Evaluating Clusterings by Estimating Clarity

by

John Samuel Whissell

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2012

Abstract

In this thesis I examine clustering evaluation, with a subfocus on text clusterings specifically. The principal work of this thesis is the development, analysis, and testing of a new internal clustering quality measure called informativeness.

I begin by reviewing clustering in general. I then review current clustering quality measures, accompanying this with an in-depth discussion of many of the important properties one needs to understand about such measures. This is followed by extensive document clustering experiments that show problems with standard clustering evaluation practices.

I then develop informativeness, my new internal clustering quality measure for estimating the clarity of clusterings. I show that informativeness, which uses classification accuracy as a proxy for human assessment of clusterings, is both theoretically sensible and works empirically. I present a generalization of informativeness that leverages external clustering quality measures. I also show its use in a realistic application: email spam filtering. I show that informativeness can be used to select clusterings which lead to superior spam filters when few true labels are available.

I conclude this thesis with a discussion of clustering evaluation in general, informativeness, and the directions I believe clustering evaluation research should take in the future.

Acknowledgements

I would like to thank my supervisor, Dr. Charlie Clarke, for providing guidance in this work, and for helping me to become a better researcher in general. I would also like to thank Dr. Chyrsanne DiMarco for being there at the beginning of this work. Finally, I would like to thank all my friends and family for the numerous discussions we had relating to this work, and more importantly, for keeping me sane while perusing it, and other things, over the years.

Dedication

This thesis is dedicated to my daughter Melissa. She reminds me every day, just by being herself, that richness does not come from coins, and that meaning does not have to come from moving mountains.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In an widely cited clustering survey Jain et al. [88] defined clustering as *the unsupervised classification of patterns (observations, data items, or feature vectors) into groups (clusters)*. This simplistic definition, while accurate in a technical sense, belies not only the inherent complexity of clustering, but the staggering array of applications it has. The contributions I present to the expansive field of clustering research that are contained within this thesis are:

1. A survey of clustering. This includes discussions on clustering outputs, features, similarity/distance measures, and many varieties of clustering algorithms.
2. A survey of common clustering quality measures, both internal and external.
3. An in-depth discussion of many important properties of clustering quality measures and how common clustering quality measures behave with respect to them.
4. An extensive document clustering experiment that investigates feature weighting and evaluation in document clustering.

5. The design and validation of a new internal clustering quality measure that is aimed at estimating the *clarity* of a clustering. This is the primary work of this thesis, and is discussed below. I also present a generalization of my new internal clustering quality measure that leverages external clustering quality measures.

6. The application of the new clustering quality measure to a real problem domain: email spam filtering.

Considering clusterings to be on independent and identically distributed samples of populations, I define the clarity of a clustering to be how well a human expert in the data type of its population can assign previously unseen members of that population to the most appropriate cluster in the clustering (e.g., botanists can judge the clarity of clusterings of plants, computer scientists can judge the clarity of clusterings of computer science journal publications). The estimation of clarity that I design is fully automatic and aids in the detection of meaningful and useful clusterings and is highly general.

The subject of evaluating clusterings, sometimes referred to as *cluster analysis* or *cluster validation*, has quite possibly received as much attention as the design of clustering algorithms themselves, and rightly so. Electronically accessible datasets (such as Web pages, news articles, consumer marketing data, protein sequences/structures, and geological information) are growing in both number and size. This growth has been accompanied by an increased need for efficient and effective machine learning algorithms that can make use of the datasets in important tasks such as Web search, document routing, directed advertising, and protein structure prediction. In order to make use of clustering in these tasks, and others, it is necessary to be able to identity when a clustering is good; thus the subject of how to evaluate clusterings is receiving ever increasing attention.

It is generally accepted that the ideal way to evaluate a clustering is situational human assessment. If a clustering helps with whatever an individual wants to use it for, then it is a good clustering, for them. In typical clustering literature, this assessment is provided indirectly via comparing a clustering to a gold standard generated by a human. Judging clusterings situationally with humans is sensible as it focuses on a clustering's actual uses as opposed to abstract notions of clustering goodness, but it is often not practical for various reasons. Among its notable issues are inconsistent evaluation (both between people and by just a single individual); uncertainty of what is actually sought (as in exploratory data analysis); and infeasible amounts of time being required to select good clusterings from among many. Problems like these have resulted in the profusion of internal clustering quality measures, i.e., ones that use no human input about what is expected, hereafter referred to as *ICQM*s, being used to aid people in selecting good clusterings.

ICQMs have their own issues. They usually focus on specific mathematical notions of what makes a clustering good (within-cluster scatter, the margin between clusters, etc.). Such notions, while allowing the measures to be computed easily, have been found lacking in terms of generality; one can always find clusterings, applications, and/or datasets for which a particular ICQM is not appropriate. For example, consider Fig. 1. With respect to assessing the true clustering of each dataset as high quality; within-cluster scatter is a useful measure for datasets (a) and (b) only; a margin between clusters is most useful for (a) and (c); while (d) requires some path-based measure in order to handle the small bridge between the clusters.

Recent theoretical works on clustering evaluation have outlined disparities between IC-QMs, highlighting further problems with them. For example, Ackerman and Ben-David [2] show that some commonly accepted notions of clustering quality are in fact pairwise in-

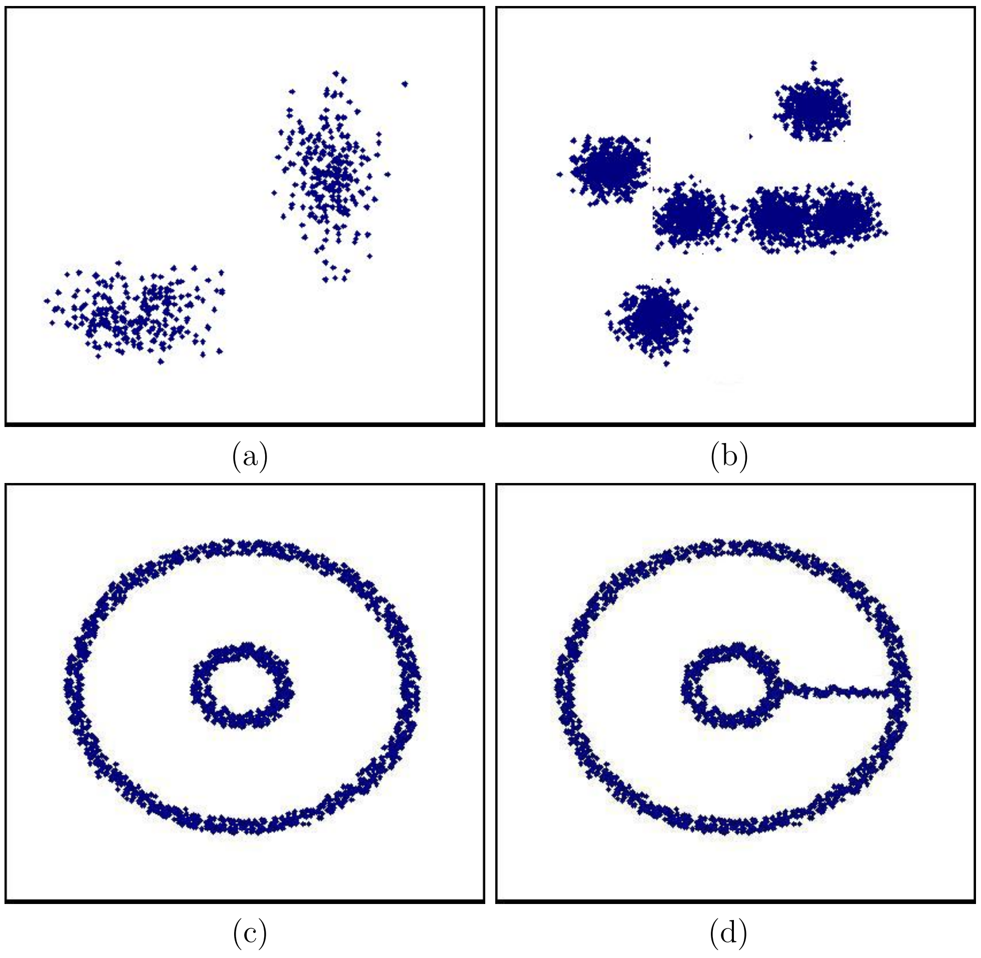


Figure 1.1: Datsets containing various structures. (a) is two Gaussians. (b) is 6 unevenly spaced Gaussians. (c) is two ring clusters, and (d) is two ring clusters joined by a bridge.

consistent despite seeming similar. The arbitrary number of clustering structures possible, combined with the lack of agreement between ICQMs on what an ideal clustering is, makes one highly skeptical as to if a single ICQM for assessing universal clustering goodness can exist.

Given the above, I do not attempt to design an ICQM that measures universal clustering goodness in this thesis; rather I design an ICQM for estimating the clarity of clustering, where clarity is as I have defined previously. I believe the major appeal of clarity is its generality; I argue that, from its definition, it is something we want every clustering to have, independent of the particular dataset it was made on or its intended application. This is in contrast to other clustering quality notions, which are often abstract and may or may not be useful on a particular clustering, application, and/or dataset.

The ICQM I design for estimating clustering clarity, which I call *informativeness*, leverages classification. I will show empirically that informativeness is more robust than several other common ICQMs. An example of an application of informativeness that I will discuss in this thesis is the design of an email spam filter that uses as few human labels as possible while simultaneously being as effective as possible. I will show that using informativeness to select which among many potential clusterings to use in training a spam filter in this context leads to high quality spam filters.

The rest of this thesis is organized as follows. I begin by presenting a survey of clustering in Chapter 2. While covering things from a general perspective in the survey, I will also note how the topics discussed pertain to text specifically. Clustering output structures, similarity/dissimilarity measures, features, and algorithms are discussed in the chapter.

Chapter 3 focuses on current clustering quality measures. I divide clustering quality measures into internal clustering quality measures (ICQMs) and external clustering quality measures (ECQMs). I present a large variety of both categories of quality measures.

In Chapter 4, I investigate properties of clustering quality measures. I discuss the majority of the important properties identified in previous work and some refinements of these properties that I design here. In addition, I investigate some less discussed/formalized aspects of clustering quality measures.

In Chapter 5, I present extensive document clustering experiments. I show that while some clustering algorithms/feature weightings are better for document clustering on average, there is a large amount of disagreement between how ECQMs rank documents clusterings, highlighting the disparity between what clustering quality measures are in fact measuring.

Chapter 6 deals with informativeness, my new ICQM for estimating clustering clarity. I use classification as the basis for estimating clarity in informativeness. I will discuss the merits of using informativeness in terms of the ICQM properties discussed in Chapter 4. I present a generalization of informativeness. I also present experiments on synthetic and real datasets that show that informativeness and some implementations of its generalized form are a robust ICQMs—they can detect when good clusterings have been found for a wide variety of datasets.

In Chapter 7 I present informativeness in a realistic application: email spam filtering. More specifically, I look at using clustering to minimize the amount of user labelings required to produce an effective email spam filter. I show that using informativeness to select which clustering to use in the spam filter training process I present is superior to using other ICQMs in the same process, as well as being superior to simply training a spam filter on random emails.

Chapter 8 is my conclusion and discussion of future work.

Chapter 2

Clustering Survey

The survey I present here covers things from a general perspective as well as with respect to text clustering specifically. I begin by presenting the various structures clustering algorithms produce along with their associated terminology. Dataset representations in clustering, along with feature selection, weighting, and dimensionality reduction are discussed in Section 2.2. Section 2.3 discusses the similarity/distance measures typically used by clustering algorithms (Euclidean distance, cosine, KL-Divergence, etc.).

Descriptions of specific clustering algorithms are given in Section 2.4. The algorithms are organized by the broad concept they use for clustering (center-based, linkage, etc.). Due to the sheer number of clustering algorithms, I cannot discuss every clustering algorithm (or even most). Instead, for each concept class, I cover some of its members that are more notable and/or have been applied to text clustering specifically. It should be noted that my categorization system of clustering algorithms presented in Section 2.4 is not the only means of organizing clustering algorithms. For example, one might organize clustering algorithms by their output as discussed in Section 2.1.1), or by the axioms that are assured by each clustering algorithm. Chapter 4 discusses many axioms, but they are for clustering quality measures and not clustering algorithms; for clustering algorithm axioms readers should consult works such as the one by Kleinberg [95]).

2.1 Clustering Structures

One of the principal properties of a clustering algorithm is the structure of the clustering it outputs. Depending on the intended function of a clustering, a user may want the clustering in different forms. For example, a tree-like clustering, such as that supplied by DMOZ[1] for Web pages, might enable easy browsing and interpretation by human users, or a single partition of a dataset might be required so one can create actual physical groups of objects, etc. In the following sections I describe the various structures clustering algorithms create along with their respective terminology. It should be noted that some literature surveys of clustering algorithms organize their algorithms by their output structures (as Jain et al. [88] do partly). This is a popular choice as output structures are easily understandable to those both familiar and unfamiliar with clustering research. However, such an organization does not accurately reflect the different concepts that algorithms use while clustering, so instead I discuss output structures here and organize individual algorithms later on by concept.

2.1.1 Hard versus Soft Clustering

Let X be a dataset, and x_i be the ith object of that dataset. A *hard clustering algorithm* C takes an X and zero or more additional parameters, and produces a set of subsets of X. I denote the application of C to X as $C(X, *)$, where $*$ represents the additional parameters

[1] `www.dmoz.org`

(separated by commas), and define $C(X,*) = (c_1, \ldots, c_k)$, $\forall_{c_i \in C(X,*)} c_i \subseteq X$. $C(X,*)$ is a *hard clustering* of X (typically referred to as a *clustering* unless there is a chance of confusion with other forms of clustering), and each $c_i \in C(X,*)$ is referred to as a *cluster*. A *k-clustering* (or *k-way clustering*) of X is any clustering of X with k clusters. A cluster with only one object in it is referred to as a *singleton*. If, for a given C, $\bigcap_{C(X,*)} = \phi$ for any X and $*$, C is also said to be a *disjoint* clustering algorithm—one that produces clusters with no overlapping membership. Fig. 2.1 gives an example of hard disjoint and non-disjoint clusterings of a dataset. For a given $C(X,*)$, an $x_i \in X$ that is not a member of a cluster in $C(X,*)$ is referred to as an *outlier* of C.

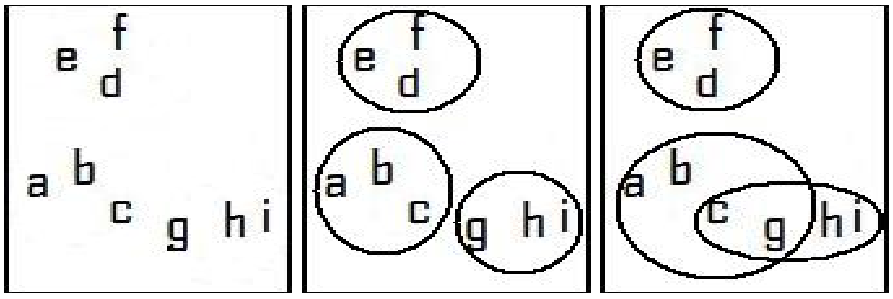


Figure 2.1: (a) is a toy dataset of 9 points. (b) is a hard and disjoint clustering of (a). (c) is a hard but non-disjoint clustering (a).

Table 2.1: Various clusterings of the dataset from Fig. 2.1(a).

Hard/Disjoint	**Hard**	**Soft**
$c_1 = (a, b, c)$	$c_1 = (a, b, c, g)$	$c_1 = ((a, 0.9), (b, 0.85), (c, 0.55), (d, 0.2), (e, 0.1), (f, 0.05), (g, 0.4), (h, 0.2), (i, 0.1))$
$c_2 = (d, e, f)$	$c_2 = (d, e, f)$	$c_2 = ((a, 0.025), (b, 0.05), (c, 0.05), (d, 0.775), (e, 0.875), (f, 0.925), (g, 0.05), (h, 0.05), (i, 0.05))$
$c_3 = (g, h, i)$	$c_3 = (c, g, h, i)$	$c_3 = ((a, 0.075), (b, 0.1), (c, 0.4), (d, 0.025), (e, 0.025), (f, 0.025), (g, 0.55), (h, 0.75), (i, 0.85))$

A *soft clustering algorithm* C takes an X and zero or more additional parameters, and produces a set of sets $(c_1, \ldots, c_k)$. I denote the application of C in a manner identical to that of hard clustering algorithms, with $C(X,*)$ being referred to as a *soft (probabilistic)* *clustering.* As with hard clustering algorithms, each $c_i \in C(X, *)$ is referred to as a cluster. In this case, though, each c_i is a set of tuples $(x_j, p(x_j))$, where $x_j \in X$ and $p(x_j)$ is a measure of how strongly x_j belongs to the cluster c_i. Table 2.1 gives an example of hard, hard/disjoint, and soft clusterings of the dataset in Fig. 2.1. The definition of an outlier for a soft clustering is somewhat ambiguous and may be arbitrarily defined. A soft clustering may be *hardened* by assigning each x_j exclusively to the c_i that has a maximum $p(x_j)$, resulting in a hard clustering (for example, in Table 2.1 hardening the soft clustering produces the hard/disjoint clustering). Hardening is typically done to facilitate the use of non-soft clustering quality measures, to compare a set of clusterings where some are soft and others are hard, and also when applications require hard clusterings.

Often the clustering resulting from the application of a soft clustering algorithm is a *mixture model*—a set of distributions where each cluster is one distribution in the mixture. Together these distributions are taken to have created the dataset. In such a situation, $p(x_j)$ for cluster c_i is the probability that c_i generated x_j. This probability is derived from properties of the distribution tied to c_i.

Unfortunately, it is somewhat common in clustering to erroneously refer to many forms of clustering output as partitional. Because of this, some works use the term partitional with quantifiers. A *hard*, or *crisp* partition refers to a clustering that is a true partition in the mathematical sense (hard, disjoint), while a *soft* or *fuzzy* partition may refer to any of the other partition-like structures one may obtain.

2.1.2 Hierarchical Clustering

A hard clustering algorithm (or much less commonly, a soft clustering algorithm) may be *hierarchical* in nature. In such a case, $C(X, *)$ creates a recursive tree-like structure where each cluster contains a set of clusters itself, with each cluster being a subset of its containing cluster. Fig. 2.2 gives an example of a hierarchical clustering.

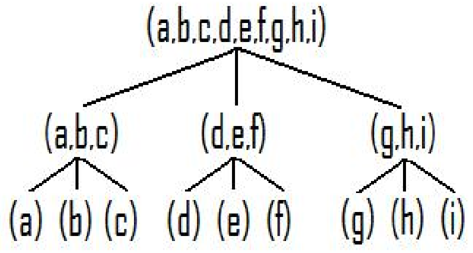


Figure 2.2: A hierarchical clustering of the dataset from Fig. 2.1(a).

Hierarchical clustering algorithms (and clusterings) have their own terms associated with them. The tree-like structure of clusters produced (as in Fig. 2.2) is referred to as a *dendrogram*. The terms *parent*, *child*, *siblings*, *descendent*, and *ascendent* have their typical tree meanings for such dendrograms. More than one root may be present in the dendrogram. Likewise, the leaves of the dendrogram need not be singleton clusters. The *arity* (i.e., number of possible children per cluster) in the dendrogram is variable and based on the individual clustering algorithm used to generate it, although it is overwhelmingly common for such dendrograms to have binary arity. The large majority of hierarchical clustering algorithms ensure the following: 1) children are subsets of their parents; 2) the union of all the children for a given cluster is equal to the cluster itself; and 3) all siblings have empty intersections. These properties allow one to create true partitionings of a dataset by cutting off all the descendants of any number of clusters in the dendrogram and taking all the remaining leaves of the dendrogram as a partitional clustering. For example, in Fig. 2.2 we may cut the children of (a, b, c) and (d, e, f) off, giving us a partitional clustering (represented by the leaves) of $((a, b, c), (d, e, f), (g), (h), (i))$. Unless noted otherwise, when I refer to a specific clustering algorithm as hierarchial, it may be taken to have the three properties mentioned above.

Hierarchical clustering algorithms typically produce their dendrograms in one of two ways: *bottom-up* or *top-down*. A bottom-up algorithm, also referred to as an *agglomerative* algorithm, usually begins with singletons clusters and merges them into progressively larger clusters [141, 88, etc.]. A top-down, or *divisive* algorithm, usually begins with everything in a single cluster and splits the clusters iteratively [125, 145, etc.]. Sometimes bottom-up and top-down algorithms begin with preclustered data. This kind of approach is used in Zhao and Karypis' constrained agglomerative clustering algorithms [172] and by Fung et al. [59]. Rarely, a hierarchical clustering algorithm may have both bottom-up and top-down aspects to it.

2.1.3 Generative versus Discriminative Clustering

Clustering algorithms may be *generative* or *discriminative*. A generative clustering algorithm produces a clustering that is a mixture model. Mixture models, as discussed previously, describe how datasets might have been generated. A discriminative clustering algorithm is one that simply tells you which points are or are not in the same cluster.

Mixture model clusterings may be converted to discriminative clusterings via hardening. Among their benefits are the ability to determine membership of previously unseen objects in a principled way (using each distribution's probability density function) and the ability to apply information criterion ICQMs to approaches to evaluate them. However, there is often a high computational cost for generative clustering algorithms relative to their discriminative counterparts; this becomes a noticeable problem with many of the larger datasets used in current clustering research. Significant effort has gone in to speeding up such generative algorithms. For example, Ordonez and Omiecinski [121] note EM's (the most well cited generative clustering algorithm) slowness and design *FREM* (Fast and Robust EM) to compensate for this problem (among other issues). Online EM [118] is another approach that speeds up EM.

As a side note, some clustering algorithms use generative approaches while clustering despite producing hard clusterings in their final results [70, 125, etc.].

2.2 Data Representations

All clustering algorithms require at least one parameter as input, the representation of X, the dataset to be clustered. The majority of clustering algorithms (and notably, a majority of clustering algorithms used on text clustering) are based around X being represented as numeric valued vectors, that is to say, $x_i = (x_{i1}, x_{i2}, \ldots, x_{im})$, where each $x_{ij} \in \mathbb{R}$ (see Fig. 2.3(a)). m is referred to as the *dimensionality* of X, with each index of the vectors of X being referred to as a *feature* or *dimension*. This formulation makes X equivalent to an n by m matrix, with rows being objects and columns being features. In Section 2.2.4 I will cover representations of text specifically.

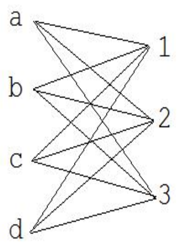
Although categorical features are somewhat common in real datasets and some clustering algorithms operate directly on them [14, 81, 64, etc.], there are many more clustering algorithms that operate exclusively on numeric feature vectors. This is possibly due to the ability to convert a categorical feature value to a numeric vector. Let CA be some categorical feature with possible values $\text{ca}_1, \text{ca}_2, \ldots \text{ca}_t$, and let ca be a particular instance of feature CA. We can represent ca as an ordered numeric vector $(\text{ca}_1^*, \text{ca}_2^*, \ldots \text{ca}_t^*)$, where $\text{ca}_j^* = 1$ if $\text{ca} = \text{ca}_j$ and 0 otherwise (this was done by Ralambondrainy [127] to apply k-means to categorical data). This allows for the conversion of representations containing any combination of numeric and categorical features to an entirely numeric vector.

$$\begin{array}{c|ccc|} & 1 & 2 & 3 \\ \text{A} & 0.5, & 0.0, & 1.0 \\ \text{B} & 0.0, & 0.5, & 1.0 \\ \text{C} & 1.0, & 0.0, & 0.5 \\ \text{D} & 0.5, & 0.5, & 0.5 \end{array}$$

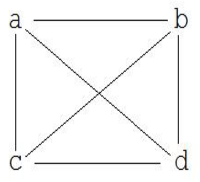
(a)

$$\begin{array}{c|ccc|} & \text{B} & \text{C} & \text{D} \\ \text{A} & 0.8, & 0.8, & 0.77 \\ \text{B} & & 0.4, & 0.77 \\ \text{C} & & & 0.77 \end{array}$$

(b)



(c)



(d)

Figure 2.3: (a) is a dataset represented as a 4x3 matrix. (b) is the upper triangular portion of (a)'s similarity matrix using cosine similarity (see Section 2.3 for a definition of cosine similarity). (c) is (a) represented as a graph. (d) is (b) represented as a graph. Edge weights are omitted in these figures.

Some clustering algorithms use an n by n real-valued matrix called a *similarity matrix*; where the value at i, j is the similarity of x_i and x_j (this similarity value is generated from the objects' base representations, Fig. 2.3(b) gives an example). n by n dissimilarity or distance matrices are also sometimes used by clustering algorithms, where such matrices contain dissimilarities and distances respectively, but I omit further discussion of these for now. Similarity matrices are particularly useful in that any clustering algorithm based on them, such as spectral approaches [40, 119, 139, etc.], may be applied to any X regardless of its object representations, be they vectors or other structures (such as syntactic parsings of natural language), as long as there exists a definition of similarity between objects with such representations. Many clustering algorithms [38, 76, 119, 139, 162, etc.] explicitly exploit the matrix nature of their input (either object by object or object by feature) to produce clusterings (usually by decomposing said matrix into components and/or approximating it, see Section 2.4).

One may phrase object by feature and object by object matrices in graph terms. The object by feature matrix may be thought of as a *bipartite* graph (as shown in Fig. 2.3(c)). A bipartite graph is any graph $G = (V, E)$, where V is the vertices and E is the set of edges between the vertices, such that V may be divided in to two sets V_1 and V_2 such that there is no edge within V_1 and V_2, but there are edges between every element of V_1 and V_2. In this case, V_1 is the set of objects, V_2 is the set of features, and the weights for the edges between the sets are simply the corresponding dimensional values of the object by dimension matrix. A similarity matrix is representable as a general graph $G = (V, E)$, where V is the set of objects in X, and the value for the edge between x_i and x_j is the similarity between x_i and x_j. Typically, one considers there to be no edge between x_i and x_j if they have the minimum similarity possible (usually 0). Beyond the ease of comprehending graph structures, phrasing clustering inputs as graphs has proven to be useful in terms of providing theoretical bases for certain clustering algorithms [119, 139, etc.].

While similarity matrices and numeric feature vectors are the norm in clustering research, they are by no means the only data representations used by clustering algorithms. Often, structural data simply has its structure stripped to produce feature vectors (as is almost universally done with text with the famous *bags of words* model that treats any text as an unordered, unstructured sequence of tokens). Sometimes, this loss of structure is harmful enough that clustering meaningfully becomes extremely difficult, in other situations it is simply not possible to remove the structure in a sensible way. In these cases, it is preferable (or required) to design algorithms that work on the structured data directly. Notable examples of structured data include social networks [154], purchasing records [65], and ranked results (such as search engine result pages from Google).

Features may be clustered like objects by transposing X and treating it like a standard dataset. This fact is exploited by *co-clustering* [72] (also known as *biclustering, direct clustering*, and more than a few other names) algorithms, where both objects and features are clustered in an attempt to yield superior clusterings [12, 30, 40, 140, etc.]. The notion of co-clustering has been extended to *multi-way* clustering [17], where one is clustering objects that have multiple feature sets that are not amenable to being merged into a single vector. For example, a Web page consists of words, images, hyperlinks, etc., and it does not necessarily make the most sense to attempt to represent all these feature sets together in a single numeric vector. The multi-way clustering approach taken by Bekkerman et al. [17] was to combine the various feature sets in a pairwise fashion and derive an overall clustering of the objects from the pairs. Multi-way clustering represents a very difficult and interesting problem with much potential for future research.

