ALPHA (HUMAN);. "
[2] Hsa.2542 X57346 gene 1 H.sapiens mRNA for HS1 protein.
[3] Hsa.36957 H43908 3' UTR 2a 183315 TRANSFORMING GROWTH FACTOR BETA 2 PRECURSOR (Gallus gallus)
[4] Hsa.2800 X55715 gene 1 Human Hums3 mRNA for 40S ribosomal protein s3.
[5] Hsa.2063 M24398 gene 1 "Human parathymosin mRNA, complete cds. (HUMAN);contains element MER22 repetitive element ;. "
[6] Hsa.36957 H43908 3' UTR 2a 183315 TRANSFORMING GROWTH FACTOR BETA 2 PRECURSOR (Gallus gallus)
[7] Hsa.954 T72938 3' UTR 1 84350 QM PROTEIN (HUMAN);.

CLUSTER ID = 6
==========
No. of genes = 10
[1] Hsa.27685 R50158 3' UTR 2a 153229 MITOCHONDRIAL LON PROTEASE HOMOLOG PRECURSOR (Homo sapiens)
[2] Hsa.489 T47144 3' UTR 1 74837 JN0549 RIBOSOMAL PROTEIN YL30.
[3] Hsa.1500 M11354 gene 1 "Human H3.3 histone, class B mRNA, complete cds.
[4] Hsa.1985 T52185 3' UTR 1 71940 P17074 40S RIBOSOMAL PROTEIN.
[5] Hsa.10363 T89730 3' UTR 2a 117433 TUBULIN BETA-1 CHAIN (Gallus gallus)
[6] Hsa.3566 T57633 3' UTR 1 75467 40S RIBOSOMAL PROTEIN S8 (HUMAN).
[7] Hsa.689 T62878 3' UTR 1 79597 CYTOCHROME C OXIDASE POLYPEPTIDE IV PRECURSOR (HUMAN);.
[8] Hsa.624 M57710 gene 1 "Human IgE-binding protein (epsilon-BP) mRNA, complete cds.
[9] Hsa.1978 T72879 3' UTR 1 84299 60S RIBOSOMAL PROTEIN L7A (HUMAN);.
[10] Hsa.3409 T53396 3' UTR 2a 68775 60S ACIDIC RIBOSOMAL PROTEIN P1 (Polyorchis penicillatus)

CLUSTER ID = 12
==========
No. of genes = 21
[1] Hsa.1822 M27539 gene 1 "Human MHC class I HLA-A cell surface antigen mRNA, (HLA-A28,-B40, -Cw3), clone LB45. "
[2] Hsa.580 Z22572 gene 1 H.sapiens CDEI binding protein mRNA.
[3] Hsa.24948 R50864 3' UTR 2a 37564 HETEROGENEOUS NUCLEAR RIBONUCLEOPROTEIN K (Homo sapiens)
[4] Hsa.2463 X83535 gene 1 H.sapiens mRNA for membrane-type matrix metalloproteinase.
[5] Hsa.3003 T57630 3' UTR 1 75459 S34195 RIBOSOMAL PROTEIN L3 -.
[6] Hsa.812 L28010 gene 1 "Homo sapiens HnRNP F protein mRNA, complete cds.
[7] Hsa.1130 Z24727 gene 1 "H.sapiens tropomyosin isoform mRNA, complete CDS.
[8] Hsa.2360 X15822 gene 1 Human COX VIIa-L mRNA for liver-specific cytochrome c oxidase (EC 1.9.3.1.).
[9] Hsa.570 L12168 gene 1 "Homo sapiens adenylyl cyclase-associated protein (CAP) mRNA, complete cds. "
[10] Hsa.36074 H28452 3' UTR 2a 181783 BRAIN CALCIUM CHANNEL BI-2 PROTEIN (Oryctolagus cuniculus)
[11] Hsa.304 L16510 gene 1 "Homo sapiens cathepsin B mRNA, complete cds.
[12] Hsa.7498 R21547 3' UTR 2a 130227 TRANS-ACTING TRANSCRIPTIONAL PROTEIN ICP0 (Bovine herpesvirus type 1)
[13] Hsa.338 D30655 gene 1 Human mRNA for eukaryotic initiation factor 4AII.
[14] Hsa.2846 X02152 gene 1 "Human mRNA for lactate dehydrogenase-A (LDH-A, EC 1.1.1.27).
[15] Hsa.915 H50623 3' UTR 1 186767 "HLA CLASS II HISTOCOMPATIBILITY ANTIGEN, DR-7 BETA CHAIN (HUMAN);. "

[16] Hsa.9817 T50334 3' UTR 2a 72201 14-3-3-LIKE PROTEIN GF14 OMEGA (Arabidopsis thaliana)
[17] Hsa.1181 L09159 gene 1 "Homo sapiens RHOA proto-oncogene multi-drug-resistance protein mRNA, 3' end. "
[18] Hsa.31882 R67075 3' UTR 2a 140771 ZINC FINGER X-CHROMOSOMAL PROTEIN (Mus musculus)
[19] Hsa.2710 X53743 gene 1 H.sapiens mRNA for fibulin-1 C.
[20] Hsa.2095 T51852 3' UTR 1 75026 VIMENTIN (HUMAN).
[21] Hsa.3295 M22806 gene 1 "Human prolyl 4-hydroxylase beta-subunit and disulfide isomerase (P4HB) gene, exon 11, clones 6B-(1,3,5,6).

"

Leukeamia

CLUSTER ID = 18
==========
No. of genes = 37
[1] KIAA0260 gene, partial cds D87449
[2] KIAA0261 gene, partial cds D87450
[3] KIAA0262 gene D87451
[4] KIAA0263 gene D87452
[5] KIAA0264 gene, partial cds D87453
[6] KIAA0266 gene D87455
[7] KIAA0269 gene
[8] KIAA0270 gene, partial cds
[9] KIAA0272 gene, partial cds
[10] KIAA0274 gene D87464
[11] KIAA0275 gene D87465
[12] KIAA0276 gene, partial cds
[13] KIAA0277 gene D87467
[14] KIAA0278 gene, partial cds D87468
[15] KIAA0279 gene, partial cds D87469
[16] Heat shock transcription factor 4 D87673
[17] KIAA0242 gene, partial cds D87684
[18] KIAA0267 gene, partial cds D87743
[19] GB DEF = Alpha(1,2)fucosyltransferase, 5'UTR partial sequence D87937
[20] RTP D87953
[21] CMP-sialic acid transporter D87969
[22] UDP-galactose transporter related isozyme 1 D87989_
[23] Retina-specific amine oxidase D88213
[24] GB DEF = (lambda) DNA for immunoglobin light chain D88270_
[25] CYSTATIN A D88422_at
[26] HGCMa D88613_
[27] Neuroblastoma D89016
[28] Proton-ATPase-like protein
[29] Src-like adapter protein mRNA
[30] C-myc binding protein D89667
[31] CGM6 Carcinoembryonic antigen gene family member 6 (NCA-95)
[32] PDHA1 Pyruvate dehydrogenase (lipoamide) alpha 1
[33] PDHB Pyruvate dehydrogenase (lipoamide) beta
[34] ATF4 CAMP-dependent transcription factor ATF-4 (CREB2)
[35] TXGP1 Tax-transcriptionally activated glycoprotein 1 (34kD)
[36] CGM7 Carcinoembryonic antigen gene family member 7 D90276_
[37] Lamin-Like Protein (Gb:M24732) HG1078-HT1078

CLUSTER ID = 56
==========
No. of genes = 27
[1] Serotonin N-acetyltransferase gene U40391

[2] Pre-pro-megakaryocyte potentiating factor U40434
[3] Ikaros/LyF-1 homolog (hIk-1) mRNA U40462
[4] NAD(P) transhydrogenase U40490
[5] GB DEF = Tyrosyl-tRNA synthetase mRNA U40714
[6] Putative voltage-gated potassium channel (KVLQT1) mRNA, partial cds U40990
[7] Retinal protein (HRG4) mRNA U40998
[8] PRELP Proline arginine-rich end leucine-rich repeat protein U41344
[9] Gu protein mRNA, partial cds U41387
[10] Deleted in split hand/split foot 1 (DSS1) mRNA U41515
[11] OS-9 precurosor mRNA U41635
[12] DGUOK Deoxyguanosine kinase U41668
[13] HOXA9 Homeo box A9 U41813
[14] 54 kDa progesterone receptor-associated immunophilin FKBP54 mRNA, partial cds U42031
[15] Ladinin (LAD) mRNA U42408
[16] 5'-AMP-activated protein kinase, gamma-1 subunit mRNA U42412
[17] Cardiotrophin-1 (CTF1) mRNA U43030
[18] CDC37 homolog mRNA U43077
[19] GNAQ Guanine nucleotide binding protein (G protein), q polypeptide U43083
[20] PTCH Patched (Drosophila) homolog U43148
[21] GB DEF = Urocortin gene U43177
[22] Selenophosphate synthetase 2 (SPS2) mRNA U43286
[23] DRP2 Dystrophin related protein 2 U43519
[24] Protein tyrosine kinase PYK2 mRNA U43522
[25] Malignant melanoma metastasis-suppressor (KiSS-1) gene, mRNA U43527
[26] Kinase suppressor of ras-1 (KSR1) mRNA, partial cds U43586
[27] Signal transducing adaptor molecule STAM mRNA U43899