2.2.1 Feature Selection

Many clustering algorithms have significant computational costs associated with them. For instance, any clustering algorithm that uses a full similarity matrix of a dataset usually requires $O(n^2m)$ time at least, where n is the number of data items and m is the number of dimensions, without even considering the actual clustering process. Some forms of data have enormous standard representations. For example, with text clustering a dataset might be a document by word matrix. As the number of distinct words for any non-trivial collection of documents tends to be rather large, some way of reducing the size of such a dataset is desirable. Gene datasets are another example of a dataset type associated with large numbers of features. Beyond just reducing the size of a dataset for speed reasons, an effect called the *curse of dimensionality* [19] occurs as m grows larger. The curse of dimensionality states that as the number of dimensions in a feature space grows larger, all points in the space approach being equidistant, resulting in distance measures such as those used in clustering becoming less meaningful. A corollary of this is that clustering itself becomes more difficult as m increases. Although some work indicates distance measures do not necessarily become meaningless as m increases [78], the curse of dimensionality is visible often enough that it provides another strong motive to reduce m when possible. *Feature selection* and *dimensionality reduction* are two techniques that seek to map a larger m dimensioned space to a smaller m^* one, where $m \gg m^*$, such that the 'important' properties of m are preserved. I discuss feature selection here, and dimensionality reduction in the next subsection.

Due to a lack of true labels, feature selection in clustering is a noticeably more challenging task than in supervised learning tasks such as classification. Indeed, Liu and Yu's survey of feature selection techniques [105] shows that one has limited choices for clustering feature selection. Nevertheless, the basic feature selection paradigm for clustering is the same as for other tasks, usually taking a four stage approach:

1. Generate candidate feature sets (subsets of m).

2. Evaluate the goodness of these feature sets.

3. See if a stopping criterion has been reached, if not, go to Step 1.

4. Validate the selected feature set.

Given that there are 2^m possible feature subsets, a brute force search using all possible feature subsets is clearly not an option for any reasonably large m. Although methods exist to do complete searches in less than $O(2^m)$ time [117, etc.], the cost of such searches is still very high, leading to the use of *sequential* and *random* search. Sequential search approaches are greedy hill climbing techniques that iteratively add and/or remove features to each candidate feature set. Typically, the best feature sets are kept for the next iteration while others are pruned, and the algorithm terminates when no meaningful improvement is obtainable on any of the candidate feature sets. Sequential searches are fast and easy to implement, and are also the feature selection search method of choice in clustering. Random searching (such as *simulated annealing* [52]) is usually performed exactly like sequential search, except the initial candidate feature sets may be random, and/or the manner in which potential expansions of current feature sets are investigated is random also. The randomness aspect helps escape local maxima problems, and may also result in a faster algorithm, but it also results in non-deterministic final feature sets in general.

Evaluating the goodness of a candidate feature subset is particularly challenging without true labels. Various concepts of how to evaluate goodness in this context include minimizing redundancy of features by replacing groups of highly correlated inter-correlated features with a smaller number of features [113]; deriving feature links between objects in the full feature set and selecting a feature subset that maintains the pattern of links [36]; attempting to find highly distinct feature sets [94]; and others.

As a final note; the principal way in which feature selection algorithms are categorized is a three way split between *wrapper*, *filter*, and *hybrid* models. A filter model [36, 113, etc.] iteratively evaluates feature sets for goodness using some measure on the feature subsets directly; while a wrapper model [94, etc.] evaluates feature sets by applying a data mining algorithm (in our case, clustering) using the feature sets and evaluating the quality of the resulting structure. Hybrid approaches [79, 132, etc.] do both of these.

Feature selection in text clustering often does not involve any of the complicated approaches discussed above. Instead, it is most common to perform three very simple techniques to reduce feature space size/increase feature quality (assuming the dataset is a document by term matrix, with a document being any logical unit of text). The first is the use of a *stop word* list, which is a list of common terms that are to be stripped from the vector space. Typically, the stop word list contains words such as ‘a’, ‘the’, ‘to’, ‘but’, ‘for’, ‘it’, etc. Such words are seen as unhelpful in determining meaningful similarities/differences in text (although this is not always the case in practice, results I obtained when clustering Web queries [156] suggest that stop words can be useful in clustering).

The second technique is *df (document frequency)* pruning, in which terms that do not occur in at least some threshold of documents are stripped from the vector space. The notion behind this feature selection method is that, in clustering, one is looking for clusters of at least reasonable size in a dataset; a term that occurs in almost no documents is unlikely to provide a basis for such groups. Great care must taken in ensuring that a *df* threshold is set leniently enough that it does not harm clustering results (low *df*, but not too low, is generally viewed as useful, see Section 2.2.3). Often, huge reductions in the size of a feature space can be obtained by using even a very lenient *df* threshold (2-4) because text usually follows a Zipfian term distribution (many terms with low df and few terms with high df).

The final technique involves simply selecting some kinds of features that one does not want. For example, one might decide that any feature that is not solely composed of alphanumeric characters be pruned from the feature space. Thus, for example, I might say tags such as those in XML and HTML are not to be kept. As with *df* pruning, a significant reduction in the size of a feature space can be obtained with this, although the potential for losing information is, likewise, there.

2.2.2 Dimensionality Reduction

Although *dimensionality reduction* is sometimes referred to as a form of feature selection and vice versa, it carries a very different implication. Whereas in feature selection one envisions simply selecting features from the set of those available, dimensionality reduction involves the creation of conjugate features. In the resulting m^* space produced by dimensionality reduction, each feature is the combination of some number of features from m and, moreover, each feature of m is often involved in more than one feature m^*. Note that all the dimensionality reduction techniques discussed here are only applicable to datasets represented either as object by feature matrices or similarity matrices, as the overwhelming majority of research on dimensionality reduction is based on such matrices.

The classic version of dimensionality reduction is *singular value decomposition (SVD)*. In SVD, X is decomposed into three matrices, U, S, and V, such that $X \sim USV^T$ (with ^{T} indicating the transpose of a matrix). U is an n by p matrix, V is m by p matrix, and S is a p by p diagonal matrix containing the singular values of X (typically ordered from least to greatest), where $1 \leq p \leq \min(n, m)$. For the matrix U, every dimension is orthogonal (uncorrelated), and further each dimension is a combination of some dimensions in the original space of X. The dimensions are ordered by decreasing amounts of the variance in X's features they account for, thus the first dimension is the most important, and the last the least. A similar notion may be applied to V, except there, features and objects swap places. The matrices produced by SVD have an interesting property; if we approximate

X as X^*, where $X^* = US^*V^T$ and S^* is S with all columns after the first m^* zeroed out, then $||X - X^*||_F$ (the Frobenius norm) is minimized among all possible choices of X^* that are rank m^*. The Frobenius norm of a matrix Y that is n by m is defined as $||Y||_F = \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{m} Y_{ij}^2}$, thus $||X - X^*||_F$ is how different X and X^* are—a desirable thing to minimize.

SVD is expensive, leading to approximation methods and partial computations. In a landmark paper on applied dimensionality reduction, Deerwester et al. [38] used SVD on a text dataset for an information retrieval task, coining their procedure *Latent Semantic Indexing/Analysis (LSI/LSA)*. In this procedure, a standard document by term matrix is inverted and has SVD applied to it, after which an approximation of the matrix is computed as $X^* = U^*S^*V^{*T}$. S^* has the previously given definition, U^* is the first m^* columns of U, V^{*T} is the first m^* rows of V^T, and X^* is n by m^* (this application of SVD is referred to as *truncated* SVD). To find documents that are similar to queries, one can represent queries as vectors and *fold* them in to the reduced space of X^*, after which standard similarity measures can be used to find documents related to the query. More generally, one may use any set of objects in LSI, as long as they can be represented as vectors, and perform folding of new objects to find the previous objects that are most similar to the new objects (allowing for operations such as classification, etc.). A staggering number of uses for LSI, with clustering numbering among them, have been invented, and LSI has led directly to other research such as *Probabilistic Latent Semantic Indexing/Analysis (PLSI/PLSA)* [76] (which is also often used on text).

Principal Component Analysis (PCA) is another very common dimensionality reduction method that is closely linked with SVD. If one adjusts X so that the mean of all features is zero, then the PCA of X is $(X^TU)^T$, where SVD$(X) = USV^T$, as before (thus one may perform PCA using SVD, although other methods such as using covariance/eigendecomposition or expectation maximization are possible). Each column of $(X^TU)^T$ is orthogonal (uncorrelated) to the other columns of the matrix. As in SVD, to achieve dimensionality reduction one trims the higher dimensions of $(X^TU)^T$ as they are less important. Interestingly, it has been shown that the individual principal components of a dataset correspond to relaxed cluster membership indicators of k-means [47], indicating that the k-means objective function and PCA are similar notions. One notable limitation of PCA is that it assumes that objects of the dataset are linear combinations of the principle components. This may be overcome by performing kernel PCA [136], which involves mapping X to a larger dimensioned kernel space (as is done in non-linear support vector machines) and then performing PCA in the kernel space.

Independent Component Analysis (ICA) [31] is much like PCA, except it produces components that are independent, with independence being a stronger property than PCA's uncorrelated components. ICA is principally applied in signal processing, but it can be used in clustering [23, etc.].

Eigendecomposition can be used as dimensionality reduction when clustering. However, because it requires square matrices, it is applied to modified similarity matrices and not object by feature matrices. Most notably, it forms the basis of spectral clustering algorithms [119, 139, etc.]. In eigendecomposition, a square matrix X is decomposed into QEQ^{-1}. Q is an n by n matrix where each column is referred to as an eigenvector (usually with length normalized to 1), while E is an n by n diagonal matrix of eigenvalues. As with SVD and PCA, lower dimensions of Q are more important. The approach taken by spectral clustering methods is usually to generate a specialized similarity matrix called a Laplacian (detailed in Section 2.4), perform eigendecomposition on it, and use only some number of columns of Q (eigenvectors) to generate a clustering.

A trend in clustering is to apply dimensionality reduction dynamically as a dataset is clustered [49, etc.]. Results of clustering algorithms based on this concept have been quite encouraging as they appear to perform well relative to the standard procedure of applying static dimensionality reduction to a dataset then clustering the reduced representation. There are many avenues to explore with respect to dynamic dimensionality reduction, both for clustering and other machine learning techniques.

While many other forms of dimensionality reduction exist, those described above are the most common with respect to clustering. I summarize this subsection by noting two things; firstly, there is more than sufficient research to conclude that clustering benefits significantly from dimensionality reduction over a wide range of applications and datasets; and secondly, dimensionality reduction of any kind, including when it is applied dynamically, is very costly relative to the application of simple clustering algorithms. One may think of dimensionality reduction as a tradeoff between quality and speed.

2.2.3 Feature Weighting

Often times, we have a notion that some features in a feature space are more important than others, but we do not desire to simply remove the less important features (as in feature selection) or transform the space (using dimensionality reduction). In such a case, feature weighting is the operation of choice. Here the influence of features we deem are important have increased impact on clustering results by being given increased weights in the vectors of the dataset. For example, consider a dataset of objects with 4 features, each ranging from 0 to 1. Let us say we know the first feature is the most important. Then we might multiply the value of the first feature for each object by some number greater than 1. As a result, when we are clustering using the measures detailed in Section 2.3, the first feature will likely have a greater overall influence on the clustering result than any other individual feature. Feature weighting is focused on improved clustering results, and usually does not offer speed increases during clustering that come with reduced feature spaces created by feature selection or dimensionality reduction. Because weighting methods tend to be extremely domain specific, I only discuss text feature weighting in this thesis.

2.2.4 Text Dataset Representations

Although there are many distinct kinds of text datasets (Web pages, emails, scientific abstracts, news articles, etc.), the vast majority are given the same common representation, referred to as a 'bag of words' model, while clustering. In such a model the raw text (after some preprocessing) of each object is treated as a sequence of tokens (with tokens often, but not necessarily, being delimited by white spaces). The tokens are often subjected to a *stemmer* which uses linguistic knowledge to conflate some different tokens to the same form. For example, 'run', 'runs', and 'running' might all be stemmed to 'run'. While there are many varieties of stemmers, the most ubiquitous for English is the Porter stemmer [126]. I refer to all the different kinds of tokens found in the dataset after the above process *terms*. The order in which tokens occur within each object is discarded, resulting in a conceptual 'bag of words' for the object containing terms and how often they occurred. Note that 'bag of terms' would be more accurate to use here because tokenization defines the units of text in the bag, allowing for non-word tokens, but bag of words is the standard in the literature. It is straightforward to take all the bags of words for a dataset as a whole and transform them into a document by term matrix X, where x_{ij} is the number of times term

j occurs in document i (the *term frequency (tf)* of j in i, denoted tf_{ij} for short). While it is clear that this bag of words model strips all the structural information out of the text, it is nonetheless the basis of most document clustering and produces reasonable results.

Typical practice in text clustering is to incorporate an *inverse document frequency (idf)* feature weighting into the raw document by term matrix:

$$x_{ij} = tf_{ij} \log(\frac{n}{n_j}), \tag{2.1}$$

where n is the number of documents in X, and n_j is the number of documents in X that contain term j. To avoid unfavorable biases based on different document lengths, it is also a standard to make each document unit Euclidean length ($||X_i||_2 = 1$) by setting each x_{ij} as follows:

$$x_{ij}^{'} = \frac{x_{ij}}{\sqrt{\sum_{j=1}^{m} x_{ij}^2}} \tag{2.2}$$

The majority of document clustering literature discusses using *tf-idf* weighting, mostly with length normalization [8, 140, 145, 162, 80, 172, 173, etc.]. Less commonly, raw document by term matrices, with and without length normalization, are used directly in clustering algorithms [16, 38, 59, 76, etc.]. It is interesting to note that Eq. 2.1 has been superseded in other fields of research which use text (such as Information Retrieval) by better weighting functions, yet there is little indication of adoption/investigation of new weights for text clustering. In Chapter 5, I will investigate this issue, showing that there are better text weighting methods than simple tf-idf.

Before any weighting is done on the document by term matrix *df* pruning, stop word lists, and general feature pruning are often applied. After weighting, dimensionality reduction is sometimes applied before clustering.

2.3 Similarity/Dissimilarity Measures

Every clustering algorithm is based, directly or indirectly, on some particular notion of what makes individual pairs of objects similar/dissimilar. Although the exact method in which pairwise similarities/dissimilarities are used to derive a clustering varies greatly by the clustering algorithm, the similarity/dissimilarity measures themselves are usually drawn from a small pool of well known measures, the most common of which I describe here.

I denote a similarity measure between two objects as s, with $s(x_i, x_j)$ being the similarity between x_i and x_j; higher $s(x_i, x_j)$ indicates x_i and x_j are more similar. I denote a dissimilarity measure as d, with $d(x_i, x_j)$ being the dissimilarity of x_i and x_j; lower $d(x_i, x_j)$ indicates higher similarity. A clustering algorithm may use either a similarity measure or a dissimilarity measure while clustering (or it may use neither directly, but in such case it will still be based indirectly on one or the other). Often, but not always, a particular d used in a clustering algorithm is a distance function over $\mathbb{R}^m$ for any positive integer m. That is to say, d satisfies all of the following:

1. Non-negativity: For all $x_i, x_j \in \mathbb{R}^m$, $d(x_i, x_j) \geq 0$.
2. Identity: $d(x_i, x_j) = 0$ if and only if $x_i = x_j$.
3. Symmetry: For all $x_i, x_j \in \mathbb{R}^m$, $d(x_i, x_j) = d(x_j, x_i)$.
4. Triangle Inequality: For all $x_i, x_j, x_l \in \mathbb{R}^m$, $d(x_i, x_j) + d(x_j, x_l) \geq d(x_i, x_l)$

I first consider measures that apply to objects represented as numeric vectors in $\mathbb{R}^m$ as that is the standard representation for text in clustering. The most well-known of these, and possibly the most widely used distance function in clustering today, is *Euclidean distance*:

$$d_{euc}(x_i, x_j) = \sqrt{\sum_{l=1}^{m} (x_{il} - x_{jl})^2}. \tag{2.3}$$

This function measures the straight line distance between x_i and x_j. Euclidean distance forms the basis of many clustering algorithms, notably most k-means type algorithms [91, 106, etc.], and is also often used in notions of clustering quality (see Chapter 4). Euclidean distance is often referred to as L^2, as it is a specific instance of the L^p norm distances. An L^p norm distance is defined as:

$$d^{L^p}(x_i, x_j) = \sqrt[p]{\sum_{l=1}^{m} (x_{il} - x_{jl})^p}, \tag{2.4}$$

where p is any real number between 1 and ∞ inclusive. From this definition we can see L^1 is *Manhattan distance* (also known as *city-block* distance):

$$d_{man}(x_i, x_j) = \sum_{l=1}^{m} |x_{il} - x_{jl}|, \tag{2.5}$$

and L^∞ is *Chebyshev distance*:

$$d_{chby}(x_i, x_j) = \max_{l=1..m} |x_{il} - x_{jl}|. \tag{2.6}$$

These three are the L^p-norm distances one is likely to see in clustering, with Euclidean being the most common by far.

Kullback-Leiber Divergence (KL-Divergence) has been used in clustering as a dissimilarity measure [12, etc.]. For its application to make sense, each object must be a vector with entries that sum to one, as KL-Divergence expects probability distributions as input. Formally, KL-Divergence is:

$$d_{kl}(x_i, x_j) = \sum_{l=1}^{m} x_{il} \log(\frac{x_{il}}{x_{jl}}). \tag{2.7}$$

Many datasets are not amenable to representations as probability distributions so KL-Divergence has more limited use in clustering than other measures discussed here. Further, it is not a distance function as it is not symmetric, although it can trivially be made so:

$$d_{skl}(x_i, x_j) = d_{kl}(x_i, x_j) + d_{kl}(x_j, x_i). \tag{2.8}$$

The Jensen-Shannon divergence can act as a smoothed, symmetric KL-Divergence:

$$d_{js}(x_i, x_j) = 0.5 d_{kl}(x_i, avg(x_i, x_j)) + 0.5 d_{kl}(x_j, avg(x_i, x_j)). \tag{2.9}$$

In this context, $avg(x_i, x_j)$ is the average of the distributions x_i and x_j ($avg(x_i, x_j)_l = \frac{x_{il}+x_{jl}}{2}$).

With respect to text, one can represent text units as probability distributions over terms, allowing KL-Divergence to be applied.

Spectral clustering algorithms [119, 139, etc.] and some others often use *Gaussian affinity* as a similarity measure when clustering, where Gaussian affinity [119] is defined as:

$$s_{gauss}(x_i, x_j) = e^{\frac{-d_{euc}(x_i, x_j)^2}{2\sigma^2}}. \tag{2.10}$$

σ is a parameter, the selection of which has been seen to be crucial to Gaussian affinity's effectiveness when used in spectral clustering [119]. For a good discussion on the selection of this parameter, readers may consult Mouysset et al. [116]. Gaussian affinity and spectral clustering have been paired together and used on text in several papers [162, etc.].

The *Mahalanobis distance* [107] between two vectors that come from the same distribution Y is:

$$d_{maha}(x_i, x_j) = \sqrt{(x_i - x_j)^T \text{CoVar}(Y)^{-1}(x_i - x_j)}, \tag{2.11}$$

where $\text{CoVar}(Y)$ is the covariance of Y. Mahalanobis distance is sometimes used as a object-wise distance measure in clustering, although it is often also used as a measure of distance between a vector x_i and a distribution Y (as in Gaussian EM [39]). In such a case the formula becomes:

$$d_{maha}(x_i, Y) = \sqrt{(x_i - \mu_Y)^T \text{CoVar}(Y)^{-1}(x_i - \mu_Y)}, \tag{2.12}$$

where μ_Y is the mean vector for the distribution Y.

Cosine similarity, arguably the most common similarity measure, is defined as:

$$s_{cos}(x_i, x_j) = \frac{\sum_{l=1}^{m} x_{il} x_{jl}}{\sqrt{\sum_{l=1}^{m} x_{il}^2}\sqrt{\sum_{l=1}^{m} x_{jl}^2}}. \tag{2.13}$$

Cosine similarity ranges between -1 and 1 and, as the name suggests, measures the cosine of the angle between x_i and x_j. Its popularity as a similarity measure in text clustering is likely traceable back to Salton and Buckley's early use of cosine in information retrieval experiments [134].

Extended Jaccard similarity is a less popular alternative to cosine:

$$s_{ej}(x_i, x_j) = \frac{\sum_{l=1}^{m} x_{il}x_{jl}}{\sum_{l=1}^{m} x_{il}^2 + \sum_{l=1}^{m} x_{jl}^2 - \sum_{l=1}^{m} x_{il}x_{jl}}. \tag{2.14}$$

Categorical object vectors use different measures of similarity/dissimilarity than numeric ones. For binary categorical vectors x_i and x_j, let $n_{ab}(x_i, x_j)$ be the number of pairs of features (l_1, l_2) such that $x_{il_1} = x_{jl_1} = a$ and $x_{il_2} = x_{jl_2} = b$, let $n_a(x_i, x_j)$ be the number of features for which $x_{il} = x_{jl} = a$. Three of the most common similarity measures for binary categorical vectors are *Dice's coefficient* [46]:

$$s_{dice}(x_i, x_j) = \frac{n_1(x_i, x_j)}{2||x_i||}, \tag{2.15}$$

the *Jaccard coefficient* [85]:

$$s_{jac}(x_i, x_j) = \frac{n_{11}(x_i, x_j)}{n_{00}(x_i, x_j) + n_{01}(x_i, x_j) + n_{10}(x_i, x_j)}, \tag{2.16}$$

and the *Rand* Index [128]:

$$s_{rand}(x_i, x_j) = \frac{n_{11}(x_i, x_j) + n_{00}(x_i, x_j)}{n_{00}(x_i, x_j) + n_{01}(x_i, x_j) + n_{10}(x_i, x_j) + n_{11}(x_i, x_j)}. \tag{2.17}$$

Hamming distance [71] is often used on binary and non-binary categorical vectors. It measures the number of features for which x_i and x_j have different values:

$$d_{ham}(x_i, x_j) = |\{(x_{il}, x_{jl}) : x_{il} \neq x_{jl}\}|. \tag{2.18}$$

Hamming distance is often used for comparing strings, and is one of a class of similar definitions for *edit distance*, the amount of editing required to make one string equivalent to another.

For a dataset that is transposed so that it is feature by object (one where we are trying to cluster features), the use of a correlation similarity measure makes sense. *Pearson's correlation coefficient* [130] r is often used as a similarity measure in such cases:

$$s_{r^2}(x_i, x_j) = \frac{ss(x_i, x_j)^2}{ss(x_i, x_i)ss(x_j, x_j)}, \tag{2.19}$$

where $ss(x_i, x_j) = \sum_{l=1}^{m}(x_{il} - \bar{x_i})(x_{jl} - \bar{x_j})$ and $\bar{x_i}$ is the average value of index i over the dataset. While other correlation coefficients such as *Kendall's Tau* [93] and *Spearman's Rank Correlation* [144] can be applied to feature by object matrices, Pearson's correlation coefficient is the standard choice.

There are many other notions of similarity/dissimilarity between objects beyond those described above that one may run into, including extensions of those described here. For clustering though, a reader will find that the large majority of papers use the measures described here.

2.4 Algorithms

In this section I describe some of the clustering algorithms appearing in the literature. The algorithms are organized by the broad concept type they use while clustering. Within each concept type I cover some of the more famous representatives and many of those that have been applied to text clustering specifically. Throughout this section I use the term *objective function* to refer to what a clustering algorithm is trying to optimize. Note that this distinct from the actual optimization technique used by a clustering algorithm.

2.4.1 Center-Based Clustering

Arguably the most well recognized concept in clustering, the *k-means objective function* has provided a basis for a staggering number of clustering algorithms. It is formally defined as follows: *for some dataset X, we seek a set of vectors $V = (v_1, v_2, ...v_k)$, over the same feature space as X, such that:*

$$\sum_{x_i \in X} \min_{v_j \in V} d_{euc}(x_i, v_j)^2 \tag{2.20}$$

is minimized. The notion behind this definition of quality is compression based: we want to represent each vector of X as one of k vectors (*vector quantization*). Eq. 2.20 conceptually defines a 'center-based' clustering C of X with k clusters. At the center of each cluster c_j is the vector v_j, referred to as the *centroid* of c_j. The members of c_j are all $x_i \in X$ such that $d_{euc}(x_i, v_j)^2 = \min_{v_l \in V} d_{euc}(x_i, v_l)^2$ (all objects which are closer to c_j's centroid than any other centroid).

Minimizing the k-means objective function for a clustering over a dataset is NP-hard, which has led to many heuristics and approximations. The simplest, and most often used, method for obtaining a local minimum in Eq. 2.20 is Lloyd's method [106]. In this method the initial centroids are selected randomly. Then a repetition of two steps occurs; 1) all objects of X are assigned to their nearest centroid forming clusters; 2) centroids are recomputed as the mean of the objects assigned to their cluster. The loop stops when the centroids do not change from one iteration to the next, or when the improvement in Eq. 2.20 from one iteration to the next is below some threshold. Despite its age, k-means using Lloyd's method is still the most popular clustering algorithm by far, seeing widespread use [86], perhaps because it is so simple and often very fast. The k-means problem has been extended to deal with categorical features [81, 28, 127, etc.] and soft clustering [20, 53, 28, etc.]. Initial centroid selection before applying Lloyd's method has been investigated with significant success [11], with other work on designing k-means type algorithms based around making the algorithm insensitive to initial centroid selection entirely [167, etc.].

Most of the methods discussed above are heuristics for k-means type problems. There are a number of works that focus on the approximation side of things, designing clustering algorithms that are assured to yield clusterings within some fixed amount of the optimal value of Eq. 2.20 [84, 99, 108, etc.]. Despite offering assured performance, k-means approximation approaches have seen little use outside of their actual design, they are often computationally expensive and/or the bounds they offer are impractically weak.

There are numerous adjustments/improvements on the notion of using distance from centroids as quality measures. One simple change to Eq. 2.20 is to enforce the use of *medoids* in place of centroids, where the medoid for a cluster c_j is simply the $x_i \in X$ such that $d_{euc}(x_i, c_j)^2 = \min_{x_l \in X} d_{euc}(x_l, c_j)^2$ (assuming the cluster is non-empty). Eq. 2.20 using medoids is referred to as *k-medians*. *Partitioning Around Medoids* (*PAM*) [91] uses medoids, initially starting with random medoids and, for each iteration, assigning points to the closest medoid and swapping a pair of (medoid, non-medoid) objects such that the k-medians objective function is improved the most possible. It terminates when no improvement is possible. Another adjustment on the use of centroids is presented by CURE [64].

Zhao and Karypis [171, 172, 173] offer a number of objective functions similar to the traditional k-means objective function. Their E_1 objective function defines quality as:

$$\sum_{v_j \in V} |c_j| cos(v_j, V_X), \tag{2.21}$$

where V_X is the centroid of the entire dataset and $|c_j|$ is the size of cluster c_j. Other functions presented in their works that are closely center-based are I_1, I_2, H_1, and H_2. Optimizing I_1, with some algebra, can be shown to be identical to minimizing Eq. 2.20. I_2 is simply Eq. 2.20 with cosine similarity instead of Euclidean distance. H_1 is I_1/E_1, and H_2 is I_2/E_1.

With respect to text clustering, k-means with Lloyd's method has been seen to be better than *Unweighted Pair Group with Arithmetic Mean* (UPGMA) [145] (see the following section for a description of UPGMA), indicating that it provides reasonable text clustering solutions. However, repeated bisecting k-means using Lloyd's method is even better than this [145]. From this we can conclude k-means using Lloyd's method is not an ideal text clustering algorithm. It is, however, extremely fast relative to most competitors. Of the objective functions discussed from Zhao and Karypis' works, I_2, especially when applied using a repeated bisecting approaching, is an excellent text clustering technique. PCA, followed by k-means, has been shown to offer large improvement in text clustering over just k-means [47], although one can see from the results that too aggressive of a dimensionality reduction during PCA is not optimal.

2.4.2 Linkage-Based Clustering

Linkage-based clustering algorithms, as their name suggests, derive clusterings by using the links (similarities) between clusters. The three most well-known linkage based clustering algorithms, UPGMA [88], *single linkage* [88], and *complete linkage* [141], are from the same family of agglomerative clustering algorithms. In each of these clustering algorithms every object of the dataset begins as a singleton cluster and clusters are progressively merged with the best similarity. In single linkage similarity between clusters is equal to the similarity of their closest objects. In complete linkage the farthest objects are used. In UPGMA the average similarity between all objects is used. Any notion of similarity may be used by these methods.