CLUSTER ID = 31
==========
No. of genes = 29

[1] CSF2 Colony-stimulating factor 2 (GM-CSF) M13207
[2] N-MYC PROTO-ONCOGENE PROTEIN M13241
[3] ESD Esterase D/formylglutathione hydrolase M13450
[4] G1P2 Interferon, alpha-inducible protein (clone IFI-15K) M13755
[5] ADA Adenosine deaminase M13792
[6] GB DEF = Involucrin gene, exon 2 M13903
[7] RPS14 gene (ribosomal protein S14) extracted from Human ribosomal protein S14 gene M13934_cds2
[8] Mesothelial keratin K7 (type II) mRNA, 3' end M13955
[9] UROD Uroporphyrinogen decarboxylase M14016
[10] C1R Complement component C1r M14058
[11] T-cell receptor beta-chain J1.3 gene extracted from Human T-cell receptor germline beta-chain D1.1 and J1.1 to J1.6 genes M14158_cds4
[12] Diazepam binding inhibitor (DBI) mRNA M14200_rna1
[13] DCN Decorin M14219
[14] COAGULATION FACTOR XIII A CHAIN M14539
[15] PYGL Glycogen phosphorylase L (liver form) M14636
[16] GB DEF = ISG-54K gene (interferon stimulated gene) encoding a 54 kDA protein, exon 2 M14660

[17] FYN FYN oncogene related to SRC, FGR, YES M14676
[18] NGFR Nerve growth factor receptor M14764
[19] RAS-RELATED PROTEIN R-RAS M14949
[20] FCER2 Fc fragment of IgE, low affinity II, receptor for (CD23A) M15059
[21] GUSB Glucuronidase, beta M15182
[22] TK1 Thymidine kinase 1, soluble M15205
[23] RPL44 Ribosomal protein L44 M15661

[24] PCNA Proliferating cell nuclear antigen M15796
[25] SNRPB2 Small nuclear ribonucleoprotein polypeptide B" M15841
[26] LPL Lipoprotein lipase M15856
[27] C-yes-1 mRNA M15990
[28] LYN V-yes-1 Yamaguchi sarcoma viral related oncogene homolog M16038
[29] MIC2 Antigen identified by monoclonal antibodies 12E7, F21 and O13 M16279

MLL database

CLUSTER ID = 24
==========
No. of genes = 30
[1] 41096_at "Hs.272537 gnl|UG|Hs#S1611 Human terminal transferase mRNA, complete cds "
[2] 41097_at "Hs.58831 gnl|UG|Hs#S1055401 Homo sapiens anti-Fas-induced apoptosis (TOSO) mRNA, complete cds
[3] 41108_at "Hs.110309 gnl|UG|Hs#S1570456 Human DNA sequence from clone 377H14 on chromosome 6p21.32-22.1. Contains the HLA-G gene for major histocompatibility complex, class I, G (HLA 6.0) two MHC class I pseudogenes, an RPL7A (60S Ribosomal Protein L7A) pseudogene, a gene for"
[4] 41117_s_at Hs.171558 gnl|UG|Hs#S1368046 Homo sapiens mRNA for SCML2 protein
[5] 41366_at "Hs.1832 gnl|UG|Hs#S1170306 qi61f11.x1 Homo sapiens cDNA, 3' end "
[6] 41375_at Hs.89751 gnl|UG|Hs#S4524 Human mRNA for CD20 receptor (S7)
[7] 41383_at Hs.12702 gnl|UG|Hs#S1368196 Homo sapiens mRNA; cDNA DKFZp586N012 (from clone
DKFZp586N012)
[8] 41386_i_at Hs.153952 gnl|UG|Hs#S269480 Human placental cDNA coding for 5'nucleotidase (EC 3.1.3.5)
[9] 41396_at "Hs.167740 gnl|UG|Hs#S705595 Human butyrophilin (BTF5) mRNA, complete cds
[10] 41399_at Hs.68054 gnl|UG|Hs#S6010 H.sapiens Staf50 mRNA
[11] 41400_at "Hs.79219 gnl|UG|Hs#S1569334 Homo sapiens mRNA for KIAA0959 protein, partial cds "
[12] 41401_at "Hs.81071 gnl|UG|Hs#S892136 Human extracellular matrix protein 1 mRNA, complete cds "
[13] 41403_at Hs.166846 gnl|UG|Hs#S1570318 Homo sapiens mRNA; cDNA DKFZp434F222 (from clone
DKFZp434F222); partial cds
[14] 41406_at Hs.177556 gnl|UG|Hs#S512720 34b1 Homo sapiens cDNA
[15] 41407_at "Hs.198443 gnl|UG|Hs#S552752 Human inositol 1,4,5 trisphosphate receptor type 1 mRNA, partial cds "
[16] 41409_at "Hs.50651 gnl|UG|Hs#S1475018 DKFZp434D1112_s1 Homo sapiens cDNA, 3' end "
[17] 41423_at Hs.59545 gnl|UG|Hs#S705590 Human Ro
[18] 41438_at "Hs.6079 gnl|UG|Hs#S998781 Homo sapiens mRNA for KIAA0598 protein, complete cds "
[19] 41442_at "Hs.76884 gnl|UG|Hs#S1090351 Homo sapiens DNA sequence from PAC 150O5 on chromosome 1p36.13-36.22. Contains the E2F2 gene for transcription factor E2F-2 and the ID3 gene for Inhibitor of DNA binding 3 (dominant negative helix-loop-helix protein, 1R21, HEIR-1). Cont"
[20] 41446_f_at Hs.79015 gnl|UG|Hs#S377463 Human MRC OX-2 gene signal sequence
[21] 41447_at Hs.82749 gnl|UG|Hs#S551639 Human (clone CCG-B7) mRNA sequence
[22] 41450_at "Hs.79507 gnl|UG|Hs#S1444573 wg66h09.x1 Homo sapiens cDNA, 3' end "
[23] 41454_at "Hs.152601 gnl|UG|Hs#S592268 Homo sapiens mRNA for ceramide glucosyltransferase, complete cds "
[24] 41460_at Hs.179666 gnl|UG|Hs#S513781 48h12 Homo sapiens cDNA μ
[25] 41462_at "Hs.19280 gnl|UG|Hs#S1362042 wb55f10.x1 Homo sapiens cDNA, 3' end "
[26] 41468_at "Hs.50651 gnl|UG|Hs#S1493 Human protein-tyrosine kinase (JAK1) mRNA, complete cds "
[27] 41470_at "Hs.155530 gnl|UG|Hs#S2667 Human interferon-gamma induced protein (IFI 16) gene,

complete cds "
[28] 41471_at "Hs.1298 gnl|UG|Hs#S1945 Human common acute lymphoblastic leukemia antigen (CALLA) mRNA, complete cds "
[29] 41475_at "M21535 /FEATURE= /DEFINITION=HUMERG11 Human erg protein (ets-related gene) mRNA, complete cds "
[30] 41609_at "Hs.198443 gnl|UG|Hs#S1741 Human mRNA for type 1 inositol 1,4,5-trisphosphate receptor, complete cds "