While the above three are the standard linkage based clustering algorithms, other linkage related techniques exist. ROCK [65] is a well-known link based algorithm designed for categorical data. A less common variant of UPGMA is one where the similarity between the centroids of each cluster is used for merging [145], and the I_1 function from [171] is highly similar to UPGMA. Linkage-based clustering tends to be on the slower side of things. If we ignore initial similarity computation time, one of the fastest linkage-based clustering method is single linkage at $O(n^2)$ time.

There are good indications that UPGMA is the best of those discussed above for text clustering [145, 173], with single and complete linkage producing extremely poor text clusterings. This should not be taken to mean those algorithms are useless by any means. Single linkage, for instance, has the desirable property of finding a *minimum spanning tree* [97] with k components for any k and is amenable to theoretical analysis [4, 166, etc.] with very well understood properties.

2.4.3 Subspace Clustering

Subspace clustering is the location of clusters in a subset of the full feature space of a dataset [123]. Because many datasets have large dimensionality (especially text), and clusters are often represented in only a small number of dimensions, subspace clustering is often useful. Dimensionality reduction clustering methods [49, 47, 50, 162, etc.] are closely related to subspace clustering, as they attempt to map datasets to spaces where clusters exist. However, as discussed in Section 2.2.2, dimensionality reduction methods create aggregate dimensions; subspace clustering algorithms do not. Further, subspace clustering algorithms are different in that they purposefully select dimensions to disregard or use while clustering. This difference makes subspace clustering much more like wrapper feature selection. I discuss some of the more well known subspace clustering algorithms here. For further details consult Parsons [123] and Kriegel et al. [96].

Some subspace clustering algorithms produce only hard partitions. PROCLUS [5] is an example of this. The main body of PROCLUS operates almost exactly like k-medoids using Lloyd's method, except that the selection of cluster medoids, and the distance computation between medoids and other objects is different. Every cluster is assigned a set of relevant dimensions in each iteration (using standard deviation of the dimensions from points nearby the medoid to decide if the dimension is relevant or not). Medoids, and the distance from any medoid, are computed using only relevant dimensions. One enhancement of PROCLUS is FINDIT [158], another is ORCLUS [6], which allows for the discovery of clusters that are not necessarily parallel to the dimensions of the dataset. In ASI (Adaptive Subspace Iteration [103]), during each iteration of the algorithm every cluster is mapped to a relevant subspace, and its quality in that subspace is used for its impact on overall clustering quality and for cluster membership assignment (making the algorithm have a distinctly subspace feel to it). ASI's results on text clustering are encouraging, although they are not compared against the best algorithms from Zhao and Karypis [173], only to linkage-based clustering methods. Ding et al. [50] follow a vein similar to ASI, using PCA and latent dirichlet allocation. Again, an improvement over another text clustering algorithm is shown (PCA + k-means).

An alternative to producing hard partitions, and one which is arguably much more ambitious for a subspace clustering algorithm, is the task of locating all meaningful clusters in any subspace of the dataset, irrespective of their overlap (yielding a hard non-disjoint clustering). Grid partitioning of the feature space is a common approach to this task. CLIQUE [7] is the first well-known subspace algorithm to perform this operation. In CLIQUE, the feature space is cut up into fixed width axis parallel grid blocks along every feature. From this basis, CLIQUE behaves much like the APRIORI algorithm, each iteration dense subspaces over a dimensions, containing some number of dense grid blocks, are checked to see if they combine to make dense $a + 1$ dimension spaces. This procedure is repeated until an iteration finds no new dense subspaces. Additionally, subspaces are pruned using a minimum description length principle. Extensions and improvements on CLIQUE include ENCLUS [29], which uses new definitions of quality and allows for the discovery of more than one cluster in a dense subspace; MAFIA [62], which uses an adaptive grid partitioning of the feature space as opposed to a static one; and nCluster [104], which allows for overlapping grid units. With respect to using the APRIORI algorithm directly, frequent item set clustering algorithms [16, 59, etc.] have been designed for text clustering using it directly as their basis.

Soft-like subspace clustering algorithms are rare, but some exist, such as COSA [58]. COSA does not really cluster, but rather determines a similarity matrix for the dataset, suitable for use by a number of other clustering algorithms.

2.4.4 Density-Based Clustering

For our discussion here, a density-based clustering algorithm is one that looks for dense clusters in the entire feature space of the dataset. This is different from subspace clustering, where one may locate clusters that exist only in subsets of the full feature space. By using only the full feature space, density-based methods avoid having to investigate a staggering number of different subspaces, resulting in much faster clustering algorithms. However, this comes at a cost. Clusters may not be readily detectable in the full feature space; in such cases a strictly density-based approach is likely to fail.

DBSCAN [54] is the most well known density-based clustering algorithm. DBSCAN performs a depth first clustering of the dataset in the following manner: each $x_i \in X$ is examined in a random order. Let ϵ-neighborhood$_i$ be the objects within ϵ distance of x_i. If x_i is part of a cluster already, it is ignored, otherwise, if $|\epsilon\text{-neighborhood}_i| \geq minPts$ (some constant), $x_i \cup \epsilon$-neighborhood$_i$ becomes the basis for a new cluster. For each x_j in that cluster, recursively, its ϵ-neighborhood$_j$ is added to the cluster if $|\epsilon\text{-neighborhood}_j| \geq minPts$. After all objects have been visited, DBSCAN classifies all the unclustered objects remaining as outliers. ϵ and $minPts$ are parameters to which DBSCAN is known to be extremely sensitive, and DBSCAN is further sensitive to the order in which objects are examined. Some improvements upon DBSCAN are made by GDBSCAN [135] and OPTICS [10]. SUBCLU [92], a subspace clustering algorithm, uses DBSCAN on subspaces of the full feature space to see if they contain dense clusters.

DENCLUE [74] represents an alternative approach to finding dense clusters in the full feature space. For DENCLUE, the probability density function at any arbitrary point in the full feature space is defined using Gaussian kernels. Conceptually, clustering is done by 'pulling' each object of the dataset, using a greedy hill climbing approach, to the nearest local maxima in that probability density function. Once all objects have been pulled to their respective local maxima, all objects residing in close maxima are assigned to be members of the same cluster. DENCLUE 2 [73] offers a significant speed improvement over DENCLUE, converging to each object's local maxima much faster.

2.4.5 Matrix Factorization/Approximation Clustering

Matrix factorization/approximation techniques have proven to be amazingly powerful tools for clustering in a wide variety of applications areas. Their strength invariably lies in their ability to draw the latent (hidden) similarities in a dataset out.

One of the earliest uses of matrix factorization in a task related to clustering was LSA [38], which I have already described. LSA may be used for dimensionality reduction of a dataset before clustering and, likewise, PCA may be applied before clustering. Both of these techniques have been seen to increase the quality of text clustering results [47, 143], but require a parameter in the form of the number of dimensions to use in the reduced space when clustering. Careful selection of this parameter is necessary, as results deteriorate with too few or too many dimensions, with the ideal number of dimensions varying by dataset.

Spectral clustering algorithms are extremely popular. At the heart of every spectral clustering algorithm is its graph Laplacian L. After L is computed, spectral clustering algorithms follow one of a few simple procedures to generate a k-way clustering. One such procedure is: the eigendecomposition $L = QEQ^{-1}$ is computed, then the first k

eigenvectors of Q (where k is number of desired clusters) are used by some other clustering algorithm (usually k-means using Lloyd's method) to produce a k-clustering of X. Ideally, L is designed in such a way as to result in Q containing very clear cluster membership indicators, allowing other clustering algorithms to easily find the clusters in L that would have otherwise been hidden in X.

In order to create L, a spectral clustering algorithm first uses X to create an n by n (object by object) symmetric matrix referred to as a weighted adjacency matrix W. This might be done simply by having $W_{ij} = s(x_i, x_j)$ for all i, j using some similarity function that is symmetric (Gaussian affinity being a common choice). One can also define:

$$W_{ij} = \begin{cases} s(x_i, x_j), & \text{if } x_j \in nn_{i,r} \text{ or } x_i \in nn_{j,r}; \\ 0, & \text{otherwise.} \end{cases} \tag{2.22}$$

or:

$$W_{ij} = \begin{cases} 1, & \text{if } s(x_i, x_j) \geq \epsilon; \\ 0, & \text{otherwise.} \end{cases} \tag{2.23}$$

where $nn_{i,r}$ is the r-nearest neighborhood of x_i (using the appropriate similarity function). r and ϵ are parameters. Many other methods for generating W are possible. W is used to generate L in a manner that, again, depends on the algorithm. For example, in unnormalized spectral clustering [40], $L = D - W$, where D is the degree matrix of W.

Two of the most popular spectral clustering algorithms are presented by Shi and Malik [139] and Ng et al. [119]. Shi and Malik's algorithm, commonly referred to as *N-Cut*, *normalized cut*, or *NC*, uses $L = I - D^{-1}W$, where I is the identity matrix. Let C be a clustering, and let l_i be the ith object in the laplacian L (which is also the ith object in

X, but with a different vector representation). Then Shi and Malik's approach is aimed at optimizing the normalized cut objective function:

$$\text{N-Cut}(C) = \sum_{i=1}^{k} \frac{cut(c_i, L - c_i)}{assoc(c_i, L)}, \tag{2.24}$$

where $cut(c_i, L - c_i) = \sum_{l_j \in c_i} \sum_{l_o \in (L - c_i)} L_{jo}$ and $assoc(c_i, L) = \sum_{l_j \in c_i} \sum_{l_o \in L} L_{ok}$. The cost of each cluster for N-Cut is its internal edges divided by all its edges (internal and external). Normalized cut favours balanced cluster sizes. Ng. et al. use the same cut notion, but set $L = D^{-1/2}(D - W)D^{-1/2}$. Other cuts for spectral clustering include *Ratio-Cut* [66], which is defined as:

$$\text{Ratio-Cut}(C) = \sum_{i=1}^{k} \frac{cut(c_i, L - c_i)}{|c_i|}, \tag{2.25}$$

and MinMax-Cut [48]:

$$\text{MinMax-Cut}(C) = \sum_{i=1}^{k} \frac{cut(c_i, L - c_i)}{\sum_{l_j, l_o \in c_i} L_{jo}}, \tag{2.26}$$

which also serves as the basis for the G' objective function in Zhao and Karypis [173]. Ratio-Cut weights each cluster by its size, making no use of similarities within each cluster. As a result, optimizing Ratio-Cut may result in objects being grouped together that share very low similarity (they need only exhibit the same dissimilarities to other objects). MinMaxCut defines each clusters' quality as its external edges divided by its internal edges, closely linking it with B/W ICQM I discuss in Chapter 3 (average between cluster scatter divided by average within cluster scatter).

Because of their ability to make non-Gaussian type clusters easy to find, spectral clustering algorithms fair well in tasks such as image segmentation [139, 159, etc.], text clustering (where N-Cut is often used as a baseline to compare against as in Wei et al. [162, etc.]), and other domains. However, a notable drawback of spectral clustering is speed, the use of eigendecomposition makes spectral clustering very slow. A full eigendecompostion need not be computed to perform spectral clustering though, techniques such as the power method allow one to approximately compute the required eigenvectors more quickly. Sample-based spectral clustering [56] is relatively fast also. Alternatively, Dhillon et al. [41] note that, at least for N-Cut, one does not need to use expensive eigendecomposition at all, other faster methods of obtaining an optimization of N-Cut exist.

Non-negative Matrix Factorization (NMF), sometimes referred as *Non-negative Matrix Approximation* (as it is typically used to obtain an approximation of a matrix), is one of the newer matrix based clustering methods. The basic concept of NMF is to approximate $X \sim WH$, where W is n by k and H is k by m. X, W, and H are required to be non-negative, and k is the number of clusters parameter. After W and H are computed, W is taken, after some normalization, as a cluster membership indicator matrix (the value W_{ij} being the magnitude of object i's membership in cluster j).

In NMF W and H are usually seeded with random values to start. Updating functions are applied to W and H iteratively such that the quality of approximation of X by WH is strictly greater than or equal to that of previous iterations. Any of the standard stopping criteria used by k-means with Lloyd's method may be applied to terminate an NMF algorithm (number of iterations, convergence, lack of meaningful improvement). The exact updating functions used are entirely dependent on how one defines 'quality of approximation' and can be rather complicated. For example, in Wei et al. [162], the Frobenius norm

$||X - WH||_2$ is the approximation measure, allowing for the multiplicative update rules of:

$$H_{ij} \leftarrow \frac{(W^T X)_{ij}}{(W^T W H)_{ij}}, \tag{2.27}$$

and:

$$W_{ij} \leftarrow \frac{(X H^T)_{ij}}{(W H H^T)_{ij}}, \tag{2.28}$$

to be used. In general, many updates rules are possible for a given quality of approximation notion, although some result in much slower NMF algorithms than others. A discussion of a large number of other potential approximation measures, and their corresponding update functions, can be found in Dhillon [44]. NMF has been seen to be well suited to text clustering when using the Frobenius norm [162], and has had various extensions introduced to handle all manner of extra clustering issues such as prior information [165]. It is worth noting that NMF does not ensure orthogonality in W and H, making the results it yields potentially very different from PCA, LSA, or spectral clustering. Bao et al. [13] investigate using NMF with approximation/update functions that do enforce orthogonality, obtaining encouraging results on text.

2.4.6 Model-Based Clustering

In model-based clustering we usually seek a description of how the dataset was created and a measure of how good the description is (see Section 2.1.3). The *Expectation Maximization* (*EM* [39]) procedure forms the basis of most model-based clustering methods. It is a simple two step procedure. We have some model of our dataset (typically a set of distributions) with some parameters. An EM procedure alternates between computing the most likely parameters for the model given the dataset and computing the likelihood of the dataset given model. This simple definition introduces a 'chicken and the egg problem' that is solved by bootstrapping initial parameter values in to EM. In the case of clustering, it is not uncommon to apply k-means using Lloyd's method to obtain starting parameters in an EM clustering algorithm. EM type algorithms are almost always slower than other simpler algorithms such as k-means using Lloyd's method.

Probabilistic Latent Semantic Analysis (*PLSA* [76, 77]) uses EM. In PLSA, objects are represented as probability distributions over features. The dataset is taken to have been generated by a hidden (latent) set of topics $Z = (z_1, z_2, \ldots z_k)$, where each $z_i \in Z$ is a probability distribution over features. PLSA alternates between computing the distributions $p(z_i)$ (the weight of each topic), $p(f_j|z_i)$ (the membership of each feature in each topic), and $p(x_j|z_i)$ (the membership of each object in each topic). If one considers each z_i to be a cluster, then the $p(x_j|z_i)$ distribution may be viewed as soft cluster membership indicators for X. Alternatively, PLSA may be applied as a form of dimensionality reduction, and a secondary clustering may be applied to $p(x_j|z_i)$ to create a final clustering. PLSA has had various extensions, such as being made hierarchical [60], and is a very popular algorithm for text applications [60, 76, 77, 163, etc.] and other many other areas. PLSA has been shown to be closely linked with both NMF [51] and *Latent Dirichlet Allocation* (*LDA* [22]) [61], which is described below.

On closer investigation of PLSA, one may note that it does not define a true model of the dataset. Given some previously unseen object x_*, PLSA does not allow the determination of its cluster membership based on the distributions it has computed previously. Instead, one must add x_* to X, fix $p(f_j|z_i)$, and recompute $p(x_j|z_i)$ and $p(z_i)$ using the PLSA procedure all over again to derive the values $p(x_*|z_i)$ values for x_*. LDA [22] tends to produce better results than PLSA while dealing with this problem. In LDA, an additional layer is added on top of the PLSA in the form of a dirichlet prior distribution that specifies probabilities of sampling the various topics in Z. Unsurprisingly, the additional layer used by LDA comes with a speed cost, and the algorithm itself is significantly more complicated than PLSA. An example of another, more recent EM-type clustering algorithm, along with pointers to other such algorithms, can be found in Kurihara and Welling [100].

One interesting vein in model-based clustering that is not strictly related to EM is the use of fitting measures to simultaneously cluster the dataset and select the number of clusters [70, 55, 125, etc.]. X-means [125] uses information criteria such AIC [75] to select clusterings. Unfortunately, X-means often leads to over segmentation of datasets [70] (too many clusters). G-means [70] uses k-means with Lloyd's method and a fast measure of Gaussian fitness to determine when clusters need to be split more, and is effective when true clusters have minimal overlap. PG-means [55] is a significant improvement over this, allowing the use of any EM algorithm for Gaussian distributions while clustering, and therefore being able to handle clusters with significant overlap.

2.4.7 Information Theoretic Clustering

Information theoretic clustering algorithms work explicitly with the trade-off between *compression* and *distortion* when clustering. Compression is how concise a clustering is, while distortion is how much (important) information is lost in the clustering. Compression is a value to maximize and distortion is a value to minimize, with improving one generally resulting in degrading the other. An information theoretic clustering algorithm attempts to balance these two properties in a clustering solution.

The Information Bottleneck Method (*IBN* [148]) is the principle information theoretic clustering method. Let C be a soft clustering of X, and let Y be a variable representing the 'important' information in X, then IBN seeks to minimize:

$$\mathrm{MI}(X, C) - \beta \mathrm{MI}(C, Y), \tag{2.29}$$

where $\mathrm{MI}(A, B)$ is the mutual information between two random variables A and B (see NMI in Section 3.2 for its calculation). $\mathrm{MI}(X, C)$ is the compression component, $\mathrm{MI}(C, Y)$ is the distortion component, and β is the trade-off parameter representing how much value distortion has relative to compression. An alternating minimization scheme for Eq. 2.29 is given in [148]. While IBN is a highly principled formulation of clustering, it relies on the existence of this 'important' information variable Y, and knowing the joint distribution between X and Y.

Slonim and Tishby [140] present an application of IBN to clustering of a document by term matrix. In their IB_{double} algorithm they use IBN to produce word clusters. These word clusters are then used as the Y in clustering the document by term matrix using IBN. One way IBN and an agglomerative hierarchical use of IBN are also presented in their work, but IB_{double} seems to fair better than these on their datasets. Dhillon et al. [43] give an IBN approach to simultaneous clustering of objects and features, and another divisive hierarchical application of IBN to feature clustering [42], both of which are applied to text.

In terms of more general information theoretic clustering, Banerjee et al. [12] present co-clustering using information theory and the Bregman divergences. Bekkerman et al. [17] investigate using information theory in multi-way clustering. Their approach seems to be particularly effective for clustering, being better than any of the previous information theoretic clustering algorithms discussed here.

It is worth noting that many authors of information theoretic papers that involve text clustering use non-standard text representations (Slonim and Tishby [140] being a notable exception). Specifically, they usually represent their text objects as probability distributions over features, often doing this for every clustering algorithm, be it information theoretic or not. While probability distributions make sense for information theoretic clustering algorithms, such representations are close to length normalized raw count information, which has been shown to be a less than ideal feature weighting in document clustering [155].

2.4.8 Maximum Margin Clustering

Support Vector Machines (*SVMs*) are well-known and powerful tools for supervised learning. Broadly speaking, given a dataset with known classes, they conceptually map datasets to feature spaces where the classes of the datasets may be partitioned using a simple linear hyperplane. The *margin* of the hyperplane is the distance from the hyperplane to its *support vectors*, the objects of the dataset which 'hold' the hyperplane in place. As the margin of a SVM is something to maximize, SVMs are often called *maximum margin classifiers.* With respect to an SVMs' use in clustering, the function of a clustering algorithm may be phrased as discovering the true labeling of a dataset. Given that standard SVMs require preexisting labels, their application to clustering has, unsurprisingly, required some clever manipulation.

One of the earliest applications of an SVM-like procedure to clustering was Support Vector Clustering [18] (*SVC*). In SVC, the entire dataset is mapped to a larger feature space using a Gaussian kernel mapping function. The smallest sphere possible enclosing the dataset in the larger space is computed. This sphere is then mapped back to the original space where it produces some number of disconnected ‘contours’. All points contained within the same contour are assigned to the same cluster.

Xu et al. [161] present the use of SVMs in binary clustering in an approach called Maximum Margin Clustering (MMC). They formulate the binary clustering task as one where they are seeking a clustering such that the soft margin between the two clusters is maximized, subject to a class size balance constraint. Their particular formulation of binary clustering can be seen as largely ignoring within cluster scatter. Despite this, MMC is shown to be highly effective, if extremely slow, in their paper, and it is the inspiration for most current SVM clustering work. In a later work [149], Generalized Maximum Margin Clustering (GMMC) is presented as overcoming some of MMC's effectiveness limitations as well as being significantly faster. However, it can still be seen to be very slow [168]. Zhang et al. [168] identity the source of these previous approaches' slowness as reliance on SDP (semi-definite programming), and develop a method for using support vector regression (IterSVR) that is orders of magnitude faster than either of the previous approaches. It should be noted that their approach is still many times slower than other clustering methods, making its application to text datasets difficult, and it requires bootstrapping of an initial clustering result to work. Zhao et al. [169], however, do present an SVM clustering algorithm that works rather quickly on decently sized text datasets (by exploiting feature sparsity).

As suggested in Zhang et al. [168], any of the approaches in the previous paragraph can be made to produce k-clusterings by using them to perform recursive binary splits. Considering that those papers mostly present results based on binary clustering tasks, it would be highly informative to test using those methods to generate k-clusterings, $k > 2$ in this manner and compare the results to other non-SVM methods of obtaining k-clusterings. With respect to directly obtaining a k-way clustering, Zhao et al. [170] illustrate a multi-class support vector clustering method that outperforms other SVM methods as well as N-Cut and k-means using Lloyd's method.

As with information theoretic clustering approaches, those SVM clustering methods that have been applied to text seem to not use standard text weightings method for their competitors or themselves. In this case, such a choice is rather curious, as it has been shown that *idf* weighting is useful for SVMs, at least when they are applied to classification [157].

Chapter 3

Clustering Quality Measures

In this chapter I focus on clustering quality measures. I define many common clustering quality measures and mention some of their more notable aspects.

I divide clustering quality measures into two classes: internal measures and external measures. For clarity I reiterate the definition of an ICQM that I supplied in Chapter 1. An ICQM is a clustering quality measure that uses only information contained within a clustering to evaluate it. *Measures that use any other information are referred to as external clustering quality measures, hereafter denoted as ECQMs.* I further divide the two larger classes into smaller conceptual groups. *Note that just because measures fall into the same conceptual group does not mean that they behave similarly in practice.* For discussions on how the measures actually behave in use readers should consult Chapter 4; there I go over most important clustering quality measure properties in detail and how most of the measures discussed here behave relative to them.

The notation I use for clustering quality metrics is as follows: Let M be a clustering quality metric. I denote its application to a clustering C as $M(C, *)$, where * is a comma separated list of parameters that M takes in addition to C. When in the parameter list, X is always the same dataset on which C was created, and d is always some distance function. Unlike X, d may be entirely unrelated to any distance function(s) used to generate C, although commonly it is related. T is a always a gold standard of X, t is the number of classes in T, and k is the number of clusters in C. c_i is the ith cluster of C, and t_j is jth class of T. $x_i \sim_C x_j$ denotes the set of all pairs of objects in X such that they share the same cluster in C. $x_i \nsim_C x_j$ denotes the set of all pairs of objects in X such that they are in different clusters in C.

A review/reference source for many older clustering quality measures, a number of which are very common in current clustering research, can be found in Milligan and Cooper [112]. Halkidi et al. [67, 68, 69] offer more recent discussions/general overviews of evaluation in clustering. Axiomatic discussions of clustering quality measures are also available, such as those in Ackerman and Ben-David [1, 3], and Meilă [109].

3.1 Internal Clustering Quality Measures

The universal strength of ICQMs is just that—their internal nature. Because ICQMs do not rely on information external to a clustering when assessing it, they are conceptually clear and can be applied in true exploratory data analysis situations. This functionality comes at a cost though: biases are coded directly into ICQMs in lieu of using external information.

3.1.1 Within-Cluster Scatter Measures

The most straightforward notion of clustering quality is that a good clustering is one with clusters exhibiting high internal consistency. The simplest ICQM formalization of this concept is *within cluster scatter (WCS)*:

$$\mathrm{WCS}(C, X, d) = \sum_{x_i \sim_C x_j} d(x_i, x_j), \tag{3.1}$$

Clusterings minimizing WCS are considered better. Variants of WCS include:

$$\mathrm{WWCS1}(C, X, d) = \sum_{c_i \in C} \frac{\sum_{x_j, x_k \in c_i} d(x_j, x_k)}{2|c_i|}, \tag{3.2}$$

$$\mathrm{WWCS2}(C, X, d) = \sum_{c_i \in C} \frac{\sum_{x_j, x_k \in c_i} d(x_j, x_k)}{2|c_i|(|c_i| - 1)}, \tag{3.3}$$

$$\mathrm{aWCS}(C, X, d) = \frac{\sum\limits_{x_i \sim_C x_j} d(x_i, x_j)}{|x_i \sim_C x_j|}. \tag{3.4}$$

WWCS1 is the classic k-means objective function if d is Euclidean distance. WWCS2 weights every cluster equally in its final measure, regardless of the percentage of objects contained within each cluster. aWCS is the average distance of objects in the same cluster.

ICQMs like those in Eqs. 3.1-3.4 are/have been very popular in clustering research, both as clustering quality measures and as the basis for clustering algorithms. Unfortunately, they have notable shortcomings. Besides not considering distance between clusters, basic WCS measures improve as the number of clusters increases, with a perfect clustering often being one with all singleton clusters (Eq. 3.1-3.4 yield 0 in such a case); this necessitates adjustments if they are to be used to compare clusterings with varying values of k. Examples of such adjustments can be found in the gap statistic [147] (which uses WWCS1) and the weighted gap statistic [164].

3.1.2 Between-Cluster Scatter Measures

Between cluster scatter (BCS), defined as:

$$\mathrm{BCS}(C, X, d) = \sum_{x_i \not\sim_C x_j} d(x_i, x_j), \tag{3.5}$$

and variants of BCS such as *average between cluster scatter (aBCS)*, defined as:

$$\mathrm{aBCS}(C, X, d) = \frac{\sum_{x_i \not\sim_C x_j} d(x_i, x_j)}{|x_i \not\sim_C x_j|}, \tag{3.6}$$

can be used as ICQMs. Such measures can be thought of as complements to WCS measures; however, unlike WCS measures, they are not commonly used by themselves as ICQMs. An example of an exception to this is Ratio-Cut, which can take the form of (Ackerman and Ben-David [1]):

$$\text{Ratio-Cut}(C, X, d) = \sum_{(x_i, x_j) \in x_i \not\sim_C x_j} \frac{1}{d(x_i, x_j)^2} \sum_{c_i \in C} \frac{1}{|c_i|} \tag{3.7}$$

when used as an ICQM; smaller Ratio-Cuts indicate better clusterings.

3.1.3 Measures Using all Within and Between-Cluster Distances

Considering all within and between cluster distances when assessing the quality of a clustering is arguably the most sensible thing for an ICQM to do, and there are many ICQMs that do this. I detail some ICQMs of this variety below.

WCS measures are often combined with BCS measures to create new ICQMs. The simplest form of this in active use is probably B/W, obtained by diving aBCS by aWCS:

$$\mathrm{B/W}(C, X, d) = \frac{\mathrm{aBCS}(C, X, d)}{\mathrm{aWCS}(C, X, d)}. \tag{3.8}$$

Higher B/W is better. N-Cut, when rephrased as an ICQM of the form (Ackerman and Ben-David [1]):

$$\text{N-Cut}(C, X, d) = \sum_{(x_i, x_j) \in x_i \not\sim_C x_j} \frac{1}{d(x_i, x_j)^2} \sum_{c_i \in C} \frac{1}{\sum_{x_j, x_l \in c_i, x_j \neq x_l} \frac{1}{d(x_j, x_l)^2}}, \tag{3.9}$$

can be seen as a direct combination of a between-cluster scatter component:

$$\sum_{(x_i, x_j) \in x_i \not\sim_C x_j} \frac{1}{d(x_i, x_j)^2},$$

and a within-cluster scatter component:

$$\sum_{c_i \in C} \frac{1}{\sum_{x_j, x_l \in c_i, x_j \neq x_l} \frac{1}{d(x_j, x_l)^2}}.$$

Note that how distance is used in the translation of graph cuts such as N-Cut and Ratio-Cut to ICQMs is flexible, here I am simply using it in the form $1/d(x_i, x_j)^2$, as suggested by Ackerman and Ben-David [1]. As with Ratio-Cut, smaller values from N-Cut indicate better clusterings.