CLUSTER ID = 69
==========
No. of genes = 26

[1] 36637_at "Hs.116481 gnl|UG|Hs#S1247 Human B cell differentiation antigen mRNA, complete cds "
[2] 36638_at "Hs.112360 gnl|UG|Hs#S952801 Homo sapiens AC133 antigen mRNA, complete cds "
[3] 36641_at Hs.278571 gnl|UG|Hs#S472864 H.sapiens mRNA for mosaic protein LR11
[4] 36644_at Hs.180737 gnl|UG|Hs#S892186 Homo sapiens clone 23664 and 23905 mRNA sequence
[5] 36645_at Hs.40034 gnl|UG|Hs#S4072 Human mRNA for integrin alpha-4 subunit
[6] 36650_at "Hs.78361 gnl|UG|Hs#S1091058 Homo sapiens m6b1 mRNA, complete cds "
[7] 36652_at "Hs.78993 gnl|UG|Hs#S377250 Human RasGAP-related protein (IQGAP2) mRNA, complete cds "
[8] 36654_s_at "Hs.79440 gnl|UG|Hs#S705733 Homo sapiens putative RNA binding protein KOC (koc) mRNA,
complete cds "
[9] 36657_at "Hs.151518 gnl|UG|Hs#S342562 Human TAR RNA loop binding protein (TRP-185) mRNA, complete cds "
[10] 36661_s_at Hs.17144 gnl|UG|Hs#S1263316 Homo sapiens retinal short-chain dehydrogenase
[11] 36666_at "Hs.171734 gnl|UG|Hs#S342586 Human protein phosphatase 2A B'alpha1 regulatory subunit mRNA, complete cds "
[12] 36667_at "Hs.172028 gnl|UG|Hs#S876214 Homo sapiens ADAM10 (ADAM10) mRNA, complete cds "
[13] 36668_at "Hs.172674 gnl|UG|Hs#S4253 Homo sapiens NF-AT4c mRNA, complete cds "
[14] 36669_at Hs.211563 gnl|UG|Hs#S226081 H.sapiens mRNA for BCL7A protein
[15] 36674_at "Hs.238990 gnl|UG|Hs#S1207368 qo19f03.x1 Homo sapiens cDNA, 3' end "
[16] 36675_r_at Hs.250641 gnl|UG|Hs#S5518 Human mRNA for fibroblast tropomyosin TM30 (pl)
[17] 36676_at "Hs.238126 gnl|UG|Hs#S442104 zh96a09.r1 Homo sapiens cDNA, 5' end "
[18] 36678_at "Hs.5947 gnl|UG|Hs#S1486094 wj88e11.x1 Homo sapiens cDNA, 3' end "
[19] 36684_at Hs.74441 gnl|UG|Hs#S305305 H.sapiens mRNA for 218kD Mi-2 protein
[20] 36685_at Hs.82845 gnl|UG|Hs#S592284 Human clone 23815 mRNA sequence
[21] 36687_at "Hs.82890 gnl|UG|Hs#S226058 Human mRNA for DAD-1, complete cds "
[22] 36690_at "Hs.25816 gnl|UG|Hs#S389357 zb53g09.s1 Homo sapiens cDNA, 3' end "
[23] 36933_at "Hs.758 gnl|UG|Hs#S2192 Human GTPase-activating protein ras p21 (RASA) mRNA, complete cds "
[24] 36937_s_at "Hs.78993 gnl|UG|Hs#S377250 Human RasGAP-related protein (IQGAP2) mRNA, complete cds "
[25] 36938_at "Hs.172674 gnl|UG|Hs#S4253 Homo sapiens NF-AT4c mRNA, complete cds "
[26] 36940_at "Hs.171734 gnl|UG|Hs#S342586 Human protein phosphatase 2A B'alpha1 regulatory subunit mRNA, complete cds "

CLUSTER ID = 88
==========
No. of genes = 50

[1] 1826_at "Hs.234762 gnl|UG|Hs#S133263 yg02e11.s1 Homo sapiens cDNA, 3' end "
[2] 1827_s_at "Hs.241597 gnl|UG|Hs#S875622 hPROT=brain-specific L-proline transporter [human, hippocampus, mRNA Partial, 1911 nt] "
[3] 1830_s_at Hs.143513 gnl|UG|Hs#S875878 Homo sapiens mRNA for C8FW phosphoprotein
[4] 1792_g_at "Hs.166109 gnl|UG|Hs#S6085 Human ELAV-like neuronal protein-2 Hel-N2 mRNA, complete cds "
[5] 1794_at "Hs.2605 gnl|UG|Hs#S4558 H.sapiens CMRF35 mRNA, complete CDS "
[6] 1795_g_at "Hs.64797 gnl|UG|Hs#S554014 APPH=amyloid precursor protein homolog [human, placenta, mRNA,
3727 nt] "
[7] 1796_s_at "Hs.225932 gnl|UG|Hs#S2672 Human pregnancy-specific beta 1-glycoprotein 7 (PSG7) mRNA, complete cds"
[8] 1797_at "Hs.83731 gnl|UG|Hs#S1967 Human differentiation antigen (CD33) mRNA, complete cds "
[9] 1801_at "Hs.166146 gnl|UG|Hs#S1263611 Homo sapiens homer-3 mRNA, complete cds "
[10] 1810_s_at "Hs.188021 gnl|UG|Hs#S3602 Human putative potassium channel subunit (h-erg) mRNA, complete cds "
[11] 1790_s_at Hs.143434 gnl|UG|Hs#S4560 H.sapiens contactin mRNA
[12] 1776_at Hs.1437 gnl|UG|Hs#S553536 Human lysosomal alpha-glucosidase gene exon 1
[13] 1779_s_at "Hs.155342 gnl|UG|Hs#S622 Homo sapiens mRNA for protein kinase C delta-type, complete cds "
[14] 1780_at "Hs.198287 gnl|UG|Hs#S5801 Human pregnancy-specific glycoprotein 13 (PSG13') mRNA, complete cds "
[15] 1782_s_at "Hs.31074 gnl|UG|Hs#S305515 Human N-sulphoglucosamine sulphohydrolase mRNA, complete cds "
[16] 1786_at "Hs.54434 gnl|UG|Hs#S377209 Human interferon regulatory factor 5 (Humirf5) mRNA, complete cds "
[17] 1789_at Hs.75082 gnl|UG|Hs#S5337 H.sapiens rhoG mRNA for GTPase
[18] 1768_s_at Hs.771 gnl|UG|Hs#S1055352 untitled
[19] 1748_s_at Hs.83968 gnl|UG|Hs#S2778 Human leukocyte adhesion protein (LFA-1)
[20] 1750_at "Hs.101382 gnl|UG|Hs#S822 Homo sapiens B94 protein mRNA, complete cds "
[21] 1751_g_at "Hs.1432 gnl|UG|Hs#S2177 Human 80K-H protein (kinase C substrate) mRNA, complete cds "
[22] 1752_at Hs.195464 gnl|UG|Hs#S1570079 Homo sapiens mRNA; cDNA DKFZp586K1720 (from clone DKFZp586K1720)
[23] 1725_s_at Hs.5097 gnl|UG|Hs#S998351 Homo sapiens mRNA for synaptogyrin 2
[24] 1728_at "Hs.74573 gnl|UG|Hs#S472959 Human HU-K4 mRNA, complete cds "
[25] 1737_s_at Hs.75108 gnl|UG|Hs#S5299 Human mRNA for ribonuclease
[26] 1706_at "Hs.75725 gnl|UG|Hs#S614 Human mRNA for KIAA0120 gene, complete cds "
[27] 1707_g_at Hs.279929 gnl|UG|Hs#S226284 H.sapiens mRNA for gp25L2 protein
[28] 1693_s_at Hs.75873 gnl|UG|Hs#S472832 Homo sapiens mRNA for zyxin
[29] 1695_at "Hs.76159 gnl|UG|Hs#S3060 Human vacuolar H+ ATPase proton channel subunit mRNA, complete cds "
[30] 1659_s_at "Hs.76494 gnl|UG|Hs#S305423 Human prolargin (PRELP) gene, 5' flanking sequence and "
[31] 1660_at Hs.77436 gnl|UG|Hs#S5238 Human mRNA for pleckstrin (P47)
[32] 1662_r_at "Hs.89674 gnl|UG|Hs#S3733 Human mRNA for KIAA0115 gene, complete cds "
[33] 1663_at "Hs.9999 gnl|UG|Hs#S876059 Human hematopoietic neural membrane protein (HNMP-1) mRNA, complete cds "
[34] 1665_s_at Hs.155597 gnl|UG|Hs#S779 Human adipsin
[35] 1667_s_at Hs.76171 gnl|UG|Hs#S592150 H.sapiens mRNA for CCAAT
[36] 1675_at "Hs.29764 gnl|UG|Hs#S205920 yj12d03.s1 Homo sapiens cDNA, 3' end "
[37] 1636_g_at "Hs.227789 gnl|UG|Hs#S377217 Human mitogen activated protein kinase activated protein kinase-3 mRNA, complete cds "
[38] 1637_at "Hs.155342 gnl|UG|Hs#S622 Homo sapiens mRNA for protein kinase C delta-type, complete cds "

[39] 1640_at Cell Division Cycle Protein 2-Related Protein Kinase (Pisslre)
[40] 1641_s_at "Hs.5831 gnl|UG|Hs#S268583 Human gene for tissue inhibitor of metalloproteinases, partial sequence "
[41] 1643_g_at "Hs.6441 gnl|UG|Hs#S2843 Human metalloproteinase inhibitor mRNA, complete cds "
[42] 1644_at "Hs.77266 gnl|UG|Hs#S342598 Homo sapiens bone-derived growth factor (BPGF-1) mRNA, complete cds "
[43] 1647_at "Hs.72924 gnl|UG|Hs#S2454 Homo sapiens cytidine deaminase (CDA) mRNA, complete cds "
[44] 1649_at Guanine Nucleotide-Binding Protein G25k
[45] 1650_g_at "Hs.157236 gnl|UG|Hs#S554448 Human BRCA1, Rho7 and vatI genes, complete cds, and ipf35 gene, partial cds "
[46] 1651_at "Hs.54434 gnl|UG|Hs#S377209 Human interferon regulatory factor 5 (Humirf5) mRNA, complete cds "
[47] 1652_at "Hs.54434 gnl|UG|Hs#S377209 Human interferon regulatory factor 5 (Humirf5) mRNA, complete cds "
[48] 1633_g_at Hs.79572 gnl|UG|Hs#S268913 Human cathepsin D (catD) gene
[49] 1622_at "Hs.994 gnl|UG|Hs#S1431 Homo sapiens phospholipase C-beta-2 mRNA, complete cds "
[50] 1627_at Hs.10029 gnl|UG|Hs#S270386 H.sapiens mRNA for cathepsin C

References

Abdullah, A., and A. Hussain. 2006. “A new biclustering technique based on crossing minimization.” *Jounal of Neurocomputing* 69:1882–1896.

Abdullah, A., and A. Hussain. 2005. “Biclustering gene expression data in the presence of noise.” *Artificial Neural Networks* 3696:611-16.

Achlioptas, D., and F. McSherry. 2005. “On spectral learning of mixtures of distributions.” Pp. 458–469 in *Proceedings of the 18th Annual Conference on Learning Theory (ACLT)*.

Ackerman, M., and S. Ben-David. 2008. “Measures of clustering quality: A working set of axioms for clustering.” in *Proceedings of the 2008 Conference in Neural Information Processing Systems (NIPS)*.

Aggarwal, C., A. Hinneburg, and D. Keim. 2001. “On the surprising behavior of distance metrics in high dimensional space.” *International Conference on Database Theory (ICDT)* 420–434.

Ahmad, W., and A. Khokhar. 2007. “cHawk: An Efficient Biclustering Algorithm based on Bipartite Graph Crossing Minimization.” in *Workshop on Data Mining in Bioinformatics, 33rd VLDB*, vol. 2007.

Alon, U. et al. 1999. “Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays.” *Proceedings of the National Academy of Sciences* 96:6745.

AlSumait, L., and C. Domeniconi. 2007. “Text clustering with local semantic kernels.” in *In Workshop on Text Mining, SIAM International Conference on Data Mining (SDM)*.

Amati, G., and C. J Van Rijsbergen. 2002. “Probabilistic models of information retrieval based on measuring the divergence from randomness.” *ACM Transactions on Information Systems (TOIS)* 20:357–389.

Ankerst, M., M. M Breunig, H. P Kriegel, and J. Sander. 1999. “OPTICS: Ordering points to identify the clustering structure.” *Proceedings of the ACM SIGMOD* 28:49-60.

Armstrong, S. A et al. 2002. “MLL translocations specify a distinct gene expression profile that distinguishes a unique leukemia.” *Nature genetics* 30:41–47.

Arthur, D., and S. Vassilvitskii. 2007. “k-means++: The advantages of careful seeding.” P. 1035 in *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*.

Backer, E., and A. K. Jain. 2009. “Clustering Performance Measure Based on Fuzzy Set Decomposition.” *IEEE Transactions on Pattern Analysis and Machine Intelligence* 66-75.

Baeza-Yates, R., and B. Ribeiro-Neto. 1999. “Modern Information Retrieval.” *Modern Information Retrieval, Harlow: Addison-Wesley, GB* 367–395.

Banerjee, A., I. Dhillon, J. Ghosh, S. Merugu, and D. S Modha. 2004. “A generalized maximum entropy approach to bregman co-clustering and matrix approximation.” Pp. 509–514 in *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Banerjee, A., and J. Ghosh. 2002. “Frequency sensitive competitive learning for clustering on high-dimensional

hyperspheres." Pp. 1590–1595 in *Proc. IEEE Int. Joint Conference on Neural Networks (IJCNN)*.

Banerjee, A., C. Krumpelman, J. Ghosh, S. Basu, and R. J Mooney. 2005. "Model-based overlapping clustering." Pp. 532-537 in *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*.

Barkow, S., S. Bleuler, A. Prelic, P. Zimmermann, and E. Zitzler. 2006. "BicAT: a biclustering analysis toolbox." *Journal of bioinformatics* 22:1282-1288.

Bekkerman, R., R. El-Yaniv, and A. McCallum. 2005. "Multi-way distributional clustering via pairwise interactions." in *Proceedings of the 22nd International Conference on Machine Learning (ICML)*.

Bekkerman, R., R. El-Yaniv, N. Tishby, and Y. Winter. 2003. "Distributional word clusters vs. words for text categorization." *The Journal of Machine Learning Research* 3:1183–1208.

Bellman, R. 1961. *Adaptive control processes: a guided tour.* Princeton University Press.

Ben-Dor, A., B. Chor, R. Karp, and Z. Yakhini. 2003. "Discovering local structure in gene expression data: the order-preserving submatrix problem." *Journal of Computational Biology* 10:373–384.

Benzécri, J. P. 1969. "Statistical analysis as a tool to make patterns emerge from data." *Methodologies of pattern recognition* 35-74.

Bergmann, S., J. Ihmels, and N. Barkai. 2003. "Iterative signature algorithm for the analysis of large-scale gene expression data." *Physical review E* 67:31902.

Berkhin, P. 2006. "Survey of clustering data mining techniques." *Grouping Multidimensional Data* pp 25-71.

Berry, M. W. 2007. *Survey of text mining: clustering, classification, and retrieval*. Springer-Verlag New York Inc.

Beyer, K., J. Goldstein, R. Ramakrishnan, and U. Shaft. 1999. "When is" nearest neighbor" meaningful?." *in International Conference on Database Theory (ICDT)*.

Bingham, E. 2003. "Advances in independent component analysis with applications to data mining." *Helsinki University of Technology: Dissertation for the degree of Doctor of Science in Technology*.

Bisson, G. 1992. "Conceptual clustering in a first order logic representation." Pp. 458–462 in *Proceedings of the 10th European conference on Artificial intelligence (ECAI)*.

Bisson, G., and F. Hussain. 2008. "Chi-Sim: A New Similarity Measure for the Co-clustering Task." Pp. 211–217 in *Proceedings of the 2008 Seventh International Conference on Machine Learning and Applications*.

Blondel, V. D, A. Gajardo, M. Heymans, P. Senellart, and P. Van Dooren. 2004. "A measure of similarity between graph vertices: Applications to synonym extraction and web searching." *Siam Review* 46:647–666.

Boser, B.E., I.M. Guyon, and V.N. Vapnik. 1992. "A training algorithm for optimal margin classifiers." *Proceedings of the fifth annual workshop on Computational learning theory*.

Brock, G. N, J. R Shaffer, R. E Blakesley, M. J Lotz, and G. C Tseng. 2008. "Which missing value imputation method to use in expression profiles: a comparative study and two selection schemes." *BMC bioinformatics* 9:12-18.

Budanitsky, A., and G. Hirst. 2006. "Evaluating wordnet-based measures of lexical semantic relatedness." *Journal of Computational Linguistics* 32:13–47.

Buhmann, J. M. 2003. "Data clustering and learning." *The Handbook of Brain Theory and Neural Networks* 278–

281.

Busygin, S., G. Jacobsen, E. Kr\ämer, and C. Ag. 2002. "Double conjugated clustering applied to leukemia microarray data." in *2nd SIAM International Conference on Data Mining (SDM), Workshop on clustering high dimensional data*.

Busygin, S., O. Prokopyev, and P. M Pardalos. 2008. "Biclustering in data mining." *Computers and Operations Research* 35:2964–2987.

Cai, R., L. Lu, and L. H Cai. 2005. "Unsupervised auditory scene categorization via key audio effects and information-theoretic co-clustering." Pp. 1073–1076 in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP05)*.

Candillier, L., I. Tellier, F. Torre, and O. Bousquet. 2006. "Cascade evaluation of clustering algorithms." Pp. 574–581 in *European Conference on Machine Learning*.

Carrasco, J. J., D. C. Fain, K. J. Lang, and L. Zhukov. 2003. "Clustering of bipartite advertiser-keyword graph. The ICDM 2003 Third IEEE International Conference on Data Mining." in *Workshop on Clustering Large Data Sets*.

Carroll, J. D, P. E Green, and C. M Schaffer. 1986. "Interpoint distance comparisons in correspondence analysis." *Journal of Marketing Research* 23:271–280.

Chakraborti, S., R. Lothian, N. Wiratunga, and S. Watt. 2006. "Sprinkling: supervised latent semantic Indexing." *Advances in Information Retrieval* 3936:510-514.

Chakraborti, S., R. Mukras, et al. 2007. "Supervised Latent Semantic Indexing using Adaptive Sprinkling." Pp. 1582–1587 in *Proc. of International Joint Conference on Artificial Intelligence (IJCAI)*.

Chakraborti, S., N. Wiratunga, R. Lothian, and S. Watt. 2007. "Acquiring Word Similarities with Higher Order Association Mining." *Case Based Resoning, Research and Development* 4626:61-76.

Chen, Y., E. K Garcia, M. R Gupta, A. Rahimi, and L. Cazzanti. 2009. "Similarity-based classification: Concepts and algorithms." *The Journal of Machine Learning Research* 10:747–776.

Cheng, Y., and G. M Church. 2000. "Biclustering of expression data." Pp. 93–103 in *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology*.

Chmielewski, M. R, and J. W Grzymala-Busse. 1996. "Global discretization of continuous attributes as preprocessing for machine learning." *International Journal of Approximate Reasoning* 15:319–331.

Cho, H., I. S Dhillon, Y. Guan, and S. Sra. 2004. "Minimum sum-squared residue co-clustering of gene expression data." in *Proceedings of the SIAM Data Mining Conference (SDM)*.

Choi, Heejun. 2003. "Processing and Analysis of DNA Microarrays." Doctoral Thesis, University of Louisville http://bioinformatics.louisville.edu/brg/publications.htm.

Choi, Y. S, and B. R Moon. 2007. "Feature Selection in Genetic Fuzzy Discretization for the Pattern Classification Problems." *IEICE TRANSACTIONS on Information and Systems* 90:1047.