An interesting approach to limiting the influence a few distances/objects can have on a clustering's quality can be found in the Gamma Index [112], defined as:

$$\text{Gamma}(C, X, d) = \frac{S_{+}(C, X, d) - S_{-}(C, X, d)}{S_{+}(C, X, d) + S_{-}(C, X, d)}. \tag{3.10}$$

$S_{+}(C, X, d)$ is the number of pairs $x, y \in X$ and $w, z \in X$ such that x and y are in the same cluster in C, w and z are in different clusters in C, and $d(x, y) < d(w, z)$. $S_{-}(C, X, d)$ is the opposite of $S_{+}(C, X, d)$. Higher Gamma values are better. By using counts, and not raw distances, the impact any small group of objects can have on the final score is limited. While this property is certainly good, one side effect of this limiting aspect is the counterintuitive concept that making clusters more compact/well separated may not have any impact on the final score.

3.1.4 Measures using Selective Distances

In the previous section, I looked at ICQMs that use *all* within and between cluster distances in their measurement; an alternative to this is to selectively use *some* within and between cluster distance. While allowing some distances to be ignored might be dangerous, it greatly increases the concepts of clustering quality that we can measure. Some measures that fall in to this category are the Silhouette statistic, the Dunn Index [53], the Davies-Bouldin Index [37], and the C-Index [83].

The Silhouette statistic [133] can be used as an ICQM that compares each object's between and within-cluster distances, and aggregates these individual comparisons into a full ICQM. For some $x_i \in X$ and a clustering C of X, the Silhouette of x_i with respect to C is defined as:

$$\text{Silhouette}(x_i, C, X, d) = \frac{d_{\nsim}(x_i) - d_{\sim}(x_i)}{\max\left(d_{\nsim}(x_i), d_{\sim}(x_i)\right)}, \tag{3.11}$$

where $d_{\sim}(x_i)$ is the average distance of objects in the same cluster as x_i to x_i, and $d_{\nsim}(x_i)$ is the average distance of x_i to objects in the closest other cluster. Silhouette(x_i, C) is aggregated into an ICQM as follows:

$$\text{Silhouette}(C, X, d) = \frac{1}{n} \sum_{x_i \in X} \text{Silhouette}(x_i, C, X, d). \tag{3.12}$$

Let W_i be some within-cluster scatter type measure of c_i, and let $M_{i,j}$ be a measure of separation between clusters c_i and c_j, then the Dunn Index is defined as:

$$\text{Dunn}(C, X, d) = \min_{c_i \in C}\left(\min_{c_j \in C, c_j \neq c_i} \left(\frac{M_{i,j}}{\max_{c_k \in C} W_k} \right)\right). \tag{3.13}$$

Typical definitions for $M_{i,j}$ and W_k result in:

$$\text{Dunn}(C, X, d) = \min_{c_i \in C}\left(\min_{c_j \in C, c_j \neq c_i} \left(\frac{d(v_i, v_j)}{\max_{c_k \in C} \frac{\sum_{x_l \in c_k} d(x_l, v_k)}{|c_k|}} \right)\right), \tag{3.14}$$

where v_i is the centroid of cluster c_i, and $|c_i|$ is the number of objects in cluster i.

Higher values are better for the Dunn Index. Because the Dunn Index does a form of worst case evaluation, it is fragile. If even one pair of clusters are poorly separated, the measure can be arbitrarily large, even if the rest of the clustering is good. The C-Index, which is less fragile, is defined as:

$$\text{C-Index}(C, X, d) = \frac{\text{WCS}(C, X, d) - S_{min}(X, l)}{S_{max}(X, l) - S_{min}(X, l)}, \tag{3.15}$$

where $l = |x_i \sim_C x_j|$, $S_{min}(X, l)$ is the minimum l distances over all pairs of objects in X, and $S_{max}(X, l)$ is the maximum l distances over pairs of objects in X. Note that the C-Index incorporates between cluster distances indirectly (we may reasonably assume that $S_{max}(X, l)$ consists mostly of between cluster distances). The C-Index handles cluster proximity in a reasonable manner.

Using the notation from the Dunn Index, the Davies-Bouldin Index [37] is defined as:

$$\text{Davies-Bouldin}(C, X, d) = \frac{1}{k} \sum_{c_i \in C} \max_{c_j \in C, c_j \neq c_i} \frac{W_i + W_j}{M_{i,j}}. \tag{3.16}$$

Typical definitions for W_i and $M_{i,j}$ result in this becoming:

$$\text{Davies-Bouldin}(C, X, d) = \frac{1}{k} \sum_{c_i \in C} \max_{c_j \in C, c_j \neq c_i} \frac{\frac{\sum_{x_l \in c_i} d(x_l, v_i)}{|c_i|} + \frac{\sum_{x_l \in c_j} d(x_l, v_j)}{|c_j|}}{d(v_i, v_j)}. \tag{3.17}$$

Lower Davies-Bouldin scores indicate superior clusterings. Like the Dunn Index, the Davies-Bouldin Index does a form of worst case evaluation.

Margin-based ICQMs, which have only recently become common as SVMs have gained in precedence; are another example of ICQMs that use selective distances. A notable margin-based ICQM is *relative margin (RM)*, which can be found in Ackerman and Ben-David [1]. Let a *representative set* of C, denoted $R(C)$, be defined as any set of objects from X that contains exactly one object of every cluster in C. Let $R(C)^*$ be the set of all representative sets of C. Then the relative margin of C is:

$$\text{RM}(C, X, d) = \min_{R(C) \in R(C)^*} \frac{\sum_{x_i \in X \setminus R(C)} \min_{x_j, x_l \in R(C), x_j \neq x_l} \left(\frac{d(x_i, x_j)}{d(x_i, x_l)}\right)}{k}. \tag{3.18}$$

Relative margin looks for a representative set of points from the clusters in C that splits up the rest of the dataset by the minimum average margin ratio (closest representative distance divided by second closest representative distance) possible. Unsurprisingly, this seems very much like what a SVM does. Additive margin [1] is another margin-based ICQM that is similar to relative margin, as is point-wise margin, a new ICQM which I design and discuss in Chapter 6. Due to the newness of margin-based measures, there are not many comparisons of their efficacy relative to older ICQMs.

Various ICQMs exist for soft clusterings (the Xie-Beni Index [160], the PBM Index [122], etc.), many of which are direct extensions of the various measures from the sections above.

3.1.5 Model-fitting Measures

Model fitting clustering quality measures are a highly distinct alternative to the measures above. I place them in this section because, given fixed model types and estimation methods, they operate much more like ICQMs than ECQMs in that they represent clear, specific notions of clustering quality.

I denote the likelihood that the model represented by a clustering C generated the dataset X as L_C. Let p be the number of free parameters in the model represented by C. The *Bayesian Information Criterion* score [137] of C is then defined as:

$$\mathrm{BIC}(C, X, *) = -2\ln(L_C) + p\ln(n). \tag{3.19}$$

Lower BICs are better. BIC balances the likelihood of the model with the number of free parameters it contains. The general concept behind it is that a good model will have high likelihood, but will also require as few parameters as possible (as fitness of models naturally increases as the number of parameters increases). A score related to BIC, the

Akaike Information Criterion score [75], is defined as

$$\mathrm{AIC}(C, X, *) = -2\ln(L_C) + 2p. \tag{3.20}$$

AIC is similar to BIC, but penalizes free parameters less severely (it favours solutions with more clusters relative to BIC). X-means [125] shows the use of both AIC and BIC in clustering. AIC and BIC are by no means the only model-fitting evaluation techniques, Hamerly et al. [70, 55] give examples of other fitness measures that can be used while clustering.

Obviously, the selection of model type/estimation method is of pivotal importance in making good use of model fitting clustering quality measures. If a clustering does not conform to the models used to represent it when being evaluating, the likelihood of the model, and therefore the final score itself, will be poor.

3.2 External Clustering Quality Measures

ECQMs use information outside of the clustering in assessing it, with the information usually taking the form of a gold standard. ECQMs are useful measures of relative quality and/or consistency, but are not really suitable for use in determining general quality as they use arbitrary external information. Despite this, ECQMs are often used to show which clusterings/clustering algorithms are superior in general.

The ECQMs I discuss here are all based on comparisons between clusterings/gold standards where, as before, a gold standard is a set of 'true' classes/groups for a dataset. Before giving exact definitions, it should be noted that those measures are sometimes discussed as being *between two partitions*. However, when actually being used in clustering evaluation, they are used *between a clustering and a clustering or a clustering and a gold standard*. Measures below that are symmetric with respect to their parameters may be used in either situation, while non-symmetric measures are typically used only for the latter. That said, I present all ECQMs from the perspective of being between a gold standard (T) and a clustering (C). Note that unlike ICQMs, the ECQMs I present make no use of the actual vectors of X, only cluster membership ids and gold standard memberships.

3.2.1 Mutual Information Based Measures

Perhaps no concept is so pervasive in ECQMs as *mutual information (MI)*. Typically, maximum likelihood estimates of $p(c_i) = \frac{|c_i|}{n}$, $p(t_i) = \frac{|t_i|}{n}$, $p(c_i, t_j) = \frac{|c_i \cap t_j|}{n}$ are used in MI type measures. $\mathrm{MI}(C, T)$, the mutual information between C and T, is defined as:

$$\mathrm{MI}(C,T) = \sum_{t_i \in T} \sum_{c_j \in C} p(c_j, t_i) \log\left(\frac{p(c_j, t_i)}{p(c_j)p(t_i)}\right), \tag{3.21}$$

As its name suggests, mutual information measures how much information C tells us about T, and vice versa.

NMI is by far the most common ECQM using MI, and possibly the most common clustering quality measure in all clustering research. One version of NMI [146] is defined as:

$$\mathrm{NMI}(C,T) = \frac{\mathrm{MI}(C,T)}{\sqrt{\mathrm{H}(C)\mathrm{H}(T)}}, \tag{3.22}$$

where $\mathrm{H}(C)$ is the entropy of C:

$$\mathrm{H}(C) = \sum_{c_i \in C} p(c_i) \log(p(c_i)). \tag{3.23}$$

Alternative definitions of NMI include:

$$\mathrm{NMI}(C,T) = \frac{\mathrm{MI}(C,T)}{\min(\mathrm{H}(C), \mathrm{H}(T))}, \tag{3.24}$$

and:

$$\mathrm{NMI}(C,T) = \frac{\mathrm{MI}(C,T)}{\mathrm{H}(C,T)}, \tag{3.25}$$

where $H(C,T)$ is the joint entropy of C and T, defined as:

$$\mathrm{H}(C,T) = \sum_{c_i \in C} \sum_{t_j \in T} p(c_i, t_j) \log(p(c_i, t_j)). \tag{3.26}$$

NMI variants have a value between zero and one and are symmetric, with higher values indicating C is a better clustering.

While every NMI variation uses mutual information, their normalizations of it are distinct and result in markedly different behaviors. Eq. 3.22 favours clustering with close to the number of true classes, being able to obtain a value of one only when the number of clusters is equal to the number of true classes. This is not the case for Eq. 3.24 though, as it scales by minimum entropy. Using minimal entropy results in clusterings that are strict refinements of the true labeling (i.e. no cluster contains more than one type of true label) scoring highly, which might be a useful property, but it also results in the awkward situation of having a set of singleton clusters always obtaining $\mathrm{NMI}(C,T) = 1$ for any T.

Newer information theoretic measures using mutual information include *Variation of Information* (*VI*) [110] and various adjusted information theoretic measures [151]. VI is defined as:

$$\mathrm{VI}(C,T) = \mathrm{H}(C) + \mathrm{H}(T) - 2\mathrm{MI}(C,T). \tag{3.27}$$

VI has numerous benefits including a strong information theoretic backing like NMI and the fact that it is a proper distance function (as discussed in Section 2.3). Unfortunately, the upper bound of VI increases with the number of different clusters/classes being evaluating (e.g., more information can be shared between more complicated structures simply because they contain more information). Normalizing to account for this is straightforward, but this breaks many of the properties that make VI appealing.

Adjusted information theoretic measures, as discussed by Vinh et al. [151], account for correlation by random chance in an information theoretic measure. Vinh et al. show that this is especially necessary when the number of clusters/classes is large relative to the number of objects in the dataset. Adjusted Mutual Information (*AMI*), which can take forms such as:

$$\mathrm{AMI}(C,T) = \frac{\mathrm{MI}(C,T) - E[\mathrm{MI}(C,T)]}{\max(\mathrm{H}(C),\mathrm{H}(T)) - E[\mathrm{MI}(C,T)]}, \tag{3.28}$$

is an example of a corrected information theoretic measure in relatively common use, where $E[\mathrm{MI}(C,T)]$ is the expected value of $\mathrm{MI}(C,T)$ by chance.

V-measure [131] is another interesting information theoretic measure that was explicitly presented as focusing on the concepts of homogeneity and completeness (see Section 4.4). V-measure is defined as:

$$\text{V-measure}(C,T) = \frac{(1+\beta)\text{homogeneity(C,T)} * \text{completeness(C,T)}}{(\beta * \text{homogeneity(C,T)}) + \text{completeness(C,T)}}. \tag{3.29}$$

β is a real-valued parameter of the measure. homogeneity(C,T) is defined as:

$$\text{homogenetiy(C,T)} = \begin{cases} 1, & \text{if } H(T|C) = 0; \\ \frac{H(T|C)}{H(T)}, & \text{otherwise.} \end{cases} \tag{3.30}$$

completeness(C,T) is defined as:

$$\text{completeness(C,T)} = \begin{cases} 1, & \text{if } H(C|T) = 0; \\ \frac{H(C|T)}{H(C)}, & \text{otherwise.} \end{cases} \tag{3.31}$$

Section 4.4 briefly explains how homogeneity and completeness are closely linked with information retrieval's precision and recall concepts. This, combined with the fact Eq. 3.29 is like F-measure [150], suggests that one may interpret V-measure as an information theoretic version of FQ (Eq. 3.45). Note that V-measure has not seen as widespread use as the previously discussed measures.

Although it does not use MI, cluster entropy is conceptually similar enough to measures using MI that I include it here. The cluster entropy of some cluster c_i, relative to a true labeling T, is defined as:

$$\text{Entropy}(c_i, T) = \sum_{t_j \in T} \frac{p(c_i, t_j)}{p(c_i)} \log(\frac{p(c_i, t_j)}{p(c_i)}). \tag{3.32}$$

This can be aggregated in a simple fashion to form a full ICQM [173]:

$$\text{EQ}(C, T) = 1 - \sum_{c_i \in C} p(c_i) \frac{1}{-log(q)} \text{Entropy}(c_i, T), \tag{3.33}$$

	Pairs with Same Labels in C	**Pairs with Different Labels in C**
Pairs with Same Labels in T	$L(C,T)$	$D'(C,T)$
Pairs with Different Labels in T	$D(C,T)$	$L'(C,T)$

Table 3.1: Notation for the major kinds of pair-counts used in pair-counting ECQMs.

where q is the number of classes in T. While EQ is similar to NMI, it does not consider size balance as NMI does. Consider some T and C of X where $p(t_1) = 0.95$, $p(t_2) = 0.05$, and C has two clusters c_1 and c_2 such that both clusters are 95% label t_1 and 5% label t_2. We can see that $\text{NMI}(C,T) = 0$ (the worst possible), yet $\text{EQ}(C,T) \sim 1$ (almost the best possible). Assuming all classes are relatively close in size, the difference between NMI and EQ diminishes, but for unbalanced sizes, one may prefer NMI as it considers size balance. For further discussion on this readers should consult Section 4.8.

3.2.2 Pair-Counting Measures

Pair-counting ECQMs, as their name suggests, are measures that use counts of pairs of objects in their quality assessments. The pair counts typically used by pair-counting ECQMs are given Table 3.1. Note that while it is common to see just $L(C,T)$ and $L'(C,T)$ in pair-counting measures, readers should be aware that measures that use those implicitly use $D(C,T)$ and $D'(C,T)$, as we have $L(C,T)+L'(C,T)+D(C,T)+D'(C,T) = \binom{n}{2}$. Example of pairs counting measures are the Rand Index [128], the Adjusted Rand Index [82], the Jaccard Index [85], and the Fowlkes-Mallows Index [57].

In the context of ECQMs, the Rand Index is defined as:

$$\text{Rand}(C,T) = \frac{\text{L}(C,T) + \text{L}'(C,T)}{\binom{n}{2}}. \tag{3.34}$$

This base form of the Rand Index is not commonly used; it has been superseded by its adjusted form below.

The Jaccard Index is defined as:

$$\text{Jaccard}(C,T) = \frac{\text{L}(C,T)}{\binom{n}{2} - \text{L}(C,T)}. \tag{3.35}$$

It is interesting to note that the Jaccard Index makes no use of $L'(C,T)$, whereas Rand Index uses both $L(C,T)$ and $L'(C,T)$; this is akin to the difference we saw between measures using only within or between cluster versus those that consider both in the ICQM section. The Fowlkes-Mallows Index [57] is very similar to the Rand and Jaccard Indices.

The Adjusted Rand Index is an alteration of the Rand Index to compensate for correlation by random chance, providing a stronger measure. It is defined as:

$$\text{ARI}(C,T) = \frac{\text{Rand}(C,T) - E[\text{Rand}(C,T)]}{\max(\text{Rand}(C,T)) - E[\text{Rand}(C,T)]}, \tag{3.36}$$

where:

$$E[\text{Rand}(C,T)] = \frac{\sum\limits_{c_i \in C} \binom{|c_i|}{2} \sum\limits_{t_j \in T} \binom{|t_j|}{2}}{\binom{n}{2}} \tag{3.37}$$

and:

$$\max(\text{Rand}(C,T)) = 0.5(\sum_{c_i \in C} \binom{|c_i|}{2} + \sum_{t_j \in T} \binom{|t_j|}{2}). \tag{3.38}$$

Note that the correction for random chance applied in Eq. 3.36 is identical to the correction used in Eq. 3.28. Assuming that M is a distance function between any two partitions C and T of some dataset X, it has a correction for random chance of the form [82]:

$$\text{Chance-Correction}(M,C,T) = \frac{\text{M}(C,T) - E[\text{M}(C,T)]}{\max(\text{M}(C,T)) - E[\text{M}(C,T)]}. \tag{3.39}$$

Obtaining $E[\mathrm{M}(C,T)]$ for a particular ECQM may be rather involved, as it is with adjusted mutual information [151], but this is a powerful and useful correction regardless. ARI is an extremely common ECQM and is regarded as being fairly robust.

Measures such as the Rand, Jaccard, and Fowlkes-Mallows Indices are easily extended to deal with soft clusterings [25, 26, 120, etc.].

3.2.3 Matching Measures

Some ECQMs use matches of clusters to classes in their quality assessments. The nature of the matching sought is highly variable (see the following chapter for a discussion about this). Below I list some of the most common matching measures in use.

Accuracy is an interesting ECQM. Let Z be a bijection of C on to T (a one to one and onto mapping of clusters to true labels). Let P_Z be the set of all possible Z with respect to a given C and T, then the *accuracy* of a C with respect to T is:

$$\mathrm{Accuracy}(C,T) = \max_{Z \in P_Z} \sum_{(c_i,t_j) \in Z} p(c_i, t_j). \tag{3.40}$$

Obviously, given the definition above, accuracy requires that the number of clusters and classes are equal. Accuracy requires $O(k^3)$ time to compute (using the *Hungarian method* [98] to solve Z directly). Care is advisable when interpreting a clustering paper that says they are using accuracy, as they may in fact be using a purity related measure (see below).

Accuracy is usually used in a somewhat different fashion than most other clustering quality measures when comparing clustering algorithms. A typical procedure for selecting the 'best' clustering algorithm using some generic clustering quality measure might be:

1. Select the candidate clustering algorithms.
2. Select some number of datasets.
3. For each dataset, cluster it using each clustering algorithm for some variety of number of clusters. Apply the chosen clustering quality measure to each clustering.
4. The best clustering algorithm is the one that performs the best, on average.

Accuracy is usually applied in a different manner than above as it requires the number of clusters to be equal to the number of classes. A typical clustering evaluation procedure using accuracy might look like:

1. Select the candidate clustering algorithms.
2. Select some number of datasets.
3. For each dataset, for some range of k, sample k different true classes of the dataset, and cluster those in to k clusters. Apply accuracy to evaluate the quality of each result.
4. The best clustering algorithm is the one that performs the best, on average.

This second procedure is arguably problematic as it only evaluates clusterings that have the true numbers of clusters in them, and we do not, in general, know the true number of clusters prior to clustering.

Edit distances such as Hamming distance [71] are applicable in clustering evaluation, although they require a procedure similar to that done with accuracy. Using the notation from Eq. 3.40, a Hamming distance ECQM can be defined as:

$$\text{Hamming}(C,T) = 1 - \text{Accuracy}(C,T), \tag{3.41}$$

when C and T have equal numbers of clusters/classes.

F-measure ECQMs use F_β as their basis:

$$F_\beta(c_i,t_j) = (1+\beta^2)\frac{\text{Precision}(c_i,t_j) * \text{Recall}(c_i,t_j)}{\beta^2\text{Precision}(c_i,t_j) + \text{Recall}(c_i,t_j)}, \tag{3.42}$$

where:

$$\text{Precision}(c_i,t_j) = \frac{p(c_i,t_j)}{p(c_i)} \tag{3.43}$$

$$\text{Recall}(c_i,t_j) = \frac{p(c_i,t_j)}{p(t_j)}. \tag{3.44}$$

F-measure was developed for use in information retrieval [150]. F_β allows one to tune the relative value that precision has compared to recall via the β parameter. In practice, F_1 ($\beta = 1$) is used almost universally when F-measures are applied to evaluating clusterings, although one can easily envision a case in which valuing precision over recall or vice versa is appropriate. In terms of a C and T, F_β deals only with a single (c_i,t_j) pair, F_β must therefore be aggregated in some fashion over all such pairs to produce a full evaluation measure. The most common form of aggregation is:

$$\text{FQ}(C,T) = \sum_{t_j \in T} p(t_j) \max_{c_i \in C} F_1(c_i,t_j), \tag{3.45}$$

where F_1 is F_β with $\beta = 1$. This version of F-measure looks for the best c_i to represent each t_j. The individual F_1 scores are weighted by their true label class sizes in the final scoring. As with NMI, FQ is between zero and one with higher scores indicating a better clustering.

A strength of FQ and similar measures is that they are based on well understood information retrieval concepts (precision and recall) that, unlike some other definition of quality, are known to correspond to real user notions of quality, and the ability to weight those aspects differently. One potential issue with F-measures like Eq. 3.45 is, as many classes may map to a single cluster, but each individual class only maps to one cluster, there may be 'classless' clusters. A classless cluster does not effect the score of this measure, leading one to question how good of an evaluation of a clustering's overall quality the measure is. Another issue is that minimal and maximal F-measure scores are heavily dependent on the size and number of true classes relative to the number of clusters in the solution. It is often not possible to obtain an F-measure score anywhere near zero or one for a particular dataset with a fixed number of clusters. Nevertheless, F-measures are popular in text clustering evaluation [16, 145, 172].

Cluster purity, commonly defined as:

$$\text{Purity}(c_i, T) = \max_{t_j \in T} \frac{p(c_i, t_j)}{p(c_i)} \tag{3.46}$$

measures the maximum precision possible for cluster c_i over any t_j. A simple conversion of purity into a matching ECQM is [171]:

$$\text{PQ}(C, T) = \sum_{c_i \in C} p(c_i)\text{Purity}(c_i, T), \tag{3.47}$$

PQ is different from FQ: every cluster contributes to the final score, but every true label type may not (in cardinality terms, PQ is 1 class to 0 or more clusters, FQ is 1 cluster to 0 or more classes, and accuracy is 1 class to 1 cluster).

It is notable that FQ and accuracy (as presented here) are easily applicable to hierarchical clusterings with slight modifications. This same property holds for many other formalizations of clustering quality. PQ can, with some work, be used in hierarchical definitions of clustering quality. However; standard NMI is not amenable to use on hierarchical clusterings.

Chapter 4

Properties of Clustering Quality Measures

While the previous chapter served to define a large number of the clustering quality measures in common use, this chapter focuses on general properties of clustering quality measures. I discuss properties that are important to ICQMs and ECQMs, covering many of those discussed in previous work on clustering evaluation [3, 95, 110, 111, 112, 151, etc.] as well as some others. By analyzing how clustering quality measures behave relative to the properties I discuss here, users can be provided with both theoretical and practical reasons to use certain measures. Although each property has its own section below, careful investigation can show that many of them are related. Table 4.1, included at the end of this section, presents how many of the clustering quality measures discussed in Chapter 3 behave relative to most of the properties I discuss here. Readers should be aware that using alternative forms of the clustering quality measures I present in Table 4.1 may very well result in different behavior from the clustering quality measures with respect to the properties in this chapter. For example, if I want to use object-pair distances in the Davies-Bouldin Index, I must do an entirely new analysis of its behavior with respect to the properties in this chapter. This is because the properties for Davies-Bouldin in Table 4.1 are based on Eq. 3.17, which uses object-centroid distances.

4.1 Concept

Every clustering quality measure is based on some particular concept of what makes a clustering good. Internal consistency of clusters, as exemplified by the classic k-means objective function (Eq. 3.2), is the most common quality concept, but there are many others. Each concept is often distinct from others; resulting in clustering quality measures based on different concepts behaving very differently. Even clustering quality measures based on exactly the same concept of what makes a clustering good (such as all the varieties of WCS I present) can have highly disparate opinions of which clusterings are the best due to differences in implementation.

There are no right or wrong concepts/implementations in this context, it is simply up to users to ensure that the clustering quality measures they use are consistent both in concept and implementation with what their specific needs are. For example, if a user has no reason to value between-cluster distances, then using relative margin (Eq. 3.18) is inappropriate. On the other hand, if a user was going to use the clustering to train a classifier, then relative margin might be an excellent measure, as it is similar to a SVM classifier.

4.2 Consistency

Consistency, in the context of clustering quality measures, is informally used to refer to the concept that *improving* a clustering should improve ICQMs' scores for that clustering, or at the very least not make scores worse. Intuitively, this is a property every ICQM should have, but attempting to translate it to a formal definition quickly reveals non-trivial problems: What does improvement mean? Why should every ICQM agree on what is an improvement?

The most straightforward way to answer both questions is to make the assumption, as done by Ackerman and Ben-David [1], that object-pair distances are what indicate improvement. Specifically, shrinking within cluster distances, or expanding between cluster distances, improves a clustering. The authors use this concept to create a formal definition of the previous paragraph for an ICQM of the form $M(C, X, d)$ as follows.

Definition 1 (C Consistent Variant)**.** *Distance function d' is a C* consistent variant *for a fixed X, C, and d, if $\forall_{x_i,x_j \in x_i \sim_C x_j} d(x_i, x_j) \geq d'(x_i, x_j)$ and $\forall_{x_i,x_j \in x_i \nsim_C x_j} d(x_i, x_j) \leq d'(x_i, x_j)$.*

Definition 2 (Consistency)**.** *ICQM M is* consistent *if, for any X, C, d, and C* consistent variant *d', we have $M(C, X, d) \leq M(C, X, d')$ if higher Ms indicate a better C, otherwise we have $M(C, X, d) \geq M(C, X, d')$.*

Consistency seems reasonable enough that the authors originally suggest it as an axiom for ICQMs; unfortunately, it leads to some counterintuitive situations. Consider Fig. 4.1; while the right clustering is a consistent change to the left, it may very well be assessed by an ICQM as a worse clustering because two clusters are now optimal but the solution still has three.