Comon, P., and others. 1994. "Independent component analysis, a new concept?." *Signal processing* 36:287–314.

Cormack, R. M. 1971. "A review of classification." *Journal of the Royal Statistical Society. Series A (General)* 134:321–367.

Crick, F. 1970. "Central dogma of molecular biology." *Nature* 227:561–563.

Cuesta Sánchez, F., P. J. Lewi, and D. L. Massart. 1994. “Effect of Different Preprocessing Methods for Principal Component Analysis Applied to the Composition of Mixtuers: Detection of Impurities in HPLC- DA D.” *Journal of Chemometrics and intelligent laboratory systems* 25:157–177.

Dai, W., G. R Xue, Q. Yang, and Y. Yu. 2007. “Co-clustering based classification for out-of-domain documents.” Pp. 210–219 in *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*.

Dai, W., Q. Yang, G. R Xue, and Y. Yu. 2008. “Self-taught clustering.” Pp. 200–207 in *Proceedings of the 25th international conference on Machine learning*.

Dave, R. N. 1991. “Characterization and detection of noise in clustering.” *Pattern Recognition Letters* 12:657–664.

Davis, J. V, B. Kulis, P. Jain, S. Sra, and I. S Dhillon. 2007. “Information-theoretic metric learning.” P. 216 in *Proceedings of the 24th international conference on Machine learning*.

Day, W. H.E, and H. Edelsbrunner. 1984. “Efficient algorithms for agglomerative hierarchical clustering methods.” *Journal of classification* 1:7–24.

Deerwester, S., S. T Dumais, G. W Furnas, T. K Landauer, and R. Harshman. 1990. “Indexing by latent semantic analysis.” *Journal of the American society for information science* 41:391–407.

Dennis Jr, G. et al. 2003. “DAVID: database for annotation, visualization, and integrated discovery.” *Genome Biology* 4.

Deodhar, M., H. Cho, G. Gupta, J. Ghosh, and I. Dhillon. 2007. *Bregman Bubble Co-clustering*. University of Texas, Technical Report.

Dettling, M., and P. Bühlmann. 2002. “Supervised clustering of genes.” *Genome Biology* 3:1–0069.

Dhillon, I., Y. Guan, and B. Kulis. 2005. “A unified view of kernel k-means, spectral clustering and graph cuts.” *The University of Texas at Austin, Department of Computer Sciences. Technical Report TR-04* 25.

Dhillon, I. S. 2001. “Co-clustering documents and words using bipartite spectral graph partitioning.” Pp. 269–274 in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*.

Dhillon, I. S, S. Mallela, and D. S Modha. 2003. “Information-theoretic co-clustering.” Pp. 89–98 in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*.

Diday, E., and J. C. Simon. 1976. “Clustering Analysis, Digital Pattern Recognition.” *Springer-Verlag, Secaucus, NJ, pp* 47–94.

Ding, C. 2003. “Document retrieval and clustering: from principal component analysis to self-aggregation networks.” in *Proc. Int. Workshop on Artificial Intelligence and Statistics*.

Dougherty, J., R. Kohavi, and M. Sahami. 1995. “Supervised and unsupervised discretization of continuous features.” Pp. 194–202 in *International Conference on Machine Learning (ICML)*.

Dubnov, S. et al. 2002. “A new nonparametric pairwise clustering algorithm based on iterative estimation of distance profiles.” *Journal of Machine Learning* 47:35–61.

Duda, R. O, P. E Hart, and D. G Stork. 2001. *Pattern classification*. John Wiley & Sons.

Dudoit, S., J. Fridlyand, and T. P Speed. 2002. “Comparison of discrimination methods for the classification of

tumors using gene expression data..” *Journal of the American Statistical Association* 97:77–88.

Dumais, S. T. 1994. “Latent semantic indexing (LSI): TREC-3 report.” Pp. 219–230 in *Overview of the Third Text Retrieval Conference (TREC-3)*.

Eades, P., and N. Wormald. 1986. *The median heuristic for drawing 2-layers networks*. Technical Report 69, Department of Computer Science, University of Queensland, 1986.

El-Yaniv, R., and O. Souroujon. 2001. “Iterative double clustering for unsupervised and semi-supervised learning.” Pp. 121–132 in *European Conference on Machine Learning (ECML)*.

Erkan, G. 2006. “Language model-based document clustering using random walks.” P. 486 in *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics (NAACL)*.

Erten, C., and M. Sozdinler. 2009. “A Robust Biclustering Method Based on Crossing Minimization in Bipartite Graphs.” P. 439 in *Graph Drawing: 16th International Symposium, GD 2008*.

Ertoz, L., M. Steinbach, and V. Kumar. 2002. “A new shared nearest neighbor clustering algorithm and its applications.” Pp. 105–115 in *Workshop on Clustering High Dimensional Data and its Applications at 2nd SIAM International Conference on Data Mining*.

Ester, M., H. P Kriegel, J. Sander, and X. Xu. 1996. “A density-based algorithm for discovering clusters in large spatial databases with noise.” Pp. 226–231 in *Proc. 2nd International Conference on Knowledge Discovery and Data Mining (KDD)*, vol. 96.

Everitt, B. S., S. Landau, and M. Leese. 2001. *Cluster Analysis. 2001*. 4th ed. Oxford University Press.

Farahat, A. K, and M. S Kamel. 2009. “Document Clustering Using Semantic Kernels Based on Term-Term Correlations.” Pp. 459–464 in *2009 IEEE International Conference on Data Mining Workshops*.

Fayyad, U., and K. Irani. 1993. “Multi-interval discretization of continuous-valued attributes for classification learning.” in *Proc. of the 13th Infternational Joint Conference on Artificial Intelligence (IJCAI)*.

Filippone, M., F. Camastra, F. Masulli, and S. Rovetta. 2008a. “A survey of kernel and spectral methods for clustering.” *Pattern Recognition* 41:176–190.

Filippone, M., F. Camastra, F. Masulli, and S. Rovetta. 2008b. “A survey of kernel and spectral methods for clustering.” *Pattern Recognition* 41:176–190.

Fogel, G., and D. Corne. 2003. *Evolutionary computation in bioinformatics*. Morgan Kaufmann Publications.

Gabrilovich, E., and S. Markovitch. 2005. “Feature generation for text categorization using world knowledge.” in *International Joint Conference on Artificial Intelligence (IJCAI)*.

Gao, B., T. Y Liu, X. Zheng, Q. S Cheng, and W. Y Ma. 2005. “Consistent bipartite graph co-partitioning for star-structured high-order heterogeneous data co-clustering.” P. 50 in *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*.

Gee, K. R. 2003. “Using latent semantic indexing to filter spam.” Pp. 460–464 in *Proceedings of the 2003 ACM symposium on Applied computing*.

Geffet, M., and I. Dagan. 2005. “The distributional inclusion hypotheses and lexical entailment.” Pp. 107–114 in *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics (ACL)*.

Golub, G. H., and J. M. Varah. 1974. “On a characterization of the best l 2-scaling of a matrix.” *SIAM Journal on*

Numerical Analysis 11:472–479.

Golub, T. R. et al. 1999. "Molecular classification of cancer: class discovery and class prediction by gene expression monitoring." *Science* 286:531.

Gonzalo, J., F. Verdejo, I. Chugur, and J. Cigarran. 1998. "Indexing with WordNet synsets can improve text retrieval." in *Proceedings of the ACL/COLING Workshop on Usage of WordNet for Natural Language Processing*.

Gordon, G. J et al. 2002. "Translation of microarray data into clinically relevant cancer diagnostic tests using gene expression ratios in lung cancer and mesothelioma." *Cancer research* 62:4963.

Gotlieb, C. C., and S. Kumar. 1968. "Semantic clustering of index terms." *Journal of the ACM (JACM)* 15:493–513.

Gower, J. C., and G. J. S. Ross. 1969. "Minimum spanning trees and single linkage cluster analysis." *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 18:54–64.

Griffiths, A., L. A Robinson, and P. Willett. 1984. "Hierarchic agglomerative clustering methods for automatic document classification." *Journal of Documentation* 40:175–205.

Guan, J., G. Qiu, and X. Y Xue. 2005. "Spectral images and features co-clustering with application to content-based image retrieval." in *IEEE Workshop on Multimedia Signal Processing*.

Guha, S., R. Rastogi, and K. Shim. 2001. "CURE: An efficient clustering algorithm for large databases." *Information Systems* 26:35–58.

Gundlapalli, R. S. 2005. "Microarray database resource for designing custom microarrays." Doctoral Thesis, University of Louisville.

Hagen, L., and A. B Kahng. 1992. "New spectral methods for ratio cut partitioning and clustering." *IEEE Trans. Computer-Aided Design* 11:1074–1085.

Han, J., and M. Kamber. 2006. *Data mining: concepts and techniques*. Morgan Kaufmann.

Harris, Z. S. 1968. *Mathematical structures of language*. Interscience publishers.

Hartigan, J. A. 1972. "Direct clustering of a data matrix." *Journal of the American Statistical Association* 123–129.

Hastie, T., R. Tibshirani, J. Friedman, and J. Franklin. 2005. "The elements of statistical learning: data mining, inference and prediction." *The Mathematical Intelligencer* 27:83–85.

Herrero, J., A. Valencia, and J. Dopazo. 2001. "A hierarchical unsupervised growing neural network for clustering gene expression patterns." *Bioinformatics* 17:126-135.

Hill, M. O. 1974. "Correspondence analysis: a neglected multivariate method." *Applied Statistics* 23:340–354.

Holloway, A. J, R. K van Laar, R. W Tothill, D. D.L Bowtell, and others. 2002. "Options available—from start to finish—for obtaining data from DNA microarrays II." *nature genetics* 32:481–489.

Homayouni, R., K. Heinrich, L. Wei, and M. W Berry. 2005. "Gene clustering by latent semantic indexing of MEDLINE abstracts." *Bioinformatics* 21:104.

Hotelling, H. 1933. "Analysis of a complex of statistical variables into principal components." *Journal of educational psychology* 24:417–441.

Huber, W., A. Von Heydebreck, and M. Vingron. 2003. "Analysis of microarray gene expression data." *Handbook*

of Statistical Genetics 162–187.

Hussain, S. F, and G. Bisson. 2010. "Text Categorization Using Word Similarities Based on Higher Order Co-occurrences." in *SIAM International Conference on Data Mining (SDM 10)*. Columbus, OH.

Ichino, M., and H. Yaguchi. 1994. "Generalized Minkowski metrics for mixed feature-type data analysis." *IEEE Transactions on Systems, Man, and Cybernetics* 24:698–708.

Ihmels, J., S. Bergmann, and N. Barkai. 2004. "Defining transcription modules using large-scale gene expression data." *Bioinformatics* 20.

Ihmels, J. et al. 2002. "Revealing modular organization in the yeast transcriptional network." *nature genetics* 31:370–378.

Islam, A., and D. Inkpen. 2006. "Second order co-occurrence PMI for determining the semantic similarity of words." Pp. 1033–1038 in *Proceedings of the International Conference on Language Resources and Evaluation, Genoa, Italy*.

Jain, A. K, R. P.W Duin, and J. Mao. 2000. "Statistical pattern recognition: A review." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22:4–37.

Jain, A. K., M. N. Murty, and P. J. Flynn. 1999. "Data clustering: a review." *ACM computing surveys (CSUR)* 31:264-323.

Jarvis, R. A, and E. A Patrick. 1973. "Clustering using a similarity measure based on shared near neighbors." *IEEE Transactions on Computers* 22:1025–1034.

Jeh, G., and J. Widom. 2002. "SimRank: A measure of structural-context similarity." P. 543 in *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Jessup, E. R., and J. H. Martin. 2001. "Taking a new look at the latent semantic analysis approach to information retrieval." *Computational Information Retrieval* 121–144.

Jia, X. et al. 2010. "Local Methods for Estimating SimRank Score." *Proceedings of the 13th International Conference on Extending Database Technology*.

Joachims, T. 1997. "A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization." Pp. 143–151 in *International Conference on Machine Learning (ICML)*.

Joachims, T. 1998. "Text categorization with support vector machines: learning with many relevant." in *10th European Conference on Machine Learning (ECML)*.

Jolliffe, I. T. 2002. *Principal component analysis*. 2nd ed. Springer verlag.

Jurafsky, D., J. H Martin, and A. Kehler. 2000. *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition*. MIT Press.

Karypis, G., E. H Han, and V. Kumar. 1999. "Chameleon: A hierarchical clustering algorithm using dynamic modeling." *IEEE computer* 32:68–75.

Kaufman, L., and P. J Rousseeuw. 1990. "Finding groups in data. An introduction to cluster analysis." *John Wiley and Sons, New York*.

Kim, H., G. H Golub, and H. Park. 2005. "Missing value estimation for DNA microarray gene expression data: local least squares imputation." *Bioinformatics* 21:187-195.

Kirriemuir, J. W, and P. Willett. 1995. “Identification of duplicate and near-duplicate full-text records in database search-outputs using hierarchic cluster analysis.” *PROGRAM-LONDON-ASSOCIATION FOR INFORMATION MANAGEMENT*- 29:241–241.

Klebanov, L., and A. Yakovlev. 2007. “How high is the level of technical noise in microarray data?.” *Biology Direct* 2.

Kleinberg, J. 2003. “An impossibility theorem for clustering.” P. 463 in *Proceedings of the 2002 Conference on Advances in Neural Information Processing Systems*.

Kleinberg, J. M. 1999. “Authoritative sources in a hyperlinked environment.” *Journal of the ACM (JACM)* 46:604–632.

Kluger, Y., R. Basri, J. T Chang, and M. Gerstein. 2003. *Spectral biclustering of microarray data: coclustering genes and conditions*. Cold Spring Harbor Laboratory Press.

Kolb, P. 2009. “Experiments on the difference between semantic similarity and relatedness.” *Proceedings of the Nordic Conference on Computational Linguistics (NODALIDA)* 81-88.

Kontostathis, A., and W. M Pottenger. 2006. “A framework for understanding Latent Semantic Indexing (LSI) performance.” *Information processing and management* 42:56–73.

Kural, S. Y, S. Robertson, and S. Jones. 1999. *Clustering information retrieval search outputs*. 21st European Conference on Information Retrieval (ECIR).

Lance, G. N., and W. T. Williams. 1967. “A general theory of classificatory sorting strategies: 1. Hierarchical systems.” *The computer journal* 9:373-386.

Landauer, T. K, D. S McNamara, S. Dennis, and W. Kintsch. 2007. *Handbook of latent semantic analysis*. Lawrence Erlbaum Associates.

Lander, E. S. et al. 2001. “Initial sequencing and analysis of the human genome.” *Nature Publishing Group* 408:860-921.

Lazzeroni, L., and A. Owen. 2002. “Plaid models for gene expression data.” *Statistica Sinica* 12:61–86.

Lee, D. D, and H. S Seung. 1999. “Learning the parts of objects by non-negative matrix factorization.” *Nature* 401:788–791.

Lemaire, B., and G. Denhière. 2006. “Effects of high-order co-occurrences on word semantic similarities.” *Current Psychology Letters-Behaviour, Brain and Cognition* 18.

Lertnattee, V., and T. Theeramunkong. 2004. “Effect of term distributions on centroid-based text categorization.” *Information Sciences* 158:89–115.

Li, P., H. Liu, J. X Yu, J. He, and X. Du. 2010. “Fast Single-Pair SimRank Computation.” *SIAM Conference on Data Mining (SDM)*.

Li, W. 1992. “Random texts exhibit Zipf's-law-like word frequency distribution.” *IEEE Transactions on Information Theory* 38:1842–1845.

Liu, J., and M. Shah. 2007. “Scene modeling using co-clustering.” in *in the Proc. of International Conference on Computer Vision (ICCV)*.

Liu, J., and W. Wang. 2003. “Op-cluster: Clustering by tendency in high dimensional space.” P. 187 in *Proceedings of the Third IEEE International Conference on Data Mining (ICDM)*.

Liu, N. et al. 2004. “Learning similarity measures in non-orthogonal space.” Pp. 334–341 in *Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management (CIKM)*.

Liu, T., S. Liu, Z. Chen, and W. Ma. 2003. “An evaluation on feature selection for text clustering.” in *International Conference on Machine Learning (ICML)*.

Livesay, K., and C. Burgess. 1998. “Mediated priming in high-dimensional semantic space: No effect of direct semantic relationships or co-occurrence.” *Brain and Cognition* 37:102–105.

Livne, O. E, and G. H Golub. 2004. “Scaling by binormalization.” *Numerical Algorithms* 35:97–120.

Lizza, M., and F. Sartoretto. 2001. “A comparative analysis of LSI strategies.” *SIAM Workshop on Computational information retrieval* 171–181.

Long, B., X. Wu, Z. M Zhang, and P. S Yu. 2006. “Unsupervised learning on k-partite graphs.” Pp. 317–326 in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*.

Long, B., Z. M Zhang, and P. S Yu. 2005. “Co-clustering by block value decomposition.” Pp. 635–640 in *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*.

von Luxburg, U. 2007. “A tutorial on spectral clustering.” *Statistics and Computing* 17:395–416.

MacQueen, J. B. 1966. “Some methods for classification and analysis of multivariate observations.” *Proceedings of the fifth Berkeley Symposium on Mathematical Statistics and Probability* 1:281-297.

Macskassy, S. A, A. Banerjee, B. D Davison, and H. Hirsh. 1998. “Human performance on clustering web pages: A preliminary study.” Pp. 264–268 in *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*.

Madeira, S. C, and A. L Oliveira. 2004. “Biclustering algorithms for biological data analysis: a survey.” *IEEE Transactions on computational Biology and Bioinformatics* 1:24–45.

Mäkinen, E., and M. Sieranta. 1994. “Genetic algorithms for drawing bipartite graphs.” *International Journal of Computer Mathematics* 53:157–166.

Manning, C. D, P. Raghavan, and H. Schütze. 2008. “An Introduction to Information Retrieval.” *Cambridge, UK: Cambridge University Press.* 11:2009.