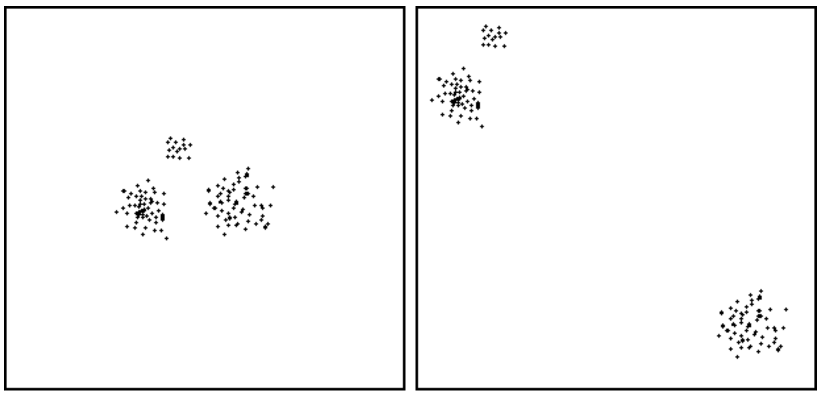


Figure 4.1: A consistent change to a 3-clustering.

Many of the counterintuitive properties of consistency are dealt with by *weak local consistency.*

Definition 3 (C Weakly Locally Consistent Variant)**.** *Distance function d' is a C* weakly locally consistent variant *for a fixed X, C, and d, if the following properties hold:*

1. *For all $c_i \in C$ there exists a constant $\lambda \leq 1$ such that for all $x_j, x_k \in c_i$ we have $d(x_j, x_l) \geq \lambda d'(x_j, x_l)$.*

2. *For all x_i, x_j in different clusters, we have $d(x_j, x_l) \leq d'(x_j, x_l)$.*

3. *There exists some set R, where R contains exactly one object from every cluster in C, such that for some constant $\lambda \geq 1$, for all $x_i, x_j \in R$ we have $d(x_i, x_j) \geq \lambda d'(x_i, x_j)$.*

Definition 4 (Weakly Locally Consistent)**.** *ICQM M is* weakly locally consistent *if, for any X, C, d, and C* weakly locally consistent variant *d', we have $M(C, X, d) \leq M(C, X, d')$ if higher Ms indicate a better C, otherwise we have $M(C, X, d) \geq M(C, X, d')$.*

Weak local consistency seems like a more suitable axiom for ICQMs than consistency.

However, it is very weak in the sense that it describes how an ICQM behaves in an extremely limited number of situations.

It is interesting to note that altering the $\geq$s and $\leq$s in both consistency and weak local consistency to $>$ and $<$ respectively, which translates to requiring that M improves, as opposed to being allowed to stay the same or improve, whenever the clustering improves, actually causes a huge number of ICQMs to fail to satisfy them. This leads us to a practical question: Which ICQMs always improve when a clustering improves? I expand this question further: Which ICQMs improve when a clustering's within cluster distances improve, and which ICQMs improve when a clustering's between cluster distance improve? I define properties below that address these questions.

Definition 5 (C Improved Within-Consistent Variant)**.** *Distance function d' is a C* improved within-consistent variant *for a fixed X, C, and d, if $\forall_{x_i,x_j \in x_i \sim_C x_j} d(x_i, x_j) \geq d'(x_i, x_j)$ and $\exists_{x_i,x_j \in x_i \sim_C x_j} d(x_i, x_j) > d'(x_i, x_j)$.*

Definition 6 (C Improved Between-Consistent Variant)**.** *Distance function d' is a C* improved between-consistent variant *for a fixed X, C, and d, if $\forall_{x_i,x_j \in x_i \not\sim_C x_j} d(x_i, x_j) \leq d'(x_i, x_j)$ and $\exists_{x_i,x_j \in x_i \not\sim_C x_j} d(x_i, x_j) < d'(x_i, x_j)$.*

Definition 7 (Improving Within Consistency)**.** *ICQM M is* improving within-consistent *if, for any X, non-trivial C, d, and C* improved within consistent variant *d', we have $M(C, X, d) < M(C, X, d')$ if higher Ms indicate a better C, otherwise we have $M(C, X, d) > M(C, X, d')$.*

Definition 8 (Improving Between Consistency)**.** *ICQM M is* improving between-consistent *if, for any X, non-trivial C, d, and C* improved between consistent variant *d', we have $M(C, X, d) < M(C, X, d')$ if higher Ms indicate a better C, otherwise we have $M(C, X, d) >$*

$M(C, X, d')$.

Improving within and between-consistency are strong characterizations of ICQMs. However, because they cause a partition of reasonable ICQMs, they are not ICQM axioms, only useful behaviors to understand. Note that I partly organized ICQMs in Chapter 3 by use of within and between cluster distance as well. Table 4.1 gives which ICQMs are improving within-consistent and which are improving between-consistent. No column is given for weak local consistency as every ICQM in the table is weakly locally consistent.

4.3 Fullness

All ECQMs of the form $M(C, T)$ are based on the joint distribution $p(c_i, t_j)$; even pair counting measures have this property. For instance, the $\mathrm{L}(C, T)$ in the Rand Index (Eq. 3.34) can be stated as

$$\mathrm{L}(C, T) = \sum_{c_i \in C} \sum_{t_j \in T} P^*(c_i, t_j, n),$$

where

$$P^*(c_i, t_j, n) = \begin{cases} \binom{p(c_i,t_j)n}{2}, & \text{if } p(c_i, t_j)n > 1; \\ 0, & \text{otherwise.} \end{cases}$$

Given the dependency of all ECQMs of the form $M(C, T)$ on $p(c_i, t_j)$ values, a natural question to ask is if all ECQMs use the entirety of that distribution, or do some ignore specific $p(c_i, t_j)$ values that fail to meet certain criteria. If we allow ECQMs to ignore some $p(c_i, t_j)$ values, then we are open to awkward situations such as entire clusters in some clusterings not effecting the clustering's quality assessment by ECQMs [155] (and possibly some classes not being a factor as well). Because of this, one may prefer ECQMs where every $p(c_i, t_j)$ has some impact for every C and T pairing. I refer to such ECQMs as *full*.

Definition 9 (Fullness). *ECQM M is* full *if, for any C and T pairing and accompanying joint distribution of $p(c_i, t_j)$, changing any single $p(c_i, t_j)$ value (changing one cluster or class membership), changes $M(C, T)$.*

Table 4.1 presents which ECQMs exhibit fullness. ECQMs adjusted for random chance (AMI and ARI) were not assessed for this property. All the ECQMs that fail to exhibit fullness fall into the matching measures category in Chapter 3; with every measure in that category using a best fit mechanic where it looks for the best matches between clusters and classes. The exact nature of the matching sought by these measures varies: accuracy (Eq. 3.40) and Hamming (Eq. 3.41) look for a bijection of classes to clusters, PQ (Eq. 3.47) allows many clusters to be matched to the same class, and FQ (Eq. 3.45) allows many classes to be matched to the same cluster. While fullness is usually a good property to have, it is entirely possible that matching patterns like these are more desirable in ECQMs than fullness for some application domains, thus I only suggest that users ensure that their ECQMs match desired behavior with respect to fullness (as opposed to requiring that all ECQMs they use exhibit it).

4.4 Homogeneity and Completeness

Homogeneity and *completeness* are ECQM concepts that have been used directly in several clustering evaluation works [9, 131, etc.] and indirectly in many others. Homogeneity refers to the concept that clusters should contain only one kind of true label, whereas completeness refers to the concept that all objects of one kind of true label should be in the same cluster. The former is closely related to precision, as it is used in information retrieval [150]. The latter is closely related to recall, again, as it is used in information retrieval.

An argument can be made that a sensible ECQM should do two things with respect to these concepts; 1) it should reward a clustering the more it exhibits these two concepts; and 2) a clustering exhibiting perfect homogeneity and completeness simultaneously should be given a score better than any other clustering.

Unfortunately, while seeming sensible (1) does not have as clean a formalization as a property like consistency; it can be implemented in many different ways. For example, V-measure (Eq. 3.29) is an ECQM based on information theoretic measurements of the two concepts, while Amigó et al. [9] suggest specific constraints that an ECQM must satisfy to be considered as respecting both homogeneity and completeness. Because of the variety of formalizations possible, I do not include detailed information on how ECQMs behave with respect to (1), although intuitively the majority of ECQMs can be said to satisfy (1) in one sense or another.

On the other hand, (2) has only one formalization: *a clustering that is a perfect copy of the true labeling must be the uniquely best clustering possible.* We can see that (2) is failed by some ECQMs. For example, PQ (Eq. 3.47) is based only on homogeneity. As long as clusters in a clustering are entirely homogenous, PQ will assign a perfect value for that clustering, regardless of the completeness of the clusters in the clustering. In general, any ECQM which does not enforce that the optimal number of clusters in a clustering is equal to the number of true label types will fail (2). The ECQMs which do satisfy this in Table 4.1 mostly satisfy (2).

4.5 Noise Tolerance

Noise tolerance is a pivotal property of any machine learning technique. With respect to clustering quality measures I use it to refer to slight changes in a clustering quality measure's parameters causing only slight changes in its output.

Some ICQMs have poorer noise tolerance than others; notable examples from this thesis are WWCS1 (Eq. 3.2), WWCS2 (Eq. 3.3), Davies-Bouldin (Eq. 3.17), Dunn (Eq. 3.14), and relative margin (Eq. 3.18). In general, poor noise tolerance may be present in any ICQM with the ability to weight individual distances/objects much more heavily than others in their computations.

Recall that the definition of WWCS1 is:

$$\mathrm{WWCS1}(C, X, d) = \sum_{c_i \in C} \frac{\sum_{x_j, x_k \in c_i} d(x_j, x_k)}{2|c_i|}.$$

Consider the addition of δ to one of the within-cluster distances used in computing WWCS1, then the new value of WWCS1 is:

$$\mathrm{WWCS1}(C, X, d) = \sum_{c_i \in C} \frac{\sum_{x_j, x_k \in c_i} d(x_j, x_k)}{2|c_i|} + \frac{\delta}{2|c_*|},$$

where $|c_*|$ is the size of the cluster containing the within-cluster distance that was increased. The change is scaled by $|c_*|$. Because of this large, changes in WWCS1 can occur with only a small change to within-cluster distances if they happen in small clusters. For example, when $|c_*| = 100$ the change is $\delta/200$, but for $|c_*| = 2$ it is $\delta/4$, approximately a 50 times greater change. Changing a single cluster id in a smaller cluster produces an overly large effect in WWCS1 as well.

WWCS2 is not vulnerable to the second problem noted above, but suffers from the first even more dramatically. Adding δ to one within-cluster distance results in a new WWCS2 of:

$$\mathrm{WWCS2}(C, X, d) = \sum_{c_i \in C} \frac{\sum_{x_j, x_k \in c_i} d(x_j, x_k)}{2|c_i|(|c_i| - 1)} + \frac{\delta}{2|c_*|(|c_*| - 1)}.$$

Thus, for WWCS2, when $|c_*| = 100$ the change is $\delta/19800$, but for $|c_*| = 2$ it is $\delta/4$, approximately a 5000 times greater change. The Dunn and Davies-Bouldin Index behave like WWCS1 and WWCS2, allowing changes in smaller clusters to have disproportional effects on their assessments of clusterings' quality.

Relative margin is arguably one of the least noise tolerant ICQMs possible, as it uses only one member of each cluster in its final quality assessment. Fig. 4.2 gives an example of this problem using a 2-clustering; by changing the cluster id of only one object in the dataset, I make the clustering look many times better for relative margin. ICQMs in this

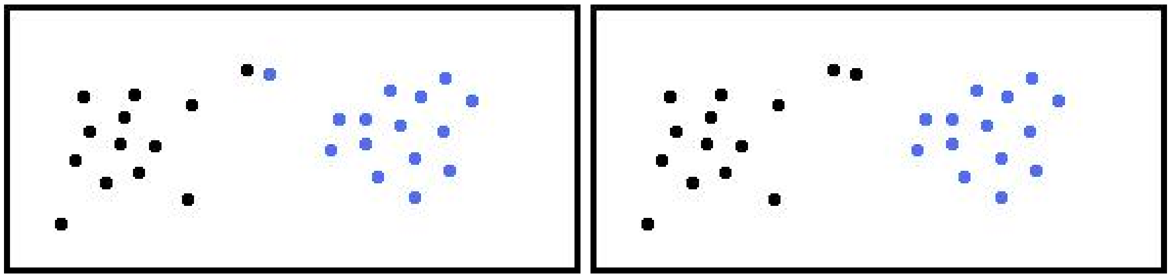


Figure 4.2: A single cluster membership change in a 2-clustering that causes a large change in relative margin's quality assessment of the clustering.

thesis that are not mentioned above can be considered more noise tolerant (WCS (Eq. 3.1), N-Cut (Eq. 3.9), Silhouette (Eq. 3.12), etc.).

Unlike ICQMs, most ECQMs are relatively stable with respect to small parameter changes. For example, changing a single label in C or T can alter Rand(C, T) (Eq. 3.34) by no more than $1/n$. Given that there are n objects in the dataset C and T are based on, and that the range of the Rand Index is 0 to 1, this seems like an ideal maximum reaction to such a change. Even ECQMs that allows certain cluster/class ids to have larger impact on their assessments have built in counterbalances that prevent high noise sensitivity. In general, noise tolerance is not much of an issue for ECQMs.

4.6 Number of Clusters in an Optimal Clustering

One of the most important and heavily examined tasks in clustering is that of selecting the right number of clusters for a dataset. A naive approach to this is to find the clustering on the dataset that scores the maximum by a chosen ICQM and state that the number of clusters contained in that clustering is right for the dataset. However, it is easy to show that this procedure is flawed. A great many ICQMs have biases with respect to the number of clusters. While an in-depth analysis of the by-number of clusters biases of various ICQMs is beyond the scope of this thesis, users should be aware that fairly complicated procedures [70, 55, 112, 125, 147, 164, etc.] are often used in combination with ICQMs in selecting the right number of clusters for a dataset. In general, a user should always be cautious when comparing clusterings with ICQMs when the clusterings contain a large spread in number of clusters; making the ICQM compensate for random chance (see Section 4.8) can mitigate this concern.

With respect to an ECQM of the form $M(C, T)$, all the ones in Chapter 3 return optimal scores when $C = T$, so we have $k = t$ for an optimal clustering, as we might expect. However, for many ECQMs the clustering returning the optimal score for some fixed T is not unique. Consider EQ (Eq. 3.33), the quality of a single cluster in EQ is:

$$\text{Entropy}(c_i, T) = \sum_{t_j \in T} \frac{p(c_i, t_j)}{p(c_i)} \log(\frac{p(c_i, t_j)}{p(c_i)}).$$

For some C and T on the same X, C is a *refinement* of T if:

$$\forall_{c_i \in C} \exists_{t_j} p(c_i, t_j) = p(c_i). \tag{4.1}$$

Informally, a refinement of T is any clustering that can be made by splitting the classes in T zero or more times (refining each class). We can see that when C is a refinement of T, we have $\forall_{c_i \in C}\text{Entropy}(c_i, T) = 0$. Substituting this in to Eq. 3.33 we have:

$$\begin{aligned} \text{EQ}(C, T) &= 1 - \sum_{c_i \in C} p(c_i) \frac{1}{-log(q)} 0 \\ &= 1. \end{aligned}$$

Combining this with the fact that a refinement must have t or more clusters, we can say that any clustering C with t or more clusters that is a refinement of T is an optimal scoring clustering with respect to T for EQ. This property seems like it might be useful for an ECQM in some situations, but problematic in others (notably, for an ECQM with this property any clustering C with all singleton clusters is always an optimal scoring clustering with respect to T). It is easy to show that PQ (Eq. 3.47) behaves exactly like EQ with respect to optimal scoring clusterings. My second NMI variant (Eq. 3.25) is even more relaxed than EQ, giving optimal scores not only when C is a refinement of T, but also when T is a refinement of C, allowing for potentially any number of clusters up to n to be optimal scoring. All the other ECQMs in Chapter 3 can be shown to have a single unique optimal clustering $C = T$ (which means $k = t$), this is arguably a more sensible/expected property, as we have told the ECQM that t is right number of clusters.

4.7 Scale Invariance

As with weak local consistency, Ackerman and Ben-David present *scale invariance* as an axiom for ICQMs [1]. Informally, a scale invariant ICQM is one where multiplying the distances between all the objects in any dataset by some constant value does not alter the ICQM's score for any clustering of that dataset. As with consistency, I assume the form of all ICQMs is $M(C, X, d)$ for the following discussion.

Definition 10 (Scale Invariance)**.** *ICQM M is* scale invariant *if, for any X, C, d, and $\lambda \in \mathbb{R}^+$, we have $M(C, X, d) = M(C, X, \lambda d)$.*

Intuitively, scale invariance seems like a reasonable property. However, given that some ICQMs that have been used effectively for many years are not scale invariant (for example, basic k-means using Lloyd's method, as described in Chapter 2, uses WWCS1, which is not scale invariant) it is possible that scale invariance is too restrictive.

There is nothing inherently wrong with clustering quality measures of C changing as we shrink/expand the X's object representations uniformly, rather the problem is that relative quality of clusterings may change. Uniform changes to datasets should never change ICQMs' *relative* quality assessments of clusterings on the datasets. Based on this, I define the property of relative scale invariance as follows.

Definition 11 (Relative Scale Invariance)**.** *ICQM M is* relative scale invariant *if, for any*

X, C, C', d, and $\lambda \in \mathbb{R}^+$ we have:

$$\frac{M(C, X, d)}{M(C', X, d)} = \frac{M(C, X, \lambda d)}{M(C', X, \lambda d)}. \tag{4.2}$$

Clearly, scale invariance implies relative-scale invariance, but we can see from Table 4.1 that relative-scale invariance includes a larger number of ICQMs that are in active use and widely accepted. Given this, relative-scale invariance might be a more suitable ICQM axiom.

4.8 Random Chance

Consider performing clustering using the following steps: 1) select a random number of clusters; and 2) randomly place each object into a cluster. Consider further that we have a set of true labels for the objects. We may reasonably expect that all clusterings generated by this process will score roughly the same for a fixed CQM. Unfortunately, what we find in practice is that the quality of random clusterings like this, for a given CQM, often vary based on the number of clusters and cluster sizes. This 'variable' amount of random chance hinders our ability to determine if a CQM has detected a meaningful clustering. It is therefore desirable to identify which clustering quality measures have variable random chance and, whenever possible, correct for it.

Let M be a clustering quality measure. I will first consider the situation of variable random chance when M is an ECQM. For this case, I begin by assuming that M has the form $M(C, T)$. There are three scenarios where we want to know if the M has variable amounts of random chance; the first situation is when both C and T may vary, the second situation is when only C may vary, and the final situation is when only T may vary. Each of these scenarios is important for a different kind of clustering evaluation experiment. Scenario one is important when we are looking for the best (C,T) pairing possible, where C and T are both treated identically (neither is a gold standard); scenario two is important when we are looking for the best clustering relative to a single gold standard; and finally scenario three is important when we are looking for the gold standard among many that most closely matches a single clustering. All three of these situations arise in clustering evaluation, so an ideal M will correct for variable random chance in all of them.

Let $E[M(C,T)]$ be the value of $M(C,T)$ that we expect by random chance for some unrelated C and T. A simple assumption may lead one to believe that $E[M(C,T)]$ is based on only C, or only T, or is always 0 for an ECQM, specifically that for unrelated C and T we have:

$$\forall_{c_i \in C} \forall_{t_j \in T} p(c_i, t_j) = p(c_i)p(t_j). \tag{4.3}$$

Eq. 4.3 can be used to erroneously derive what variable random chance exists for an ECQM. For example, consider determining $E[M(C,T)]$ for PQ (Eq. 3.47). Recall that the purity for a single cluster using PQ is defined as:

$$\text{Purity}(c_i, T) = \max_{t_j \in T} \frac{p(c_i, t_j)}{p(c_i)}.$$

Using Eq. 4.3 this becomes:

$$\begin{aligned} \text{Purity}(c_i, T) &= \max_{t_j \in T} \frac{p(c_i)p(t_j)}{p(c_i)} \\ &= \max_{t_j \in T} p(t_j). \end{aligned}$$

Substituting this in to PQ's final equation (Eq. 3.47) we get our measure of $E[\text{PQ}(C,T)]$:

$$\begin{aligned} E[\text{PQ}(C,T)] &= \sum_{c_i \in C} p(c_i) \max_{t_j \in T} p(t_j) \\ &= \max_{t_j \in T} p(t_j). \end{aligned}$$

From this we assume that PQ's variable random chance depends on only T. Similar reasoning may be used to derive $E[M(C,T)]$s that are 0 for a good number of ECQMs such as NMI variants.

Unfortunately, our assumption that $p(c_i, t_j) = p(c_i)p(t_j)$ is not correct as $p(c_i, t_j)$ is part of the joint distribution of two finite populations and requires a more complicated distribution to model properly. In such a situation, $p(c_i, t_j) \neq p(c_i)p(t_j)$ in general but it is dependent on both $p(c_i)$ and $p(t_j)$. Given this, and that all the ECQMs I consider use $p(c_i, t_j)$ values (as discussed Section 4.3), we can say that there is variable random chance in many ECQMs for all three of our scenarios.

As a side point, $p(c_i, t_j) = p(c_i)p(t_j)$ if C and T are on an infinite dataset but do not contain infinite clusters/classes themselves. It is worthwhile knowing what happens to variable random chance in this situation as there are many times when the size of a dataset is large enough, and C and T have few enough clusters/classes, that variable random chance will behave almost as if $p(c_i, t_j) = p(c_i)p(t_j)$. To that end, in Table 4.1 I present two columns for ECQMs, one for the situation where the ECQM ever has any variable random chance, and a second for what its variable random chance is based on asymptotically as a dataset approaches infinite size but C and T remain fixed in number of clusters/classes.

I now investigate Ms that are ICQMs. Assuming M has the form $M(C, X, d)$, M's variable random chance may depend C, X, and/or d. I make the assumption that all ICQMs have variable random chance with respect to X and d, and investigate only if variable random chance depends on C as well.

Let $E[M(C, X, d)]$ be the expected value of M for some X, C, d combination. It can be shown that all the ICQMs in Table 4.1 have variable random chance with respect to C. I give two examples, WCS and N-Cut.

Lemma 1 (WCS has variable random chance with respect to C)**.**
Proof. *It suffices to show that there are at least two clusterings of some X that have different $E[M(C, X, d)]$s for some d. For WCS (Eq. 3.1); let A be a set of all singleton clusters of any X and let B be a clustering of the same X with one cluster containing everything. Clearly, we have WCS$(A, X, d) = 0$ (as there are no within clusters distances to count) and WCS$(B, X, d) = \sum\limits_{x_i \in X} \sum\limits_{x_j \in X, x_i \neq x_j} d(x_i, x_j)$ (every distance is a within cluster distance). As these always hold, we may treat them as $E[WCS(C, X, d)]$s for clusterings in their situations (i.e. all singletons clusters and one cluster, respectively), but we have $WCS(A, X, d) \neq WCS(B, X, d)$ in general. This means $E[WCS(C, X, d)]$ is not fixed as C changes, so WCS has a variable amount of random chance depending on C.* □

Lemma 2 (N-Cut has variable random chance with respect to C)**.**
Proof. *Consider a size balanced random clustering with k clusters (n/k objects per cluster), generated on an X where $\forall_{x_i, x_j \in X, x_i \neq x_j} d(x_i, x_j) = \sqrt{\lambda}$ and $\sqrt{\lambda}$ is some constant in $\mathbb{R}^+$. N-Cut is defined as*

$$N\text{-}Cut(C, X, d) = \sum_{x_i, x_j \in x_i \nsim_C x_j} \frac{1}{d(x_i, x_j)^2} \sum_{c_i \in C} \frac{1}{\sum_{x_j, x_l \in c_i, x_j \neq x_l} \frac{1}{d(x_j, x_l)^2}}.$$

We can use the fact we know the number of clusters and the size of each cluster, combined *with λ, to derive $E[\textit{N-Cut}(C,X,d)]$:*

$$\begin{aligned}
E[\textit{N-Cut}(C,X,d)] &= \sum_{x_i,x_j \in x_i \nsim_C x_j} \frac{1}{\lambda} \sum_{c_i \in C} \frac{1}{\sum_{x_j,x_l \in c_i, x_j \neq x_l} \frac{1}{\lambda}} \\
&= (\frac{n(n-1)}{2} - k\frac{(n/k)(n/k-1)}{2})\frac{1}{\lambda}k\frac{2\lambda}{(n/k)(n/k-1)} \\
&= (n(n-1) - k(n/k)(n/k-1))k\frac{1}{(n/k)(n/k-1)} \\
&= \frac{nk(n-1)}{(n/k)(n/k-1)} - k^2 \\
&= k^2(\frac{n-1}{(n/k-1)} - 1).
\end{aligned}$$

This shows a dependence on k for $E[\textit{N-Cut}(C,X,d)]$. As k is not fixed for all clusterings, we can say that N-Cut has variable random chance with respect to C. □

It seems that variable random chance with respect to C is ignored in ICQMs in general; though it is worth noting that C-Index comes close to dealing with it. If we replace $S_{min}(X,l)$ with the average we would obtain from adding together $|x_i \sim_C x_j|$ random distances from X in C-Index (Eq. 3.15), then C-Index implements an ICQM's version of chance correction (Eq. 3.39) for WCS (Eq. 3.1). In general, we can correct any ICQM M of the form $M(C,X,d)$ for variable random chance with respect to C using the following adjustment:

$$\text{Chance-Correction}(M,C,X,d) = \frac{\text{M}(C,X,d) - E[\text{M}(C,X,d)]}{\max(\text{M}(C,X,d)) - E[\text{M}(C,X,d)]}. \tag{4.4}$$

To the best of my knowledge, this correction is not in active use for any ICQM that I know of even though many ICQMs could benefit greatly from its application; I will use it in my new ICQM though. As all the ICQMs I have considered have variable random chance with respect to C, both in general and asymptotically, I omit the variable random chance columns for the ICQMs in Table 4.1.

Overall, we can say the clustering quality measures in widespread use do not usually handle variable random chance well, even though they need to do so to be the most effective they can be.

4.9 Richness

Richness is another property for ICQMs discussed by Ackerman and Ben-David [1].

Definition 12 (Richness)**.** *ICQM M is rich if, for any non-trivial C and X, we are always able to pick a d such that $M(C, X, d) = \max_{C' \in C^*} M(C', X, d)$, where C^* is the set of all clustering possible on X.*

The concept here is that we can make any clustering look like the best clustering possible through careful selection of our distance function. We can argue that this is a necessary property for every ICQM if we agree that distances are the only things that make a clustering good or bad (much like we had to agree on using distances to define improvement in consistency). Unfortunately, it can be seen that many ICQMs have implicit value assigned other things, such as the number of clusters, thus the authors later suggested *co-final richness* [1] as an axiom.

4.10 Time Complexity

Clustering algorithms often have substantial runtime; as I have noted it is not uncommon for such algorithms to require greater than $O(n^2)$ runtime, and the computation of a full similarity/distance matrix which many of them require takes $O(n^2m)$ runtime. Such runtimes are prohibitively expensive given the massive datasets on which clustering algorithms are often applied, it is therefore important that clustering quality measures do not exacerbate the problem and have as little runtime as possible while still being effective.

Table 4.1 gives the runtimes of most of the clustering quality measures in Chapter 3. For the ICQMs it is assumed a distance matrix has been pre-computed for them. We can see that the runtimes of most of the ICQMs are reasonable, but there are some that are notably high: C-Index; Gamma; and relative margin. When $k \leq \sqrt{n}$ and $t \leq \sqrt{n}$, every ECQM has equal or superior runtime to the fastest ICQM. The slowest ECQMs under these assumptions are the pair-wise comparison measures (the Jaccard, Rand, and Adjusted Rand Index) at $O(n^2)$.

Table 4.1: Properties of some of the clustering quality measures discussed in Chapter 3.

ICQM	Eq.	Improving Consistency		Scale Invariant	Relative Scale Invariant	Runtime
		Between	Within			
aWCS	3.4	no	yes	no	yes	$O(n^2)$
B/W	3.8	yes	yes	yes	yes	$O(n^2)$
C-Index	3.15	no	yes	yes	yes	$O(n^2 log(n))$
DB Index	3.17	no	no	yes	yes	$\max(O(nm), O(k^2m))$
Dunn Index	3.13	no	no	yes	yes	$\max(O(nm), O(k^2m))$
Gamma	3.10	no	no	yes	yes	$O(n^2 log(n))$
N-Cut	3.9	yes	yes	yes	yes	$O(n^2)$
Ratio-Cut	3.7	yes	no	no	yes	$O(n^2)$
Rel. Margin	3.18	no	no	yes	yes	$O(n^k)$
Silhouette	3.12	yes	yes	yes	yes	$O(n^2)$
WCS	3.1	no	yes	no	yes	$O(n^2)$
WWCS1	3.2	no	yes	no	yes	$O(n^2)$
WWCS2	3.3	no	yes	no	yes	$O(n^2)$

ECQM	Eq.	Fullness	Possible Optimal Scoring # of Clusters Relative to # of Classes	Variable Random Chance	Asymptotic Variable Random Chance	Runtime
Accuracy	3.40	no	NA	yes	with C, T	$\max(O(n), O(k^3))$
ARI	3.36	-	$k = t$	no	none	$O(n^2)$
AMI	3.28	-	$k = t$	no	none	-
EQ	3.33	yes	$k \geq t$	yes	with C,T	$\max(O(n), O(tk))$
FQ	3.45	no	$k = t$	yes	with C,T	$\max(O(n), O(tk))$
Hamming	3.41	no	NA	yes	with C,T	$\max(O(n), O(k^3))$
Jaccard	3.35	yes	$k = t$	yes	with C,T	$O(n^2)$
NMI 1	3.22	yes	$k = t$	yes	none	$\max(O(n), O(tk))$
NMI 2	3.24	yes	$1 \leq k \leq n$	yes	none	$\max(O(n), O(tk))$
NMI 3	3.25	yes	$k = t$	yes	none	$\max(O(n), O(tk))$
PQ	3.47	no	$k \geq t$	yes	with T	$\max(O(n), O(tk))$
Rand	3.34	yes	$k = t$	yes	with C,T	$O(n^2)$
VI	3.27	yes	$k = t$	yes	with C,T	$\max(O(n), O(tk))$

Chapter 5

Improving Document Clustering using Okapi BM25 Feature Weighting

Anil K. Jain, the co-author of a prominent clustering review paper cited in the first paragraph of this thesis [88], and a book on the subject of clustering [87], recently pointed out [86] that it is more than 50 years after the design of the first k-means type algorithm for clustering, but that algorithm is still in widespread use today. One may ask, given all the supposedly superior clustering algorithms that exist now, why is that one still so popular? I argue that a large part of the reason for this, and most of the issues with clustering today, is due to problems with evaluating clusterings.

In the two previous chapters I looked at clustering quality measures (CQMs) and many of their important properties. I showed some of the wide array of clustering quality measures available, as well as several conceptually complex properties to consider, some of which are not necessary properties, but merely interesting ways to differentiate clustering quality measures. These chapters lead to fundamental questions.

What does it mean if one clustering is better than another using a particular CQM? I have already discussed how we cannot use such a result to indicate universal superiority of clusterings, but one may not even know what it means within a specific context, particularly as we understand most CQMs and their relationships poorly.

Unfortunately, most of clustering literature does not help us with practical questions like those above. While I believe that there are many interesting and high-quality theoretical papers in the literature [1, 2, 4, 109, 151, etc.], such papers tend not to help with practical issues/questions like these. This is not a failing of such papers, but rather it is the result of needing to remove the particulars of practical clustering in order to make any broadly applicable theoretical headway. One might expect experimental/algorithm design papers to help more with practical clustering, but they are often not helpful as well due to very weak experimental components. For example, it is extremely common for authors of papers presenting new clustering algorithms to spend the majority of the paper developing the algorithm, then try to show that the algorithm works in practice using a few datasets, a few competitors (often just three or less), and a few ECQMs (often just one). This is wholly insufficient to indicate that a specific clustering algorithm is of high quality. Further, it is very uncommon for any clustering paper at all to investigate specific domain trends in an effort to justify why certain clustering algorithms and/or quality measures are appropriate for them. As a comparison, consider classifiers. Joachims [90] gives extensive reasons for using SVM classifiers on text through investigating properties of actual text datasets. This kind of investigation seems to be missing almost entirely from the mainstream clustering literature.

In view of the above, in this chapter I present a document clustering experiment that is extremely broad. I believe that, beyond its explicit stated purposes in the following section, the experiment serves to illustrate the level of detail that is necessary for practical use to be made of the results.

5.1 Preliminaries

Certain assumptions regarding the weighting of text features are nearly ubiquitous in the text clustering literature. I explore some of these assumptions here, investigating the effect of typical text feature weighting on document clustering. The aim of the experiments in this chapter is to determine if standard term weighting strategies for document clustering can be improved upon, as well as illuminating some issues with clustering evaluation.

I reiterate some notation for clarity. X is a dataset we want to cluster, here always one of documents, x_i is the ith document in X, represented using the standard vector space model:

$$x_i = (x_{i1}, x_{i2}, \ldots, x_{im}). \tag{5.1}$$

x_{ij} is the weight of feature j in document i. *tf* weighted vectors for documents have the form

$$x_{ij} = \mathit{tf}_{ij}, \tag{5.2}$$

where tf_{ij} is the *term frequency* of j in i. Binary weighted vectors for documents have the form

$$x_{ij} = \begin{cases} 1, & \text{if } tf_{ij} > 0; \\ 0, & \text{otherwise.} \end{cases} \tag{5.3}$$

tf-idf weighted vectors for documents have the form

$$x_{ij} = tf_{ij} \log(\frac{n}{n_j}), \tag{5.4}$$

where n is the number of documents in X and n_j is the number of documents in X that contain term j. Euclidean-length normalization of a vector is done by applying the following transformation:

$$x^{'}_{ij} = \frac{x_{ij}}{\sqrt{\sum\limits_{j=1}^{m} x_{ij}^2}}. \tag{5.5}$$

Most document clustering literature discusses using (or uses) the *tf-idf* weighting in Eq. 5.4, mostly with the length normalization in Eq. 5.5 [8, 140, 145, 162, 80, 172, 173, etc.].

There is some research from fields related to clustering, such as classification, that indicate that *idf* is an important part of feature weighting for documents, while *tf* is not as useful [157]; such results suggest that the same might be true of document clustering. Despite this research, I will show that *tf* weighting is superior to binary weighting, and also that the inclusion of an *idf* component to *tf* is not necessarily beneficial in document clustering. While I will show *idf* generally has a small positive effect on clustering results, for some datasets it is substantially harmful to a wide range of clustering algorithms. Further, the effect of *idf* is heavily dependent on the clustering algorithm.

BM25 [129] is a weighting function that has been shown to be effective in many fields outside of clustering, but has only recently been seriously considered in document clustering, with works that do use BM25 still being a small minority [15, 45, 101, 153, 156]. An examination of these works suggests that document clustering using BM25 feature weighting is promising, but it also reveals several areas where research is lacking. First, no broad investigation of the suitability of BM25 feature weighting for document clustering has been done; the rational for its use, up to this date, has simply been that it has worked well in other applications. Second, suitable parameter values for BM25 when document clustering have not been investigated; researchers have simply adopted default values for them. Finally, no work has assessed the merits of using just the BM25 term saturation component as a feature weight. I investigate all three of these my experiments.

I show that replacing the *tf* in *tf-idf* weighting (Eq. 5.4) with the BM25 term saturation component, and changing nothing else about how the clustering is performed, produces results superior to *tf-idf* weighting in an extensive test. I also investigate the use of just the BM25 term saturation component as a feature weight, which I show outperforms *tf*. Parameter estimation for k_1 in BM25 is also investigated, with my research leading to the conclusion that typical values for k_1 from other tasks such as ad-hoc retrieval are unsuitable; k_1 should be higher to achieve better clustering results.

With respect to clustering evaluation in general, I will show that certain clustering algorithms are biased towards certain clustering quality measures, and, more distressingly, that clustering quality measures have large amounts of disagreement in how they rank clusterings/clustering algorithms.

The rest of this chapter proceeds as follows. Section 5.2 describes my *tf*, *tf-idf*, and binary weighting experiments and the datasets, clustering algorithms, and clustering quality measures used in them. Section 5.3 discusses the results of these experiments, highlighting some key discoveries and analyzing why they occurred. Section 5.4 demonstrates that BM25-weighted document representations produce superior clusterings when compared to their non-BM25 counterparts. Section 5.5 gives a summary.

5.2 Experimental Setup

In this section I describe my *tf*, *tf-idf*, and binary weighting experiments and the datasets, clustering algorithms, and clustering quality measures used in them.

I selected 17 clustering algorithms and eight document datasets, which are described in Section 5.2.2 and Section 6.3.1 respectively. For each dataset I generated three representations: one using *tf*, another using *tf-idf*, and a final one using binary weighting. The definitions used for the weighting functions, while creating the representations, were exactly as detailed at the start of this chapter. All three weightings were length normalized as per Eq. 5.5. Each clustering algorithm was run on each representation with two to 30 clusters. This gave a total of 11832 ($8 * 3 * 17 * 29$) clusterings. These clusterings were evaluated using the four clustering quality measures described in Section 5.2.3. The results of the clustering quality measures on the clusterings are used in Section 5.3 to analyze the effect of the weightings. The following subsections detail the specific datasets, clustering algorithms, and clustering quality measures I used.

5.2.1 Datasets

I used a total of eight datasets in my experiment, all of which are available at the Karypis Lab website[1] in the form of preprocessed term frequency count vectors. The vectors for each dataset can be created from the base documents of each dataset using a simple script called doc2mat[2]. Conceptually, doc2mat performs the following operations to generate the vectors for the datasets: 1) all non-alphanumeric characters are converted to whitespace; 2)

[1]`http://glaros.dtc.umn.edu/gkhome/views/cluto/download`
[2]`http://glaros.dtc.umn.edu/gkhome/files/fs/sw/cluto/doc2mat.html`

Table 5.1: The datasets used in my experiments.

Dataset	**# of Doc**	**# of Terms**	**# of Classes**
fbis	2463	2000	17
new3	9556	36306	44
tr31	927	10128	7
tr41	878	7454	10
tr45	690	8261	10
re0	1504	2886	13
re1	1657	3758	25
wap	1560	8460	20

documents are tokenized using whitespace as a separator; 3) a simple stopword list (built in to the program) filters out all stopwords; 4) a Porter stemmer is applied to the tokens, 5) tokens containing any non-alphabetic characters are discarded and all other tokens are case-normalized (lower case); 6) the remaining tokens are used to generate the terms for the dataset; and 7) the final term count vectors are created using the results of steps (5) and (6). I did not apply doc2mat myself; instead I simply used the preprocessed vectors provided by the site owners (it should be further noted that the default parameter settings of doc2mat do not match those discussed here, see the doc2mat documentation for details on which parameter settings were used to generate the dataset vectors).

The eight datasets (and sometimes larger versions of them) have been used in numerous publications [16, 59, 145, 162, 172, etc.] and may thus be considered as standard test sets for document clustering. Table 5.1 summarizes their characteristics.

The document collections *new3*, *tr51*, *tr41*, and *tr31* are derived from collections used at TREC (Text REtrieval Conference[3]). The *fbis* collection is from the Foreign Broadcast Information Service dataset in TREC-5. The first 2000 of the fbis documents were used in my tests. This allowed us to use a standard Java matrix package (*JAMA*[4]) which required that the number of dimensions be greater than or equal to the number of objects when

[3]`http://trec.nist.gov`
[4]`http://math.nist.gov/javanumerics/jama/`

applying singular value decomposition (fbis has 2000 dimensions). *Re0* and *re1* are from the Reuters-21578 text categorization test collection distribution 1.0[5]. The *wap* collection is the from the WebACE project [24]. Each document of the *wap* dataset was a single web page in the Yahoo! directory.

5.2.2 Clustering Algorithms

Of the 17 clustering algorithms used in my experiments, I implemented all but the six based on Zhao and Karypis [171, 172, 173] myself. Where possible, my implementations were validated through comparisons to previously published results. The algorithms using the objective functions from Zhao and Karypis were performed using the authors' own clustering toolkit[6].

Selection of the clustering algorithms was based on the following criterion: 1) Were they well-established algorithms? 2) Did they take a pre-specified number of clusters as a parameter and produce hard clusters? 3) Together, did the set of algorithms cover a breadth of the well-established techniques for document clustering? 4) Together, did the set of algorithms include those algorithms reported to produce good results in previous research? This last requirement was especially important for a meaningful analysis of the effects of *tf-idf* and other term weightings. Table 5.2 lists the 17 clustering algorithms I used.

UPGMA, Slink, Clink, and PAM operate exactly as detailed in Section 2.4. General forms of most of the other algorithms in Table 5.2 have been discussed in Section 2.4, below I give specific details for my implementation/uses of them.

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

[6]`http://glaros.dtc.umn.edu/gkhome/views/cluto/download`

Table 5.2: The clustering algorithms used in my document clustering experiments.

Algorithm	**Short**	**Reference**
K-means	K-means	[106]
Partition Around Medoids	PAM	[91]
Repeated Bisecting k-means	RB-K-means	[145]
Unnormalized Spectral	Spect-Un	[152]
Random Walk Spectral	Spect-RW	[139]
Symmetric Spectral	Spect-Sy	[119]
Principle Component Analysis+k-means	PCA-K	[124]
Non-Negative Matrix Factorization	NC-NMF	[162]
Unweighted Pair Group Method	UPGMA	[91]
Single Linkage	Slink	[88]
Complete Linkage	Clink	[88]
Repeated Bisecting I2	RB-I2	[171, 172, 173]
Repeated Bisecting H1	RB-H1	[171, 172, 173]
Direct I2	Direct-I2	[171, 172, 173]
Direct H1	Direct-H1	[171, 172, 173]
Agglomerative I2	Agglo-I2	[171, 172, 173]
Agglomerative H1	Agglo-H1	[171, 172, 173]

My *K-means* algorithm used Lloyd's method [106] with the initial centroids being selected randomly from the vectors of the dataset. I ran the algorithm 20 times for each value of k and kept only the best result according to the *k-means* internal objective function. *RB-K-means* repeatedly splits the dataset using *K-means*. Binary splitting was used, with the largest remaining cluster split at each iteration [145].

I selected three varieties of spectral clustering. For each of these three varieties the weighted adjacency matrix W was generated through an r-nearest neighbor scheme with cosine similarity using $r = 20$. *Spect-Un* clustered on the k eigenvectors of the Laplacian $L = D - W$, where D was the degree matrix of W. For details on the Laplacian for *Spect-RW* see [139], and for *Spect-Sy* see [119]. The clustering of the eigenvectors for all three methods was done using my *K-means* algorithm.

For my PCA algorithm, I first applied PCA to produce an $n \times 20$ reduced document feature space. This was followed by the application of my *K-means* algorithm. I used the *NC-NMF* version *NMF* from [162] as the authors showed it to be more effective than simple *NMF*.

Finally, I selected two of the objective functions from Zhao and Karypis [171, 172, 173]: *I2* and *H1*. For each of these, I used three distinct optimization methods: repeated bisection, direct (partitional), and agglomerative. This gave me a total of six algorithms. For details on their exact implementations, readers can consult Zhao and Karypis [171, 172, 173], as I used the authors' own clustering toolkit to perform these algorithms, along with the exact parameters specified in those papers. The *I2* function is essentially the *K-means* objective function except any similarity metric may be used in the calculation. The *H1* function is *I1*/*E1*, where *E1* is an objective function based around minimizing the weighted similarity of cluster centroids from the centroid of the whole dataset, and *I1* is an objective function similar to UPGMA.

5.2.3 Clustering Quality Measures

I used four ECQMs in my experiments; normalized mutual information (NMI); *F-measure (FQ)*; *purity (PQ)*; and *entropy (EQ)*. PQ and EQ are exactly as detailed in Chapter 3, for NMI I used the first version I detailed in Chapter 3 (Eq. 3.22), and FQ is Eq. 3.42 with $\beta = 1$ (F1). These four measures were selected because they are very common in document clustering literature.

5.3 Effects of Document Feature Weightings

Rather than compare *tf*, *tf-idf*, and binary weightings simultaneously I chose to examine two questions I believe are key with respect to document feature weighting; 1) Is the idf component of tf-idf weighting needed for document clustering?; and 2) Is term frequency more useful than simple term presence/absence for document clustering? Question (1) is

Table 5.3: The percentage change in clustering quality measures when using *tf-idf* document representations over *tf*, by dataset and overall.

Dataset	NMI	FQ	PQ	EQ
fbis	-1.6%	-0.4%	-1.0%	-1.3%
new3	0.1%	-2.3%	-2.0%	0.3
re0	-11.0%	-4.1%	-5.3%	-5.4%
re1	14.8%	4.9%	4.9%	8.6%
tr31	15.9%	8.2%	7.5%	10.4%
tr41	12.5%	7.8%	8.3%	8.8%
tr45	20.0%	17.5%	12.5%	15.6%
wap	6.6%	7.6%	6.4%	7.2%
Overall	7.2%	4.9%	3.9%	5.5%

evaluated in Section 5.3.1 by comparing my*tf* and *tf-idf* results. I will show the benefit of *idf* is heavily dependent both on the dataset and the clustering algorithm. I examine question (2) in Section 5.3.2 by comparing my *tf* and binary results. I will show that *tf* is substantially superior to binary weighting.

5.3.1 Effect of *tf-idf* on Document Clustering

To determine if *tf-idf* was having a positive effect when compared to *tf*, I first took the clustering quality measures on the 7888 tuples of (weighting, dataset, algorithm, #clusters) for the *tf* and *tf-idf* weightings and collapsed them by averaging each clustering quality measure over the number of clusters. This gave 272 tuples of (weighting, dataset, algorithm) with averaged clustering quality measures. From these tuples I derived three tables comparing *tf* and *tf-idf* weighting; 1) By dataset and average over all clustering algorithms for that dataset (Table 5.3); 2) By dataset and the best clustering algorithm for that dataset (Table 5.4); and 3) By algorithm (Table 5.5).

The overall row in Table 5.3 indicates that, on average, *tf-idf* offers improved results over *tf*. However, *tf-idf* is actually substantially worse than *tf* for re0 and somewhat worse for fbis and new3. Table 5.4 shows the best clustering algorithm for each dataset and

Table 5.4: The best algorithm for *tf* and *tf-idf* weighting on each dataset by each measure. Diff is the improvement in using the best *tf-idf* over the best *tf* algorithm.

NMI					
	tf		**tf-idf**		**Diff**
fbis	RB-H1	0.566	RB-H1	0.558	-1.9%
new3	RB-H1	0.577	RB-I2	0.590	2.3%
re0	RB-I2	0.420	RB-H1	0.417	-0.7%
re1	RB-I2	0.492	RB-I2	0.556	12.9%
tr31	RB-H1	0.533	Agglo-I2	0.591	10.8%
tr41	Agglo-I2	0.630	Agglo-I2	0.657	4.4%
tr45	RB-I2	0.622	Agglo-I2	0.667	7.2%
wap	RB-I2	0.571	Agglo-H1	0.573	0.3%
Overall					4.4%

FQ					
	tf		**tf-idf**		**Diff**
fbis	Agglo-H1	0.560	Agglo-H1	0.549	-2.0%
new3	RB-H1	0.324	RB-I2	0.323	-0.3%
re0	Direct-H1	0.478	Direct-I2	0.431	-9.8%
re1	Agglo-I2	0.470	Agglo-I2	0.479	1.8%
tr31	Clink	0.585	UPGMA	0.688	17.6%
tr41	Agglo-I2	0.611	Direct-I2	0.655	7.2%
tr45	Agglo-I2	0.590	Agglo-H1	0.672	14.0%
wap	Agglo-I2	0.507	Agglo-H1	0.539	6.4%
Overall					3.1%

EQ					
	tf		**tf-idf**		**Diff**
fbis	RB-H1	0.704	RB-H1	0.692	-1.8%
new3	RB-H1	0.601	RB-H1	0.594	-1.1%
re0	RB-H1	0.703	RB-H1	0.689	-2.1%
re1	RB-I2	0.632	RB-I2	0.663	5.0%
tr31	RB-H1	0.859	RB-I2	0.895	4.3%
tr41	RB-I2	0.860	RB-I2	0.884	2.8%
tr45	RB-I2	0.831	RB-H1	0.854	2.8%
wap	RB-I2	0.686	RB-H1	0.689	0.5%
Overall					1.3%

PQ					
	tf		**tf-idf**		**Diff**
fbis	RB-H1	0.687	RB-H1	0.676	-1.6%
new3	RB-H1	0.666	RB-I2	0.679	1.9%
re0	RB-H1	0.681	RB-H1	0.679	-0.2%
re1	RB-I2	0.626	RB-I2	0.685	9.4%
tr31	RB-H1	0.810	RB-I2	0.850	4.9%
tr41	RB-I2	0.829	RB-I2	0.859	3.7%
tr45	RB-I2	0.783	RB-H1	0.816	4.3%
wap	RB-H1	0.670	RB-H1	0.677	1.1%
Overall					2.9%

Table 5.5: Improvements by clustering algorithms when using *tf-idf* over *tf* weighting.

Main Type	**Algorithm**	**NMI**	**FQ**	**PQ**	**EQ**
Hierarchical	UPGMA	22.1%	16.4%	15.8%	22.9%
	RB-K-means	14.0%	10.5%	7.7%	9.7%
	Agglo-H1	5.6%	4.3%	2.9%	4.4%
	Agglo-I2	5.4%	3.1%	2.8%	4.5%
	RB-H1	3.1%	2.1%	1.1%	2.6%
	RB-I2	2.8%	1.2%	1.1%	2.6%
	Slink	6.6%	0.0%	0.2%	0.8%
	Clink	-5.4%	-1.2%	-0.1%	0.0%
	Overall	6.8%	4.6%	3.9%	5.9%
Partitional	NMF-NC	19.4%	13.5%	12.0%	13.2%
	Direct-I2	14.7%	7.4%	7.7%	10.5%
	PAM	10.4%	8.3%	7.0%	8.0%
	Direct-H1	11.9%	6.0%	6.1%	8.0%
	Spect-Un	6.5%	5.4%	4.3%	5.8%
	Spect-RW	5.3%	5.7%	3.3%	5.0%
	K-means	7.6%	4.4%	2.5%	4.1%
	Spect-Sy	4.1%	3.7%	2.4%	3.8%
	PCA-K	2.1%	1.2%	4.9%	-1.2%
	Overall	9.1%	6.2%	3.9%	6.4%

weighting, by each of the four clustering quality measures. One can see that Table 5.4 is mostly consistent with Table 5.3 in terms of when *tf-idf* or *tf* is better, with the best *tf* and *tf-idf* results being closer, in general, than average *tf* and *tf-idf* results.

A possible reason for *idf*'s harmful effect on some datasets is apparent from a nearest neighbor analysis. For each dataset and weighting I calculated the percentage of r-nearest neighbors, per document, that share the same label as that document. Figure 5.1 presents the results of this analysis for $r = 1$ to 30. Considering just the *tf* and *tf-idf* lines for the moment, we see definite trends. For re0, where *idf* is harmful, we see *tf-idf* yielding a consistently worse nearest neighborhood than *tf*. For the datasets where *idf* is beneficial (re1, tr31, tr41, tr45, and wap), we see *tf* yielding better small neighborhoods, but as r increases *tf-idf* reduces less than *tf*, yielding substantially better neighborhoods than *tf* at higher rs. For new3 and fbis, where *idf* is only somewhat harmful, we see that *tf* again begins with better neighborhoods, but as r increases *tf* and *tf-idf* approach the same quality of neighborhood (as opposed to *tf-idf* becoming better). As clustering algorithms tend to focus on placing nearest neighbors in similar clusters, this provides a reasonable explanation for my different by-dataset results.

The average improvement by algorithms presented in Table 5.5 are split into hierarchical and partitional groups. Note that RB-K-means and other repeated bisection methods are placed in the hierarchical section, as they generate hierarchies of clusters, even though the splitting decision at each level is based around partitioning. It is immediately noticeable from Table 5.5 that the benefit of *idf* is not divisible along partitional versus hierarchical lines. For example, UPGMA and NC-NMF gain the largest benefit from using *tf-idf*, the former being hierarchical and the latter partitional. Another notable aspect is that the better clustering algorithms (from Table 5.4, 5.6 and 5.7) benefit less from *tf-idf* than most

Table 5.6: Algorithmic Rankings by my clustering quality measures when using *tf*.

tf			
NMI	**FQ**	**PQ**	**EQ**
RB-I2	Agglo-I2	RB-I2	RB-H1
RB-H1	Agglo-H1	RB-H1	RB-I2
Agglo-I2	K-means	Agglo-H1	Agglo-I2
Agglo-H1	Direct-H1	Agglo-I2	Agglo-H1
Spect-Sy	Direct-I2	Spect-Sy	Spect-Sy
K-means	Spect-Sy	K-means	K-means
PCA-K	Spect-Un	PCA-K	RB-K-means
Spect-Un	RB-I2	RB-K-means	PCA-K
Spect-RW	RB-H1	Spect-RW	Spect-Un
RB-K-means	UPGMA	Spect-Un	Spect-RW
Direct-I2	NC-NMF	PAM	NC-NMF
Direct-H1	Clink	NC-NMF	PAM
NC-NMF	PCA-K	Clink	Clink
Clink	Spect-RW	Direct-H1	Direct-H1
PAM	RB-K-means	Direct-I2	Direct-I2
UPGMA	PAM	UPGMA	UPGMA
Slink	Slink	Slink	Slink

Table 5.7: Algorithmic rankings by my clustering quality measures when using *tf-idf*.

tf-idf			
NMI	**FQ**	**PQ**	**EQ**
Agglo-H1	Agglo-I2	RB-H1	RB-I2
RB-I2	NC-NMF	RB-I2	RB-H1
Agglo-I2	Agglo-H1	Agglo-H1	Agglo-I2
RB-H1	UPGMA	Agglo-I2	Agglo-H1
NC-NMF	Direct-I2	RB-K-means	RB-K-means
Spect-Sy	Direct-H1	NC-NMF	Spect-Sy
RB-K-means	K-means	Spect-Sy	NC-NMF
Spect-Un	Spect-Un	K-means	K-means
K-means	RB-K-means	Spect-Un	Spect-Un
Spect-RW	Spect-RW	Spect-RW	Spect-RW
Direct-I2	Spect-Sy	PAM	PCA-K
UPGMA	RB-H1	PCA-K	PAM
Direct-H1	Clink	UPGMA	Clink
PCA-K	RB-I2	Clink	UPGMA
PAM	PAM	Direct-I2	Direct-I2
Clink	PCA-K	Direct-H1	Direct-H1
Slink	Slink	Slink	Slink

Table 5.8: Kendall's τ correlation between all the rankings in Table 5.6 and Table 5.7.

		tf				tf-idf			
		NMI	**FQ**	**PQ**	**EQ**	**NMI**	**FQ**	**PQ**	**EQ**
tf	**NMI**	1.000	0.426	0.809	0.824	0.618	0.044	0.632	0.706
	FQ	0.426	1.000	0.265	0.279	0.397	0.500	0.176	0.221
	PQ	0.809	0.265	1.000	0.926	0.603	-0.059	0.735	0.779
	EQ	0.824	0.279	0.926	1.000	0.618	0.015	0.779	0.824
tf-idf	**NMI**	0.618	0.397	0.603	0.618	1.000	0.279	0.750	0.735
	FQ	0.044	0.500	-0.059	0.015	0.279	1.000	0.176	0.132
	PQ	0.632	0.176	0.735	0.779	0.750	0.176	1.000	0.926
	EQ	0.706	0.221	0.779	0.824	0.735	0.132	0.926	1.000

of the other algorithms (except Slink and Clink).