Manning, C. D, and H. Schütze. 1999. *Foundations of Natural Language Processing*. MIT Press.

Martin, D. I., and M.W. Berry. 2007. “Mathematical Foundations Behind Latent Semantic Analysis.” *Handbook of Latent Semantic Analysis*.

May, M., and K. Szkatula. 1988. “On the bipartite crossing number.” *Control and Cybernetics* 17:85–98.

Mladenic, D. 1998. “Feature subset selection in text learning.” Pp. 95–100 in *European Conference on Machine Learning (ECML)*.

Mount, D. W. 2004. *Bioinformatics: sequence and genome analysis*. CSHL press.

Munkres, J. 1957. “Algorithms for the assignment and transportation problems.” *Journal of the Society for Industrial and Applied Mathematics* 5:32–38.

Murali, T. M., and S. Kasif. 2002. “Extracting conserved gene expression motifs from gene expression data.” P. 77

in *Biocomputing 2003: Proceedings of the Pacific Symposium Hawaii, USA 3-7 January 2003*.

Murtagh, F. 1983. "A survey of recent advances in hierarchical clustering algorithms." *The Computer Journal* 26:354359.

Ng, R. T, and J. Han. 1994. "Efficient and effective clustering methods for spatial data mining." Pp. 144–144 in *Proceedings of the International Conference on Very Large Data Bases*.

Nigam, K., A. K McCallum, S. Thrun, and T. Mitchell. 2000. "Text classification from labeled and unlabeled documents using EM." *Journal of Machine Learning* 39:103–134.

Oba, S. et al. 2003. "A Bayesian missing value estimation method for gene expression profile data." *Bioinformatics* 19:2088.

Oja, E., A. Hyvarinen, and J. Karhunen. 2001. *Independent Component Analysis*. John Wiley & Sons.

Olson, C. F. 1995. "Parallel algorithms for hierarchical clustering." *Parallel Computing* 21:1313–1325.

Ozawa, K. 1985. "A stratificational overlapping cluster scheme." *Pattern Recognition* 18:279–286.

Parsons, L., E. Haque, and H. Liu. 2004. "Subspace clustering for high dimensional data: a review." *ACM SIGKDD Explorations Newsletter* 6:90–105.

Pennisi, E. 2007. "GENOMICS: DNA Study Forces Rethink of What It Means to Be a Gene." *Science* 316:1556-1562.

Pereira, Fernando, Naftali Tishby, and Lillian Lee. 1993. "Distributional Clustering Of English Words." Pp. 183--190 in *Proceedings of the 31ST Annual Meeting of the Association for Computational Linguistics (ACL)*. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.14.1729.

Ponte, J. M, and W. B Croft. 1998. "A language modeling approach to information retrieval." Pp. 275–281 in *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*.

Porter, M. F. 1997. "An algorithm for suffix stripping." *Program* 14:130-137.

Prelic, A. et al. 2006. "A systematic comparison and evaluation of biclustering methods for gene expression data." *Bioinformatics* 22:1122–1129.

Qamar, A. M, and E. Gaussier. 2009. "Online and Batch Learning of Generalized Cosine Similarities." Pp. 926–931 in *Proceedings of the 2009 Ninth IEEE International Conference on Data Mining*.

Qiu, G. 2004. "Image and feature co-clustering." Pp. 991–994 in *In the Proc. of the International Conference on Pattern Recognition (ICPR)*.

Quackenbush, J. 2002. "Microarray data normalization and transformation." *nature genetics* 32:496–501.

R. A. Fisher. 1922. "On the interpretation of chi-square from the contingency table and the calculation of p." *Journal of Royal Statistical Society* 85:87-94.

Rege, M., M. Dong, and F. Fotouhi. 2008. "Bipartite isoperimetric graph partitioning for data co-clustering." *Data Mining and Knowledge Discovery* 16:276–312.

Ringnér, M. 2008. "What is principal component analysis?." *Nature Biotechnology* 26:303.

Rohwer, R., and D. Freitag. 2004. "Towards full automation of lexicon construction." Pp. 9–16 in *Proceedings of*

the HLT-NAACL Workshop on Computational Lexical Semantics.

Saenger, W., and C.R. Cantor. 1984. *Principles of Nucleic Acid Structure*. Springer Verlag: New York.

Sahlgren, M. 2001. "Vector-based semantic analysis: Representing word meanings based on random labels." in *ESSLI Workshop on Semantic Knowledge Acquistion and Categorization*.

Sakkis, G. et al. 2003. "A memory-based approach to anti-spam filtering for mailing lists." *Information Retrieval* 6:49–73.

Salton, G. 1971. *The SMART Retrieval System---Experiments in Automatic Document Processing*. Prentice-Hall, Inc. http://portal.acm.org/citation.cfm?id=1102022 (Accessed November 20, 2009).

Salton, G. 1983. *Introduction to modern information retrieval*. McGraw-Hill New York.

Salton, G., and C. Buckley. 1988. "Term-weighting approaches in automatic text retrieval." *Information processing & management* 24:513–523.

Salton, G., and M. E. Lesk. 1968. "Computer evaluation of indexing and text processing." *Journal of the ACM (JACM)* 15:8–36.

Salton, G., A. Wong, and C. S. Yang. 1975. "A vector space model for automatic indexing." *Communications of the ACM* 18:613-620.

Schütze, H. 1998. "Automatic word sense discrimination." *Computational Linguistics* 24:97–123.

Schütze, H., and C. Silverstein. 1997. "Projections for efficient document clustering." Pp. 74–81 in *Proceedings of the ACM Special Interest Group on Information Retrieval (SIGIR)*, vol. 31.

Sebastiani, F. 2002. "Machine learning in automated text categorization." *ACM computing surveys (CSUR)* 34:1–47.

Sedding, J., and D. Kazakov. 2004. "Wordnet-based text document clustering." Pp. 104-113 in *Proceedings of the 3rd Workshop on Robust Methods in Analysis of Natural Language Data*.

Segal, E., A. Battle, and D. Koller. 2002. "Decomposing gene expression into cellular processes." P. 89 in *Biocomputing 2003: Proceedings of the Pacific Symposium Hawaii, USA 3-7 January 2003*.

Segal, E., B. Taskar, A. Gasch, N. Friedman, and D. Koller. 2001. "Rich probabilistic models for gene expression." *BIOINFORMATICS-OXFORD-* 17:243–252.

Seneta, E. 2006. *Non-negative matrices and Markov chains*. Springer-Verlag New York Inc.

Shafiei, M., and E. Milios. 2006. "Model-based overlapping co-clustering." in *Proceedings of the Fourth Workshop on Text Mining, Sixth SIAM International Conference on Data Mining (SDM)*.

Sheng, Q., Y. Moreau, and B. De Moor. 2003. *Biclustering microarray data by Gibbs sampling*. Oxford Univ Press.

Shi, J., and J. Malik. 2000. "Normalized cuts and image segmentation." *IEEE Transactions on pattern analysis and machine intelligence* 22:888–905.

Slonim, N., N. Friedman, and N. Tishby. 2002. "Unsupervised document classification using sequential information maximization." Pp. 129–136 in *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*.

Slonim, N., and N. Tishby. 2000. "Document clustering using word clusters via the information bottleneck method." Pp. 208–215 in *Proceedings of the 23rd annual international ACM SIGIR conference on Research and*

development in information retrieval.

Slonim, N., and N. Tishby. 2001. "The power of word clusters for text classification." in *Proceedings of ECIR-01, 23rd European Colloquium on Information Retrieval Research*.

Sneath, P. H.A, and R. R Sokal. 1973. *Numerical taxonomy: the principles and practice of numerical classification*. WH Freeman San Francisco.

Stekel, D. 2003. *Microarray bioinformatics*. Cambridge Univ Pr.

Strehl, A. 2002. "Relationship-based clustering and cluster ensembles for high-dimensional data mining." *PhD thesis, The University of Texas at Austin, May 2002.*.

Strehl, A., J. Ghosh, and R. Mooney. 2000. "Impact of similarity measures on web-page clustering." Pp. 58–64 in *Proc. AAAI Workshop on AI for Web Search (AAAI 2000), Austin*.

Sugiyama, K., S. Tagawa, and M. Toda. 1981. "Methods for visual understanding of hierarchical system structures." *IEEE Transactions on System, Man, and Cybernetics* 11:109–125.

Sun, X., and A. Nobel. 2006. "Significance and recovery of block structures in binary matrices with noise." *Lecture Notes in Computer Science* 4005:109.

Takamura, H., and Y. Matsumoto. 2002. "Two-dimensional clustering for text categorization." in *International Conference On Computational Linguistics*.

Tamayo, P. et al. 1999. "Interpreting patterns of gene expression with self-organizing maps: methods and application to hematopoietic differentiation." *Proceedings of the National Academy of Sciences of the United States of America* 96:2907-2914.

Tanay, A., R. Sharan, and R. Shamir. 2005. "Biclustering algorithms: A survey." *Handbook of computational molecular biology* 9:26–1.

Tanay, A., R. Sharan, and R. Shamir. 2002. "Discovering statistically significant biclusters in gene expression data." *Bioinformatics* 18:136–144.