It is highly noteworthy that my investigation of *tf* versus *tf-idf* weighting revealed that certain algorithms appear to favor certain clustering quality measures. To show this, I used the dataset collapsed by number of clusters again. For each weighting, dataset, and clustering quality measure, the clustering algorithms were ranked by their clustering quality measure, from one (best) to 17 (worst). I then computed each algorithm's average rank by weighting and clustering quality measure. Table 5.6 shows the algorithms, ordered by this average ranking (from best to worst) when *tf* is used, for each clustering quality measure. Table 5.7 shows similar results for *tf-idf* weighting. In general, the less than perfect level of agreement visible in those tables illustrates the problematic nature of clustering evaluation—if measures in common use don't agree on what is better, how can we improve and/or use clustering.

The most striking inconsistency in Table 5.6 and Table 5.7 is the behavior of FQ. We notice that RB-H1 and RB-I2, which are overall the best algorithms, rank much lower by FQ for both *tf* and *tf-idf* weighting. Also, for both *tf* and *tf-idf*, UPGMA fairs much better with FQ than with other measures, as do Direct-H1 and Direct-I2. A Kendall's τ test for correlation between pairs of the eight rankings in Table 5.6 and 5.7 is presented in Table 5.8. One may note some measure of agreement between *tf* NMI, *tf* PQ, *tf* EQ, *tf-idf* NMI, *tf-idf* PQ, and *tf-idf* EQ rankings, with the minimum τ between any pair of those being 0.618. Perhaps unsurprisingly, the *tf* FQ and *tf-idf* FQ rankings have a τ of 0.5, with much lower (even negative in one case) τ values with the other six rankings.

A potential source for FQ's large disagreement is related to the fullness property I discussed on page 76. Of the four ECQMs I used, FQ is the only one that allows entire clusters to be ignored in its quality assessment.

With respect to which algorithms are better, several algorithms have generally high ranks in Table 5.6 and Table 5.7 across all the clustering quality measures, including RB-H1, RB-h2, Agglo-H1, Agglo-I2, and Spect-Sy. RB-K-means and NC-NMF perform well with *tf-idf* clustering quality measures only. Interestingly, K-means performs reasonably well by all measures. On the other hand, we can see that Slink and Clink provide uniformly poor performance.

Summarizing this section, I note that *tf-idf* offers an improvement over *tf*, but it is not always better. Its benefit fluctuates heavily with the dataset, clustering algorithms, and evaluation measures used.

5.3.2 Effect of *tf* and binary weighting on Document Clustering

To determine if term frequency was more beneficial than binary term weights I compared the *tf* results to the binary results. The procedure for performing this experiment was the same as in the previous subsection, except my *tf-idf* results were replaced with my binary results. Table 5.9 shows the difference in the best algorithm results of binary and *tf* weightings.

It is clear from Table 5.9 that binary weighting is notably worse than standard *tf* weighting. In a few cases the best binary results are better than the *tf* results, but they are often dramatically worse. When examining the average behavior of binary weighting, both by dataset and by clustering algorithm, I likewise found it to be notably worse than *tf*. A simple explanation for binary weighting's poor performance can be found by examining Figure 5.1. One can see that in every case except re0, the nearest neighborhoods of binary weightings are greatly inferior to those of *tf*. For re0, binary weighting produces

Table 5.9: The best algorithm for *tf* and binary weighting on each dataset by each measure. Diff is the improvement in using the best binary algorithm over the best *tf* algorithm.

NMI					
	tf		**binary**		**Diff**
fbis	RB-H1	0.569	RB-H1	0.498	-12.4%
new3	RB-H1	0.577	RB-H1	0.527	-8.6%
re0	RB-I2	0.420	RB-I2	0.437	4.1%
re1	RB-I2	0.493	RB-H1	0.431	-12.4%
tr31	RB-H1	0.533	RB-I2	0.505	-5.3%
tr41	Agglo-I2	0.630	RB-H1	0.603	-4.2%
tr45	RB-I2	0.622	RB-H1	0.567	-8.8%
wap	RB-I2	0.571	Agglo-I2	0.598	4.6%
Overall					-5.4%

FQ					
	tf		**binary**		**Diff**
fbis	Agglo-H1	0.560	Clink	0.476	-15.0%
new3	RB-H1	0.324	RB-H1	0.271	-16.4%
re0	Direct-H1	0.478	Direct-H1	0.455	-4.7%
re1	Agglo-I2	0.470	Direct-H1	0.379	-19.5%
tr31	Clink	0.585	UPGMA	0.558	-4.6%
tr41	Agglo-I2	0.611	Agglo-H1	0.550	-10.0%
tr45	Agglo-I2	0.590	PCA-K	0.537	-9.0%
wap	Agglo-I2	0.507	Agglo-I2	0.589	16.1%
Overall					-7.9%

EQ					
	tf		**binary**		**Diff**
fbis	RB-H1	0.704	RB-H1	0.641	-9.0%
new3	RB-H1	0.601	RB-H1	0.554	-7.9%
re0	RB-H1	0.703	RB-I2	0.716	1.7%
re1	RB-I2	0.632	RB-H1	0.560	-11.4%
tr31	RB-H1	0.859	RB-I2	0.833	-3.0%
tr41	RB-I2	0.860	RB-H1	0.843	-2.0%
tr45	RB-I2	0.831	RB-H1	0.765	-7.9%
wap	RB-I2	0.686	RB-I2	0.710	3.4%
Overall					-4.5%

PQ					
	tf		**binary**		**Diff**
fbis	RB-H1	0.687	RB-H1	0.618	-10.0%
new3	RB-H1	0.666	RB-H1	0.622	-6.6%
re0	RB-H1	0.681	RB-I2	0.697	2.3%
re1	RB-I2	0.626	RB-H1	0.570	-8.9%
tr31	RB-H1	0.810	RB-I2	0.781	-3.7%
tr41	RB-I2	0.829	RB-H1	0.807	-2.6%
tr45	RB-I2	0.783	RB-H1	0.722	-7.7%
wap	RB-H1	0.670	RB-I2	0.691	3.2%
Overall					-4.3%

only slightly worse nearest neighborhoods (with its clustering results being correspondingly closer to *tf* in quality). From this we can conclude that term frequency counts are important in clustering; it is not sufficient to cluster on simple binary term presence/absence.

5.4 BM25 based Feature Weighting

The superiority of *tf* over binary weighting, which I demonstrated in the previous section, indicates that term counts are an important aspect of document clustering. A natural next question is if we can perform some other modification to *tf* which will yield superior clustering results. To that end, I applied BM25 [129], which contains a term frequency dampening component, as a basis for feature weighting. If Q is a query consisting of a set of terms, then X_i's BM25 score with respect to that query (Eq. 5.4) is:

$$Score(Q, X_i) = \sum_{j \in Q} \frac{tf_{ij}(k_1 + 1)}{tf_{ij} + k_1((1 - b) + b\frac{dl_i}{avgdl})} \log(\frac{n}{n_j}), \tag{5.6}$$

where dl_i is a count of the tokens in document X_i:

$$dl_i = \sum_{j=1}^{m} tf_{ij} \tag{5.7}$$

$avgdl$ is the average document length for documents in the collection, and b and k_1 are parameters that are tuned, with $k_1 \geq 0$ and $0 \leq b \leq 1$. Accounting for document length is handled by the b parameter. The term component in Eq. 5.6 *saturates* at a maximum of $k_1 + 1$ as $tf_{ij} \to \infty$, with the gain from increasing *tf* diminishing as the *tf* value increases.

As discussed at the start of this chapter, the previous rationale for the use of BM25 in document clustering was its performance at other tasks. Since its introduction in the early 1990s, the BM25 formula has been widely adopted, and it has repeatedly proved its value across a variety of search domains. The saturation characteristics of the BM25 term weighting function have been identified as a key element in the success of the formula. Unlike other proposed modifications to *tf*, growth of the BM25 term weighting function is relatively rapid when *tf* is small. However, the function quickly approaches an asymptote, limiting the impact of a single term.

Although document retrieval and clustering are not identical tasks, there is now enough clustering research to suggest BM25 might aid in document clustering [15, 45, 101, 153, 156]. This, coupled with the fact that no thorough analysis on the specific benefits of BM25 in document clustering exists, led me to use BM25 in a clustering experiment similar to my initial experiment discussed Section 5.2.

I altered the document representations of Eq. 5.2 and Eq. 5.4 to use the term saturation component of BM25. Specifically, Eq. 5.2 became

$$x_{ij} = \frac{tf_{ij}(k_1 + 1)}{tf_{ij} + k_1((1 - b) + b\frac{dl_i}{avgdl})}, \tag{5.8}$$

and Eq. 5.4 became

$$x_{ij} = \frac{tf_{ij}(k_1 + 1)}{tf_{ij} + k_1((1 - b) + b\frac{dl_i}{avgdl})} \log(\frac{n}{n_j}). \tag{5.9}$$

I refer to Eq. 5.8 as *BM25-tf* and Eq. 5.9 as *BM25-tf-idf*. The selection of values for the parameters b and k_1 is discussed in the next subsection. After creating the BM25 document representations for the various datasets, my experiment followed the procedure described in Section 5.2 exactly, including the length normalization process. Section 5.4.2 discusses the results of the experiment, comparing my BM25 weightings with their non-BM25 counterparts and with each other.

5.4.1 Parameter Estimation

Typical values for the BM25 parameters in document retrieval are $b = 0.75$ and $k_1 = 1.2$ (or 2.0), and previous BM25 clustering papers have mostly used these default values. However, as the b parameter serves the same roll as Euclidean length normalization, I chose to fix $b = 1.0$ in my experiments, and use Euclidean length normalization on top of BM25 to account for document length. I left the task of exploring the best b value to use for future work, and instead focused on estimating the k_1 parameter.

I set aside two of my datasets, fbis and tr31, to use in selecting k_1, while the other six were kept for testing. On each of these two datasets, I ran several of my algorithms using document representations based on both Eq. 5.8 and Eq. 5.9 with $b = 1.0$ and $k_1 = 0 \ldots 100$ in increments of one, various numbers of clusters were used as well. I applied my four clustering quality measures to the resulting clusterings, Figure 5.2 presents the trend in the clustering quality measures when varying the k_1 parameter of Eq. 5.9 with UPGMA clustering (results for other clustering algorithms and the other weighting are mostly consistent with these results).

While the plots in Figure 5.2 fluctuate, it is clear that a low value of k_1 such as the typical document retrieval value of 1.2 or 2.0 is not appropriate. Further, setting k_1 too high diminishes performance, although this is much less pronounced. While a more complex analysis of which k_1 is best would be appropriate, I selected a value of $k_1 = 20$ for use in my experiments based on these plots.

5.4.2 Results

From my previous experiments, we saw that *tf* and *tf-idf* behaved differently based on the dataset and algorithm, thus is made sense to compare *BM25-tf* versus *tf*, and *BM25-tf-idf* versus *tf-idf*. Table 5.10 shows the improvement by dataset of *BM25-tf* over *tf*, and Table 5.11 shows the improvement by dataset of *BM25-tf-idf* over *tf-idf*. Table 5.12 and Table 5.13 show the improvement by algorithm for *BM25-tf* over *tf* and *BM25-tf-idf* over *tf-idf* respectively.

One can see from Table 5.10 and 5.11 that using BM25 weightings improves the average clustering quality results for all clustering quality measures. Both weightings offer approximately the same improvement over their non-BM25 counterparts. The by-algorithm re-

Table 5.10: Improvement by *BM25-tf* over *tf*.

Dataset	NMI	FQ	PQ	EQ
new	-0.8%	-4.5%	-1.5%	-0.7%
re0	4.2%	2.5%	2.6%	2.3%
re1	2.2%	-0.5%	1.2%	1.4%
tr41	2.5%	0.5%	1.9%	2.5%
tr45	3.7%	3.2%	2.8%	3.6%
wap	4.2%	3.3%	2.5%	3.7%
Overall	2.7%	0.8%	1.6%	2.1%

Table 5.11: Improvement by *BM25-tf-idf* over *tf-idf*.

Dataset	NMI	FQ	PQ	EQ
new3	2.3%	3.4%	2.9%	1.2%
re0	13.1%	4.6%	6.0%	6.4%
re1	-1.0%	0.5%	0.7%	-0.7%
tr41	1.6%	-0.8%	1.0%	2.1%
tr45	-3.1%	-3.9%	-1.9%	-1.8%
wap	4.4%	4.1%	3.5%	3.5%
Overall	2.9%	1.3%	2.0%	1.8%

Table 5.12: Improvements by algorithms when using *BM25-tf* over *tf* weighting.

Main Type	Algorithm	NMI	FQ	PQ	EQ
Hierarchical	Clink	6.6%	3.5%	5.7%	8.0%
	RB-K-means	4.5%	2.0%	2.3%	3.3%
	Agglo-H1	2.4%	0.5%	1.3%	1.7%
	RB-H1	1.6%	0.0%	0.4%	1.1%
	UPGMA	2.6%	-0.5%	1.1%	2.2%
	RB-I2	1.7%	0.4%	0.8%	1.2%
	Agglo-I2	1.6%	0.2%	0.8%	1.0%
	Slink	-1.8%	0.0%	0.0%	-0.2%
	Overall	2.4%	0.8%	1.6%	2.3%
Partitional	PCA-K	7.3%	5.5%	4.5%	5.0%
	NMF-NC	3.7%	2.4%	3.2%	3.4%
	PAM	3.9%	2.9%	2.4%	2.5%
	Spect-Un	3.8%	1.3%	2.9%	3.1%
	Spect-RW	3.2%	1.8%	1.9%	2.7%
	Direct-H1	2.9%	1.2%	2.1%	2.3%
	K-means	2.9%	1.6%	1.4%	2.2%
	Spect-Sy	3.0%	0.5%	1.5%	2.4%
	Direct-I2	1.9%	1.1%	1.6%	1.5%
	Overall	3.6%	2.0%	2.3%	2.8%

Table 5.13: Improvements by algorithms when using *BM25-tf-idf* over *tf-idf* weighting.

Main Type	**Algorithm**	**NMI**	**FQ**	**PQ**	**EQ**
Hierarchical	Clink	18.0%	11.8%	8.2%	7.5%
	RB-I2	3.2%	2.6%	2.5%	1.9%
	UPGMA	3.3%	0.3%	1.8%	2.4%
	RB-H1	2.4%	1.2%	1.8%	1.6%
	Agglo-H1	2.1%	0.9%	1.3%	1.3%
	Agglo-I2	1.5%	0.5%	1.7%	0.7%
	Slink	-3.0%	0.2%	-0.2%	-0.3%
	RB-K-means	-3.3%	-4.3%	-2.3%	-2.3%
	Overall	3.5%	1.7%	1.9%	1.6%
Partitional	PCA-K	10.0%	6.1%	7.8%	9.0%
	Spect-Un	5.1%	3.1%	3.7%	3.3%
	PAM	5.4%	1.9%	2.6%	3.0%
	Spect-RW	5.0%	1.6%	2.4%	3.0%
	K-means	2.6%	2.0%	3.3%	3.0%
	Spect-Sy	4.4%	0.9%	2.1%	2.4%
	Direct-H1	0.5%	-0.7%	0.0%	0.0%
	NMF-NC	-0.6%	-0.2%	0.0%	0.0%
	Direct-I2	-1.2%	-1.1%	-0.3%	-1.6%
	Overall	3.5%	1.6%	2.4%	2.5%

sults in Table 5.12 and 5.13 reveal that the large majority of algorithms benefit from BM25 weighting. It is worth noting that the four best performing algorithms for either *tf* and *tf-idf* from my previous experiment (specifically Agglo-H1, Agglo-I2, RB-I2, and RB-i1) all improve when BM25 term saturation is used. The benefit of BM25 term saturation is likely due to its effect on nearest neighborhoods, this is visible in Figure 5.1. From Figure 5.1, we can see that *BM25-tf* always has an equal or better neighborhood than *tf*, likewise, *BM25-tf-idf* has better neighborhoods than *tf-idf*. With respect to comparing *BM25-tf* and *BM-tf-idf*, they follow a pattern similar to that of *tf* and *tf-idf*. For example, Table 5.14 shows the improvement by algorithm from using *BM25-tf-idf* over *BM25-tf*. The algorithms that benefit most from *tf-idf* can be seen to benefit most from *BM25-tf-idf*. When I analyzed the behavior of *BM25-tf-idf* versus *BM-tf* by dataset. I found it to be similar to the behavior of *tf-idf* versus *tf* as well. Additionally, the relative nearest neighborhood behaviors of *BM25-tf-idf* versus *BM25-tf* in Figure 5.1 follow the same pattern as *tf-idf* versus *tf*.

Table 5.14: Improvements by algorithms when using *BM25-tf-idf* over *BM25-tf* weighting.

Main Type	**Algorithm**	**NMI**	**FQ**	**PQ**	**EQ**
Hierarchical	UPGMA	20.0%	18.3%	17.3%	22.8%
	Agglo-H1	4.7%	4.5%	3.0%	3.8%
	Agglo-I2	4.4%	3.4%	3.3%	3.9%
	RB-K-means	5.0%	3.5%	2.6%	3.7%
	RB-I2	3.7%	2.3%	2.3%	3.1%
	RB-H1	2.4%	1.2%	1.8%	1.6%
	Clink	-0.6%	4.9%	0.5%	-2.4%
	Slink	0.7%	-0.2%	-0.2%	0.0%
	Overall	5.0%	4.7%	3.8%	4.6%
Partitional	NMF-NC	14.6%	8.5%	9.4%	10.2%
	PAM	13.2%	9.1%	9.3%	10.2%
	Spect-Un	8.3%	7.6%	6.4%	7.3%
	Spect-RW	7.4%	6.3%	4.8%	6.4%
	Direct-I2	8.4%	5.1%	5.0%	6.1%
	Direct-H1	8.6%	4.6%	4.0%	5.8%
	K-means	7.8%	4.6%	4.7%	5.7%
	Spect-Sy	4.9%	4.2%	3.5%	4.1%
	PCA-K	6.0%	2.7%	1.9%	3.0%
	Overall	8.8%	5.9%	5.4%	6.5%

On average (by algorithm, dataset, best algorithm, and nearest neighborhoods), *BM25-tf-idf* is somewhat better than *BM25-tf*, and notably better than any of the other three weightings. From my BM25 weighting experiments in this section I conclude that BM25 term saturation is superior to raw term count information when used as a component of feature weighting in document clustering. Further, the behavior of my BM25 weightings are very similar to their non-BM25 counterparts with respect to which clustering algorithms and datasets benefit the most from their application.

5.5 Summary and Discussion

In this chapter I examined the merits of applying *tf-idf* term weighting to document clustering through an experiment involving a variety of clustering algorithms, datasets, and clustering quality measures. I found that the *idf* component of *tf-idf* weighting does influence clustering results, but this result can be either positive or negative when compared against *tf* weighting alone. On average, *tf-idf* produces better results than *tf*, but the benefit of using *tf-idf* depends very heavily on the exact dataset and the clustering algorithm used. Binary weighting was also examined, and I determined that it is noticeably inferior to *tf* weighting.

An interesting point to come out of these experiments was the bias in the clustering algorithms and ECQMs I used. Certain clustering algorithms favored certain ECQMs. For example, UPGMA is biased towards FQ. I found that the clustering quality measures I used are not perfectly correlated. However, NMI, PQ, and EQ are reasonably correlated. FQ is poorly correlated with the other three measures. The algorithmic preferences of the ECQMs I used are relatively consistent across different weighting functions, but there are some notable exceptions such as NC-NMF, RB-K-means, and UPGMA ranking notably better when *tf-idf* weighting is used. In general, my findings reinforce my discussion at the outset of this chapter: insufficiently thorough experiments are problematic for clustering. As an example of what can go awry with insufficient experiments, I might have concluded that Direct-H1 is an excellent document clustering algorithm if I only used FQ in my experiments, and then attempt to design a new clustering algorithm, comparing it against Direct-H1. However, Direct-H1 actually has poor results by three other commonly used ECQMs, so comparing against it is not what I should be doing. Rather, I should compare my new algorithm against something like Agglo-I2, which is ranked highly by *all* the ECQMs I investigated, and further I should perform the comparison using multiple ECQMs and datasets.

I proposed and evaluated the use of the BM25 term weighting function in clustering. This function is noted for its term saturation characteristics. I showed that using it in place of the standard *tf* component in both *tf* and *tf-idf* leads to an improvement in clustering results.

With respect to future research related to these experiments, automatic estimation of the k_1 parameter by dataset is of particular interest. This is because I am unsure if the best k_1 to use in a setup like my own is consistent across datasets, or if it varies a little, or if it varies a great deal.

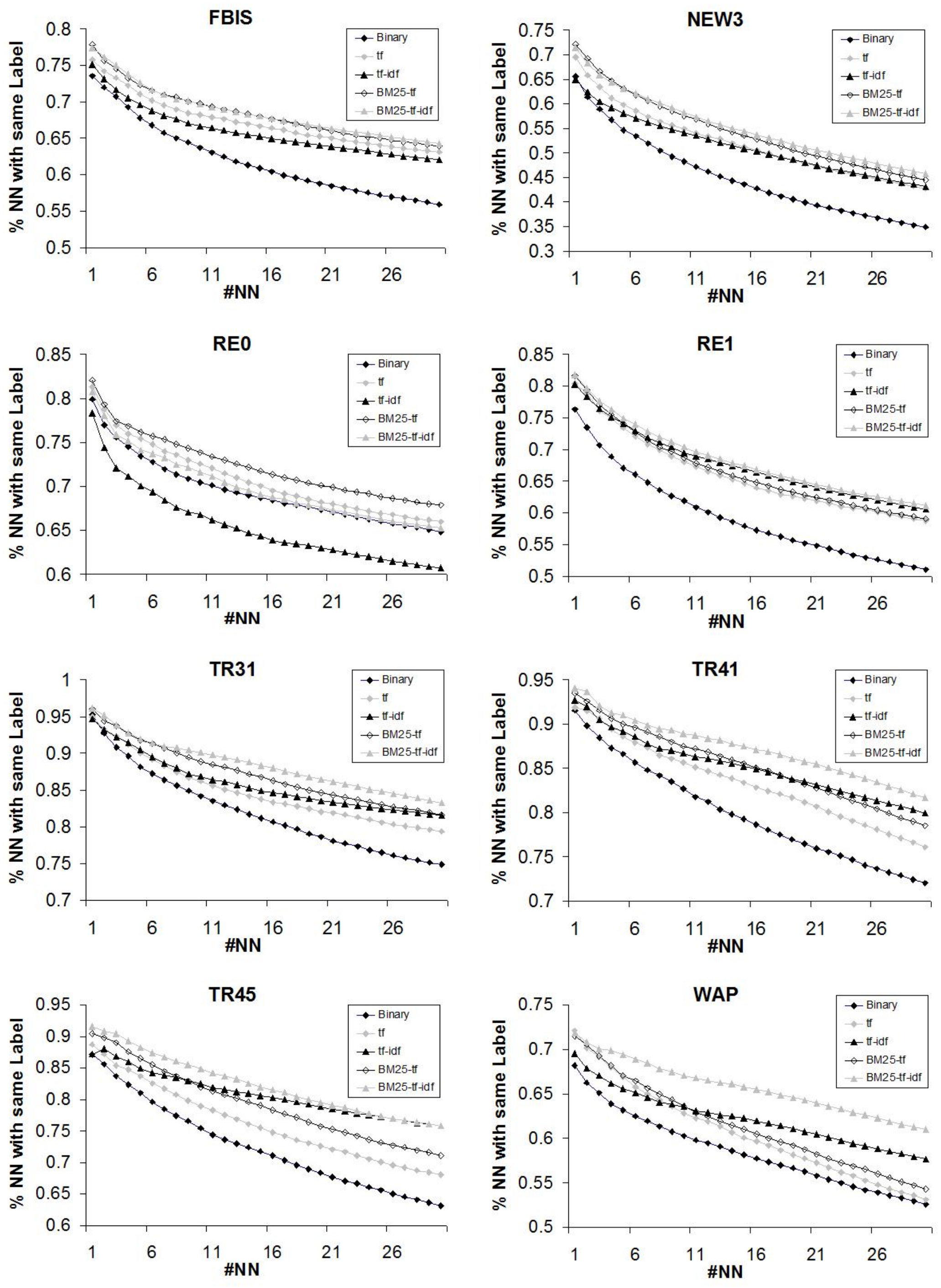


Figure 5.1: Percentages of r-nearest neighbors (using cosine) that share the same label for each dataset.

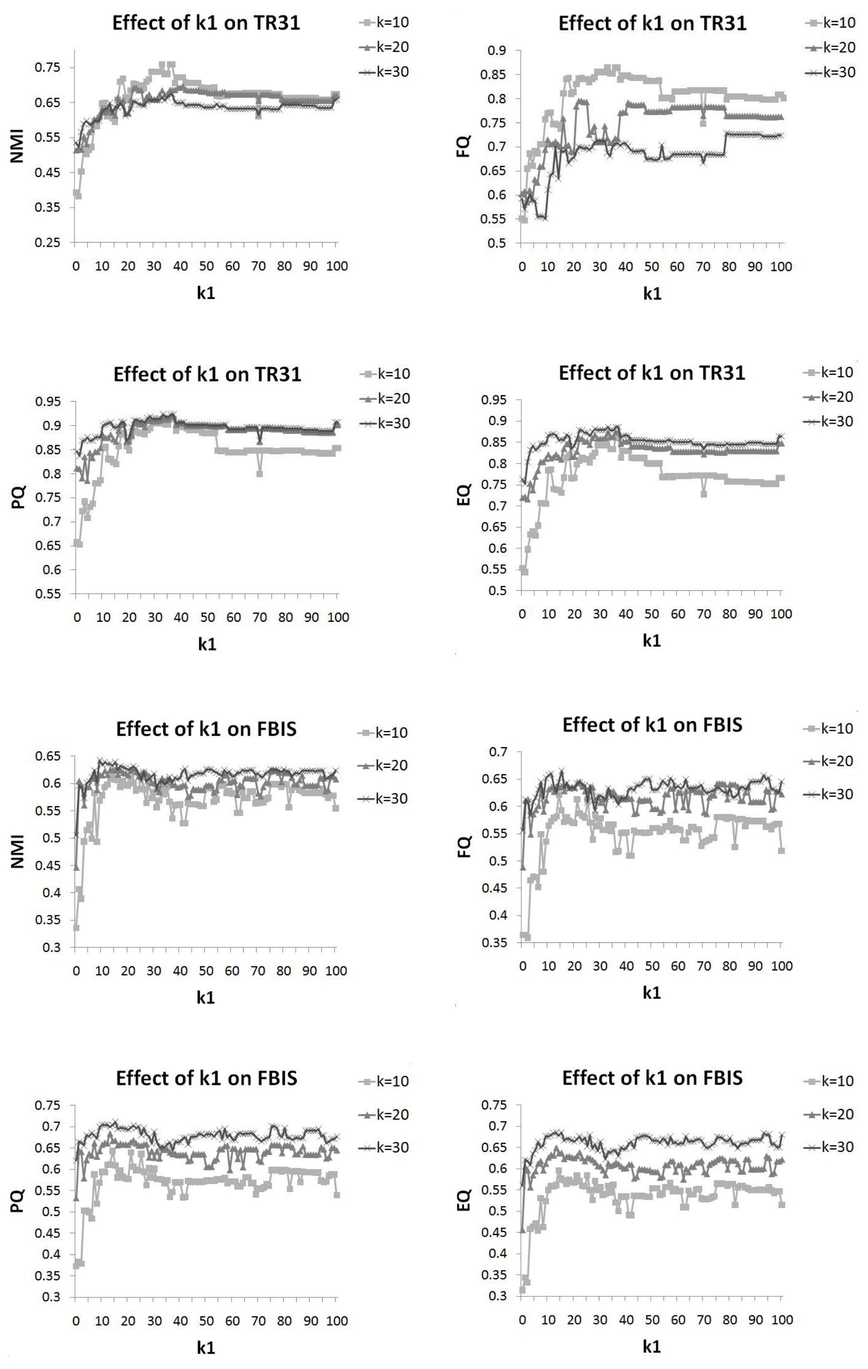


Figure 5.2: The effect on my clustering quality measures when using UPGMA clustering on the *BM25-tf-idf* document representation while varying k_1 from 0 to 100. k is the number of clusters in the clusterings.