Tanay, A., R. Sharon, and R. Shamir. 2002. "Biclustering gene expression data." in *Proceedings of International Conference on Intelligent Systems for Molecular Biology (ISMB)*.

Tibshirani, R. et al. 1999. "Clustering methods for the analysis of DNA microarray data." *Dept. Statist., Stanford Univ., Stanford, CA, Tech. Rep*.

Tishby, N., F. C Pereira, and W. Bialek. 2000. "The information bottleneck method." Pp. 367-377 in *Proceedings of 37th Annual Allerton Conference on Communication, Control and Computing*.

Turney, P. D. 2008. "A uniform approach to analogies, synonyms, antonyms, and associations." Pp. 905–912 in *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*.

Van Rijsbergen, C. J. 1979. "Information retrieval, chapter 7." *Butterworths, London* 2:111–143.

Venter, J. C et al. 2001. "The sequence of the human genome." *science* 291:1304-1351.

Wagner, D., and F. Wagner. 1993. "Between min cut and graph bisection." *Mathematical Foundations of Computer Science* 744–750.

Wang, X., A. Li, Z. Jiang, and H. Feng. 2006. "Missing value estimation for DNA microarray gene expression data by Support Vector Regression imputation and orthogonal coding scheme." *BMC bioinformatics* 7:32.

Ward Jr, J. H. 1963. “Hierarchical grouping to optimize an objective function.” *Journal of the American statistical association* 58:236–244.

Weinberger, K. Q, and L. K Saul. 2009. “Distance metric learning for large margin nearest neighbor classification.” *The Journal of Machine Learning Research* 10:207–244.

Willett, P. 1981. “A Fast Procedure for the Calculation of Similarity Coefficients in Automatic Classification..” *Information Processing and Management* 17:53–60.

Willett, P. 1983. “Similarity coefficients and weighting functions for automatic document classification: an empirical comparison.” *International Classification* 10:138–142.

Wishart, D. 1969. “An algorithm for hierarchical classifications.” *Biometrics* 25:165–170.

Wouters, L., L. Bijnens, S. U Kass, G. Molenberghs, and P. J Lewi. 2003. “Graphical exploration of gene expression data: a comparative study of three multivariate methods.” *Biometrics* 59:1131–1139.

Xu, J., and W. B Croft. 1998. “Corpus-based stemming using cooccurrence of word variants.” *ACM Transactions on Information Systems (TOIS)* 16:61–81.

Xu, R., and D. Wunsch. 2005. “Survey of clustering algorithms.” *IEEE Transactions on neural networks* 16:645–678.

Xu, W., X. Liu, and Y. Gong. 2003. “Document clustering based on non-negative matrix factorization.” Pp. 267–273 in *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*.

Yang, J. et al. 2003. “Enhanced biclustering of expression data.” in *Proceedings of IEEE Symposium on Bioinformatics and Bioengineering (BIBE)*.

Yang, M. S. 1993. “A survey of fuzzy clustering.” *Mathematical and computer modelling* 18:1–16.

Yang, Y., and X. Liu. 1999. “A re-examination of text categorization methods.” Pp. 42–49 in *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*.

Yang, Y., and J. O Pedersen. 1997. “A comparative study on feature selection in text categorization.” Pp. 412–420 in *Proceedings of the International Conference on Machine Learning (ICML)*.

Yang, Y. H, M. J Buckley, S. Dudoit, and T. P Speed. 2002. “Comparison of methods for image analysis on cDNA microarray data.” *Journal of computational and graphical statistics* 11:108–136.

Yang, Yiming. 1994. “Expert network: Effective and efficient learning from human decisions in text categorization and retrieval.” Pp. 13-22 in *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*.

Yen, L., F. Fouss, C. Decaestecker, P. Francq, and M. Saerens. 2009. “Graph nodes clustering with the sigmoid commute-time kernel: A comparative study.” *Data & Knowledge Engineering* 68:338–361.

Zahn, C. T. 1971. “Graph-theoretical methods for detecting and describing gestalt clusters.” *IEEE Transactions on computers* 20:68–86.

Zelikovitz, S., and H. Hirsh. 2001. “Using LSI for text classification in the presence of background text.” Pp. 113–118 in *Proceedings of the tenth international conference on Information and knowledge management*.

Zha, H., and X. Ji. 2002. “Correlating multilingual documents via bipartite graph modeling.” P. 444 in *Proceedings*

of the 25th annual international ACM SIGIR conference on Research and development in information retrieval.

Zhang, J. 2007. "Co-clustering by similarity refinement." Pp. 381–386 in *Sixth International Conference on Machine Learning and Applications, (ICMLA)*.

Zhang, T., R. Ramakrishnan, and M. Livny. 1996. "BIRCH: an efficient data clustering method for very large databases." *ACM Special Interest Group on Managment of Data (SIGMOD)* 25:103–114.

Zhang, Z., J. Zhang, and H. Xue. 2008. "Improved K-means clustering algorithm." Pp. 169–172 in *Proceedings of the 2008 Congress on Image and Signal Processing, Vol. 5-Volume 05*.

Zhong, H., J. Shi, and M. Visontai. 2004. "Detecting unusual activity in video." in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

Zhou, X., X. Zhang, and X. Hu. 2007. "Semantic smoothing of document models for agglomerative clustering." Pp. 6–12 in *Proceedings of the International Joint Conference on Artifical Intelligence (IJCAI)*.

Zhu, S., T. Li, and M. Ogihara. 2002. "CoFD: An algorithm for non-distance based clustering in high dimensional spaces." Pp. 52–62 in *Proceedings of the 4th International Conference on Data Warehousing and Knowledge Discovery (DaWaK)*.

Zipf, G. 1949. "Human Behavior and the principle of least effort." *Cambridge, Mass., Addision-Wesley*.

Abstract

Clustering is the unsupervised classification of patterns (observations, data items, or feature vectors) into groups (clusters). The clustering problem has been addressed in many contexts and there exist a multitude of different clustering algorithms for different settings. As datasets become larger and more varied, adaptations of existing algorithms are required to maintain the quality of clusters. In this regard, high-dimensional data poses some problems for traditional clustering algorithms known as '*the curse of dimensionality*'.

This thesis proposes a co-similarity based algorithm that is based on the concept of distributional semantics using higher-order co-occurrences, which are extracted from the given data. As opposed to co-clustering, where both instance and feature sets are hard clustered, co-similarity may be defined as a more '*soft*' approach. The output of the algorithm is two similarity matrices – one for the objects and one for their features. Each of these similarity matrices exploits the similarity of the other, thereby implicitly taking advantage of a co-clustering style approach. Hence, with our method, it becomes possible to use any classical clustering method (*k*-means, Hierarchical clustering …) to co-cluster data.

We explore two applications of our co-similarity measure. In the case of text mining, document similarity is calculated based on word similarity, which in turn is calculated on the basis of document similarity. In this way, not only do we capture the similarity between documents coming from their common words but also the similarity coming from words that are not directly shared by the two documents but that can be considered to be similar. The second application is on gene expression datasets and is an example of co-clustering. We use our proposed method to extract gene clusters that show similar expression levels under a given condition from several cancer datasets (colon cancer, lung cancer, etc).

The approach can also be extended to incorporate prior knowledge from a training dataset for the task of text categorization. Prior category labels coming from data in the training set can be used to influence similarity measures between features (words) to better classify incoming test datasets among the different categories. Thus, the same framework can be used for both clustering and categorization task depending on the amount of prior information available.

Keywords: *Clustering, co-clustering, supervised learning, text mining, co-similarity, structural similarity*

Resume

La classification de données (ou apprentissage non-supervisé) vise à regrouper un ensemble d'observations sous la forme de classes homogènes et contrastées. Lorsque les données sont caractérisées par un grand nombre de variables, il devient nécessaire d'adapter les méthodes classiques, notamment au niveau des métriques, afin de maintenir des classes pertinentes ; ce phénomène est connu sous le nom de "malédiction de la dimension".

Dans cette thèse, nous proposons une mesure de co-similarité basée sur la notion de co-occurrences d'ordre supérieur, directement extraites à partir des données. Dans le cas de l'analyse de texte, par exemple, les similarités entre documents sont calculées en prenant en compte les similarités entre mots, qui simultanément prennent en compte les similarités entre documents. Par cette approche « circulaire », nous parvenons à mettre en correspondance des documents sans mots communs mais ayant juste des mots similaires. Cette approche s'effectue de manière purement numérique sans nécessiter de thesaurus externe.

En outre, notre méthode peut également être étendue pour tirer parti de connaissances "a priori" afin de réaliser des tâches de catégorisation de textes : l'étiquette des documents est utilisée pour influencer les mesures de similarité entre les mots afin de classer de nouvelles données. Ainsi, le même cadre conceptuel, exprimable en terme de théorie des graphes, peut être utilisé à la fois pour les tâches de classification et de catégorisation en fonction de la quantité d'information initiale.

Nos résultats montrent une amélioration significative de la précision, par rapport à l'état de l'art, à la fois pour le co-clustering et la catégorisation sur les jeux de données qui ont été testés.

Mots-Clés: *Co-similarité, co-classification, systeme d'apprentissage, fouille de texts, expression génique, co-clustering.*