Chapter 6

Clustering Clarity

In this chapter I present informativeness, my new ICQM that estimates the clarity of a clustering. Section 6.1 develops informativeness using classification accuracy as a basis. Section 6.2 describes a generalization of informativeness. Section 6.3 compares informativeness to some other well-known ICQMs, as well as some implementations of its generalized form. I will show informativeness is more robust than all the alternatives; it behaves well on a wide variety of the synthetic dataset structures. Using an experimental procedure similar to that in Section 6.3, I will show in Section 6.4 that informativeness behaves in a similarly superior fashion on several commonly used real datasets. In the following chapter I will use informativeness in a real application: email spam filtering.

6.1 Informativeness

Recall that in my discussion of clarity in Chapter 1, I modeled a clustering as being on a dataset that was an independent and identically distributed sample of a population. I defined the clarity of a clustering as: *How well a human expert in the data type of its population can assign previously unseen members of that population to the most appropriate cluster in the clustering.* As a concrete example of this definition, consider the sample of objects given in Fig. 6.1. Let us assume, for simplicity's sake, that the objects in the sample contain all possible valid combinations of features in the population. Fig. 6.2 gives three clusterings of this sample: one by color, one by number of edges, and one by shape. The clusterings all have high clarity in that if we present a previously unseen object from the population to a human, they can trivially identify the cluster in which to place the object for each of the three clusterings. However, in the case of the clustering in Fig. 6.3, no combination of features can be used to assign, with 100% certainty, in which cluster a previously unseen object from the population should belong in. From this we can say that the clustering in Fig. 6.3 has low clarity.

A simple anecdote of mine illustrates relative clarity. I showed my daughter, being seven years of age at the time, the clusterings in Fig. 6.2 and Fig. 6.3. When I asked her what the groups were based on, she quickly identified what the clusters in the high clarity clusterings were based on, but was confused by the low clarity clustering (pausing for some time, and eventually saying "I don't know".)

There are two potential concerns with my definition of a clustering's clarity. The first is that is it meaningful. This question can be answered by considering the purpose of clustering: the creation of groups of objects, where objects within groups are similar, and

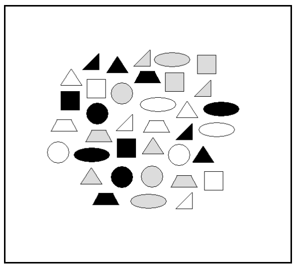


Figure 6.1: A sample of a population.

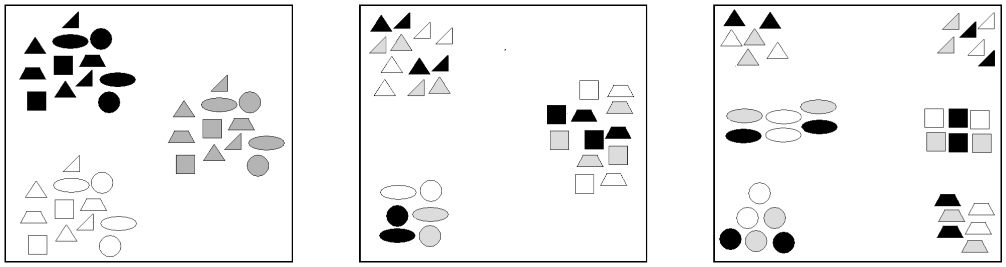


Figure 6.2: Three clusterings of the sample in Fig. 6.1 with high clarity. The first clustering is by color, the second by number of edges, and the third by shape.

objects in different groups are dissimilar. Intuitively, the higher the clarity a clustering exhibits, the more it must exhibit this kind of structure, so clarity is always a meaningful property.

The second concern is whether my clarity definition is machine computable. Computers certainly cannot emulate arbitrary human experts at this point in time. Further, often there is no access to populations from which datasets are drawn. However, with some thought we can see that machines possess a tool for estimating clarity that handles both of these problems: classification.

While nothing can perfectly replace human assignments, classifiers can be used as a machine's proxy for them. A classifier can treat a clustering as a set of classes to train on, allowing the assignment of previously unseen objects into clusters to measure clarity. Because the dataset the clustering is on is an independent and identically distributed

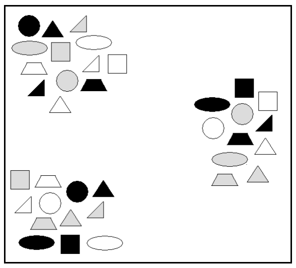


Figure 6.3: A clustering of the objects in Fig. 6.1 with low clarity.

sample of the population, we can use this process, combined with techniques like crossfold validation, to obtain an estimate of how well the population can be classified using the clustering without ever needing access to the entire population. Given this, we may use classifiers to estimate clarity.

I will now formalize how I estimate clarity with classifiers. The notation I will use here is as follows: X is a dataset whose members are an independent and identically distributed samples of a population X^*; n is the sample size of X; C is a clustering of X; and c_i is the ith cluster in C. Let k be the number of clusters in C. A simple approach to estimating clarity, based on the discussion of the preceding paragraph, is to train a classifier of some type f using C as the labels of X. The accuracy of the classifier, obtained through crossfold validation, can be interpreted as a machine estimate of C's clarity.

Let x_j be the jth object in X and let c_{x_j} be the cluster id of x_j for the clustering C. Assume we have applied crossfold validation using a classifier of type f on C to obtain predicted clusters of every $x_j \in X$. Let f_{x_j} be the predicted cluster id of $x_j \in X$ from this process. Let $r_f(c_i)$ be defined as follows:

$$r_f(c_i) = \frac{|\{x_j \in X : (x_j \in c_i) \wedge (c_{x_j} = f_{x_j})\}|}{n}$$

One possible estimate for the clarity of C is:

$$\mathrm{A}(C, X, f) = \sum_{i=1}^{k} r_f(c_i). \tag{6.1}$$

Eq. 6.1 is simply the classification accuracy of a classifier of type f when trained on C, obtained through crossfold validation. Unfortunately, Eq. 6.1 is not a useful estimate of clarity as it trivially approaches its maximum for many types of classifiers as the size of the largest cluster approaches the size of the entire dataset. This happens regardless of the actual contents of clusters, which is a very undesirable property. In a previous work [156], which was not centered around estimating clarity but rather just using classification accuracy to pick good clusterings, I handled this issue by normalizing Eq. 6.1 by the accuracy of a *trivial classifier* T, where I defined a trivial classifier as one that assigns all points of X to the largest cluster of C. My estimate of T's classification accuracy was:

$$A_T(C, X) = \frac{\max_{i=1..k} |c_i|}{n}, \tag{6.2}$$

with *normalized accuracy (NA)*:

$$\mathrm{NA}(C, X, f) = \frac{A(C, X, f)}{A_T(C, X)}, \tag{6.3}$$

being used as my measure for ranking the goodness of clusterings. I had success in locating meaningful clusterings among many candidates using NA with a linear SVM classifier.

Using NA as a measure of clarity still has several problems. One problem is that NA accounts for unbalanced cluster sizes in an ad-hoc manner that heavily favors size balanced clusterings. For example, with binary clusterings, a clustering with a 75%/25% split of

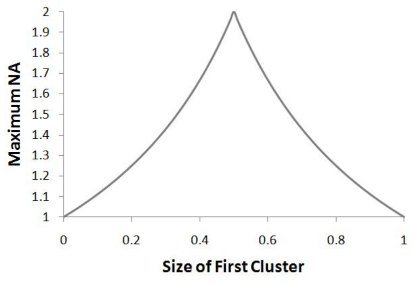


Figure 6.4: Maximum NA on a binary clustering with perfect classification accuracy.

points must have 1.5 times the classification accuracy of a 50%/50% split to obtain the same score from Eq. 6.3. A direct consequence of this kind of penalty is that there may be no way for a clustering, no matter how accurate of a classifier it produces, to obtain a higher NA than a clustering with more balanced cluster sizes (see Fig. 6.1). What is really needed from the normalization process is the penalization of unjustifiably size unbalanced clusterings (i.e. clusterings where there is not enough evidence to support smaller clusters as meaningful) while still allowing size unbalanced clusterings, in general, to potentially score as high as size balanced clusterings.

A second issue is that no accounting for the change in $A_T(C,X)$ caused by increasing the number of clusters is made in NA. As k increases, $A_T(C,X)$ decreases greatly (for perfectly sized balanced clusterings, $A_T(C,X) = 1/k$), meaning clusterings with more clusters will tend to score higher from Eq. 6.3.

Finally, using a single classifier type in computing NA unnecessarily restricts what clusterings can obtain high clarity as individual classifier types may only be accurate for clusterings with particular structures in them. I will deal with all three of the above issues in my final estimation of clarity.

I begin by noting that there is a more principled way to use classifier predictions to estimate clarity than is done in Eq. 6.3. Consider the process of repeatedly classifying unseen members of X^*. Imagine the output of this process as a stream of cluster ids from C. Let $p(c_i)$ be the percentage of cluster ids in the stream that are c_i. Then information theory tells us that to minimize the size of the stream over the course of infinitely many classifications we should assign each cluster id a code of length $-\log(p(c_i))$ to represent it in the stream. While we do not know $p(c_i)$ in general, we can use a maximum likelihood estimate of it from C:

$$p(c_i) = \frac{|\{x_j \in X : c_{x_j} = c_i\}|}{n}, \tag{6.4}$$

in minimizing the expected size of the stream. Fig. 6.5 gives an example of using maximum likelihood estimates of $p(c_i)$ values to derive an encoding to use for a stream in the situation described above.

31212333

(a)

Cluster Id	**MLE** $p(c_i)$	**MLE** $-log(p(c_i))$	**Binary Encoding**
1	0.25	2	00
2	0.25	2	01
3	0.50	1	1

(b)

Figure 6.5: Using MLE $p(c_i)$ values to obtain a binary encoding. (a) is the set of cluster ids for a clustering. (b) gives the binary encoding sizes of the cluster ids that we obtain from using Eq. 6.4, as well as an example of a real encoding that uses those sizes.

Minimizing the stream size is, by itself, not useful in measuring clarity, but consider that not everything in the stream is correct, sometimes the classifier will assign incorrect cluster ids to population members of X^*. By using $r_f(c_i)$ as a maximum likelihood estimate of the probability that a cluster id in the stream will be c_i and that c_i is the correct cluster id for the object the id corresponds to, we can estimate the average amount of correct data in the stream per cluster id it contains:

$$\mathrm{AI}(C, X, f) = -\sum_{i=1}^{k} r_f(c_i) \log(p(c_i)), \tag{6.5}$$

While Eq. 6.5 is similar to cross entropy it is not identical as $\sum_{i=1}^{k} r_f(c_i)$ may be less than one; when $\sum_{i=1}^{k} r_f(c_i)$ is one (perfect prediction of cluster ids through crossfold validation) we have:

$$\begin{aligned} \mathrm{AI}(C, X, f) &= -\sum_{i=1}^{k} r_f(c_i) \log(p(c_i)) \\ &= -\sum_{i=1}^{k} p(c_i) \log(p(c_i)) \\ &= H(C), \end{aligned}$$

where $H(C)$ is the entropy of C. Fig. 6.6 gives an example of computing AI for some classifier type trained on the clustering in Fig. 6.5.

$$31\boxed{1}1\boxed{1}333$$

(a)

$$\mathrm{AI}(C, X, f) = -(0.25 \log(0.25) + 0.0 \log(0.25) + 0.5 \log(0.50)) = 1$$

(b)

Figure 6.6: Evaluating AI. (a) is the predicted cluster ids labels for clustering from Fig. 6.5 for some classifier type f. Incorrect predicted labels are in boxes. (b) is the resulting AI. Log bases are 2 in this example.

One can think of AI as information weighted classification accuracy. The weight of a classification into cluster c_i is an estimate of how much correct data we are given, on average, from seeing a cluster label c_i from the classifier. An alternative interpretation is that AI is a balance between micro-averaged classification accuracy (Eq. 6.1) and macro-averaged classification accuracy:

$$\mathrm{A}_{\mathrm{macro}}(C, X, f) = \frac{1}{k}\sum_{i=1}^{k}\frac{r_f(c_i)n}{|c_i|}. \tag{6.6}$$

Specifically, in micro-averaging (Eq. 6.1) the relative weight of a correct classification for an object in cluster c_i is one, in macro-averaging (Eq. 6.6) it is $n/|c_i|$, and in AI (Eq. 6.5) it is $\log(n/|c_i|)$.

AI arises naturally from my definition of clarity. Given this, I chose to use it as a basis for clarity instead of Eq. 6.1, Eq. 6.3, or Eq. 6.6. However, note that AI still suffers from the same issues that I noted for NA: cluster size bias, number of clusters bias, and the use of only one classifier type in its estimation of clarity.

I deal with the first two issues I noted simultaneously by applying the correction for random chance for ICQMs in Eq. 4.4. As discussed in Chapter 4, this adjustment is identical to the adjustment for random chance in active use for ECQMs [82], and results in an equation of the form:

$$I(C, X, f) = \frac{\mathrm{AI}(C, X, f) - E[\mathrm{AI}(C, X, f)]}{\max(\mathrm{AI}(C, X, f)) - E[\mathrm{AI}(C, X, f)]}, \tag{6.7}$$

where $E[A(C, X, f)]$ is the expected value of $\mathrm{AI}(C, X, f)$ by chance and $\max(\mathrm{AI}(C, X, f))$ is the maximum value of $\mathrm{AI}(C, X, f)$ possible given a fixed C; noting that the latter is $H(C)$, Eq. 6.7 is equivalent to:

$$I(C, X, f) = \frac{\mathrm{AI}(C, X, f) - E[\mathrm{AI}(C, X, f)]}{H(C) - E[\mathrm{AI}(C, X, f)]}. \tag{6.8}$$

Defining a random classifier as one that places objects in each class with equal likelihood, I estimate:

$$E[\mathrm{AI}(C,X,f)] = -\sum_{i=1}^{k} \frac{p(c_i)}{k} \log(p(c_i)) \tag{6.9}$$

$$= \frac{H(X)}{k}. \tag{6.10}$$

This estimation is derived from noting that the maximum likelihood estimate of my random classifier placing an unseen object from the population into c_i is $1/k$, and that the maximum likelihood estimate of this being correct is $p(c_i)$. Eq. 6.9 is therefore a maximum likelihood estimate of $E[\mathrm{AI}(C,X,f)]$.

Substituting Eq. 6.10 in to Eq. 6.8, we obtain:

$$I(C,X,f) = \frac{\mathrm{AI}(C,X,f) - \frac{H(C)}{k}}{\frac{(k-1)H(C)}{k}}. \tag{6.11}$$

Eq. 6.11 solves my first and second issues. All clusterings are scaled between 0 and 1, except a one-clustering, for which it is not defined. I argue that having no one-clustering definition is acceptable as investigating Eq. 6.11 values for possible number of clusters allows us to detect if a particular number of clusters is favored over others. If none are, we may then conclude that a one-clustering appropriate. Any none one-clustering can obtain a 1 from Eq. 6.11 if it can be used to produce a classifier with perfect classification accuracy, regardless of the cluster sizes it contains. Thus the potential for an unbalanced clustering to score well is there. Penalizing unjustifiably size imbalanced clusterings is done while still allowing this by using the $-\log(p(c_i))$ cluster code sizes. Because I use these, misclassifying objects that actually belong in smaller clusters is more costly than misclassifying objects that belong in larger clusters. At the same time, a classifier is more likely to make mistakes when training on smaller clusters as they have fewer training examples. From this, I can say that it is harder for size imbalanced clusterings to have high clarity unless the clusters within them are very distinguishable. Note that this property will also help prevent clusterings with many small size balanced clusters from scoring high informativeness unless they are, likewise, highly distinguishable.

With respect to the second issue, that of scaling with k, I again note that $I(C, X, f)$ is normalized between 0 and 1 for $k \geq 2$. Further, I have extracted the expected clarity by random chance from informativeness ($E[\text{AI}(C, X, f)]$)—that is to say, there is almost no variable random chance component in Eq. 6.11. This is especially important because we can see that $E[\text{AI}(C, X, f)]$ changes both with k and the sizes of individual clusters (see the random chance section in Chapter 4 for further discussion on this).

The final problem, that of a single classifier type being too restrictive, is easily fixed by using a set of different types of classifiers in computing AI. Each classifier can be trained and tested separately on the same C and X and the highest $\text{AI}(C, X, f)$ from among the classifiers can be used in estimating the clarity of the clustering. Note that the highest is used because my definition of clarity is based on expert opinion. The expert, in a machine context, is the classifier that performs the best. AI, adjusted to handle multiple classifiers, becomes:

$$\text{AI}(C, X, f^*) = \max_{f \in f^*}\left(\sum_{i=1}^{k} r_f(c_i) \log(p(c_i))\right), \tag{6.12}$$

where f^* is the set of classifier types to be trained/tested on C and X. In theory, for the same reason that I take the best result in this computation, I should use as many classifier types in f^* as possible. However, in this thesis, and in practice, it is necessary to restrict classifier types for efficiency reasons. In the remainder of this thesis I show that this restriction does not seem to hamper my final measure of clarity much, it is still highly useful when using just a few classifier types. Because $\mathrm{AI}(C, X, f)$ is the only component of Eq. 6.11 that is based on f, it is safe to simply replace it with $\mathrm{AI}(C, X, f^*)$, yielding:

$$I(C, X, f^*) = \frac{\mathrm{AI}(C, X, f^*) - \frac{H(C)}{k}}{\frac{(k-1)H(C)}{k}}. \tag{6.13}$$

I call the Eq. 6.13 *informativeness*, where $I(C, X, f^*)$ is the informativeness of C, a clustering of X, when using f^* as the selection of classifier types. Informativeness is my ICQM for estimating a clustering's clarity, with higher values being better. The formal algorithmic form for informativeness is given below.

Algorithm 1 Informativeness

```
Input: Clustering $C$, Classifier Types $f^*$, Dataset $X$ {$C$ has two or more clusters}
bestScore ← 0 {initialize the best score found so far to the lowest value possible}
for all $f \in f^*$ do
    T ← crossFoldValidationLabeling($C, X, f$) {obtain the classifier type's predicted labeling of $C$ through crossfold validation}
    if $I(C, X, f) \geq$ bestScore then
        bestScore ← $I(C, X, f)$
    end if
end for{get the best $I(C, X, f)$ score for this clustering (Eq. 6.8)}
return bestScore
```

In Chapter 4 I covered many properties of clustering quality measures. As I discussed there, knowing how specific clustering quality measures behave relative to these is important as it can provide theoretical and practical reasons for using certain clustering quality measures over others. Given this, I analyze how informativeness behaves relative to the ICQM properties discussed in Chapter 4. I omit a discussion on concept, variable random chance, and optimal number of clusters, as I have already discussed these at length above.

One might already have realized that informativeness' behavior with respect to most of the properties in Chapter 4 depends on the classifier type(s) it uses, but it can also be influenced by the exact nature of the crossfold validation used (i.e. the number of folds and which objects are in each fold). Below I present how informativeness behaves when it uses a nearest neighbor classifier (as in Section 6.3.4, except with any number of neighbors used), with leave-one-out crossfold validation. I use a nearest neighbor classifier because it is well-understood and, more importantly, it can use any distance function. This allows me to discuss informativeness as if it has the form $I(C, X, d)$ when necessary. Note that with some work, we can derive positive results with respect to the properties below for informativeness when many other kinds of classifiers (SVMs, nearest centroid, etc.) are used in it (including multiple classifiers).

Lemma 3 (For a fixed C and X let f and f' be classifier types such that $\forall_{c_i \in C} r_f(c_i) \leq r_{f'}(c_i)$. Then $\forall_{f^*} I(C, X, f^* \cup f) \leq I(C, X, f^* \cup f')$)**.**

Proof. *It suffices to show that $AI(C, X, f) \leq AI(C, X, f')$. Let $\triangle_{c_i} = r'_f(c_i) - r_f(c_i)$. Then*

$$\begin{aligned}
AI(C, X, f') &= -\sum_{i=1}^{k} r_{f'}(c_i) \log(p(c_i)) \\
&= -\sum_{i=1}^{k} (r_f(c_i) + \triangle_{c_i}) \log(p(c_i)) \\
&= -\sum_{i=1}^{k} r_f(c_i) \log(p(c_i)) + -\sum_{i=1}^{k} \triangle_{c_i} \log(p(c_i)) \\
&= AI(C, X, f) + -\sum_{i=1}^{k} \triangle_{c_i} \log(p(c_i)).
\end{aligned}$$

As $-\sum_{i=1}^{k} \triangle_{c_i} \log(p(c_i)) \geq 0$, we have $AI(C, X, f) \leq AI(C, X, f')$. □

Lemma 4 (Informativeness is consistent when using a nearest neighbor classifier with leave-one-out crossfold validation)**.**

Proof. *Given the previous lemma, I need only show that for any X, C, d, and d' that is* *a C consistent variant of d, all correct classifications made using d are correct when using d'. If C consists of a single cluster, this is trivially true. Otherwise, for some correctly classified object x_i let x_j and x_l be elements of X such that $x_i \sim_C x_j$, $x_i \nsim_C x_l$ and*

$$d(x_i, x_j) < d(x_i, x_l). \tag{6.14}$$

From the definition of a consistent variant we have:

$$d(x_i, x_j) \geq d'(x_i, x_j)$$

and:

$$d(x_i, x_l) \leq d'(x_i, x_l).$$

Substituting these two inequalities into Eq. 6.14 we obtain:

$$\begin{aligned} d'(x_i, x_j) &\leq d(x_i, x_j) \\ &\leq d(x_i, x_l) \\ &< d'(x_i, x_l). \end{aligned}$$

Let $rNN(x_i)$ be the list of the r nearest neighbors that were used to predict x_i's label when using d. In order to make x_i become incorrectly classified when using d' we must find an $x_j \sim_C x_i$, $x_j \in rNN(x_i)$ and an $x_l \nsim_C x_i$, $x_l \notin rNN(x_i)$, such that $d(x_i, x_j) < d(x_i, x_l)$ and $d'(x_i, x_j) > d'(x_i, x_l)$. However, I have just shown that such an (x_j, x_l) pair cannot exist, independent of $rNN(x_i)$. Given this, a correctly classified object using d is correctly classified when using d'. $\square$

With respect to the refinements of consistency I discussed in Section 4.2, I note that informativeness is neither improving within or between-consistent, regardless of the classifiers it uses, as it has a maximal value beyond which improvements to a clustering can no longer increase informativeness' score on the clustering. This is not a problem though, as these are not properties I suggested a clustering quality metric must have, only points of interest.

Noise Tolerance: In the case of a nearest neighbor classifier, changing a single distance $d(x_i, x_j)$ in a clustering can result in at most two additional misclassifications/correct classifications when crossfold validation is used (one for x_i, one for x_j), but using classifiers based on larger numbers of nearest neighbors diminishes the odds of any classification changing. Removing an individual object in a clustering can effect a nearest neighbor classifier's crossfold validation accuracy more but, again, using more nearest neighbors in each classification reduces this problem. This means nearest neighbors classifiers that use several nearest neighbors are fairly noise robust.

Lemma 5 (Informativeness is scale invariant when using a nearest neighbor classifier with leave-one-out crossfold validation)**.**

Proof. *I simply note that multiplying all distances in d by a uniform amount does not change the ordering of nearest neighbors for any object regardless of C, X, and d. Thus we have $AI(C, X, d) = AI(C, X, \lambda d)$ and $I(C, X, d) = I(C, X, \lambda d)$, as required.* □

Lemma 6 (Informativeness is rich when using a r nearest neighbor classifier with leave-one-out crossfold validation if the size of each cluster in the clusterings it is used on are always at least $r + 1$)**.**

Proof. *Recall that satisfying richness requires that for any fixed M, X, and C, we are able to define some distance function d such that $C = \arg\max_{C' \in C^*} M(C', X, d)$, where C^* is* *the set of all possible clusterings of X. For any fixed C and X let d be a distance function where for all $x_i, x_j \in X$ we have $d(x_i, x_j) = 0.0$ if $x_i = x_j$, 0.1 if $x_i \sim_C x_j$ and $x_i \neq x_j$, and $d(x_i, x_j) = 1$ otherwise. Then the r nearest neighbors of every $x_i \in X$ have the same true cluster id as x_i. This assures correct prediction of all cluster ids by the classifier, giving $AI(C, X, d) = H(C)$, $I(C, X, d) = 1$, and $C = \arg\max_{C' \in C^*} I(C', X, d)$.* $\square$

Note the cluster size restriction above, whereas consistency and scale invariance had none. I argue that this is not a failing of informativeness, but rather that it does not make sense to apply nearest neighbor classification using a number of neighbors that is close to the smallest class being trained on. The number of nearest neighbors used should be much smaller than smallest class size.

Time Complexity: Assuming a distance matrix has been pre-computed, as we did for other ICQMs, when using a nearest neighbor classifier, informativeness can be computed using leave-one-out crossfold validation in $O(n^2 \log(n))$ time. This is the time it takes to sort each object's neighborhood list (after which only $O(nr)$ time is required to classify all objects). Other classifiers may yield higher time complexities for informativeness. Even using by-classifier optimization to speed up generating crossfold validation label predictions, informativeness can be said to have a high time complexity cost for an ICQM.

One can say from the above discussions that informativeness can behave positively with respect to all the properties for ICQMs discussed in Chapter 4, with the possible exception of time complexity, when using many different kinds of classifiers. Noting that many ICQMs have problems with the properties in Chapter 4, this provides a rationale for using informativeness besides its generality. In the following sections I will provide further rationale for its use with synthetic and real dataset experiments. The following chapter will show a real application of informativeness with similarly positive results.

6.2 Generalizing Informativeness

A generalization of informativeness follows from noting that Eq. 6.5 is a measurement between two partitions: 1) the clustering; and 2) the partitioning predicted by a classifier of type f when trained on the clustering using crossfold validation. One can therefore generalize informativeness to the following algorithm (assuming higher ECQM scores are better, otherwise change minValue to maxValue, and $\geq$ to $\leq$ in the if statement):

Algorithm 2 GeneralizedInformativeness

```
Input: ECQM $M$, Clustering $C$, Classifier Types $f^*$, Dataset $X$
bestEcqmScore $\leftarrow$ minValue($M$) {initialize the best score found so far to the lowest value possible}
**for all** $f \in f^*$ **do**
    T $\leftarrow$ crossFoldValidationLabeling($C, X, f$) {obtain the classifier type's predicted labeling of $C$ through crossfold validation}
    **if** $M(C,T) \geq$ bestEcqmScore **then**
        bestEcqmScore $\leftarrow M(C,T)$
    **end if**
**end for**{get the best ecqm score for this clustering}
**return** bestEcqmScore
```

Using Eq. 6.11 as the ECQM in this algorithm yields base informativeness. In the following sections, I will show that this is a reasonable choice relative to some other ECQMs.

6.3 Synthetic Dataset Experiment

In my synthetic dataset experiment I compared informativeness to several other ICQMs, among which were implementations of its generalized form, on clusterings of synthetic datasets. The synthetic datasets I used in the experiment are detailed in Section 6.3.1, the clustering algorithms in Section 6.3.2, the ICQMs I compared informativeness against in Section 6.3.3, and finally the classifiers used to obtain informativeness and its generalized form in Section 6.3.4.

For the experiment, I generated 50 instances of each dataset and clustered each instance with each clustering algorithm using from two to 20 clusters, giving 7600 (50*8*19) clusterings of each dataset. I then computed each ICQM for each clustering; ten-fold cross validation was used in computing informativeness and implementations of its generalized form. In addition, I computed how well each clustering corresponded to the true labeling of its dataset using the NMI variant from Eq. 3.22.

The correlation between NMI and the ICQMs, as well as using the ICQMs to select the optimal number of clusters for each dataset, is analyzed in Section 6.3.5. The analysis will show that informativeness is superior to the other ICQMs I consider.

6.3.1 Datasets

I used five synthetic datasets in the experiment, these are detailed below. They represented a variety of the structures that are commonly used when analyzing clustering evaluation measures; included were datasets where cluster boundaries were linear, non-linear, and/or fuzzy.

2GAUSS consisted of two Gaussian clusters with identity covariance, each with 500 points in two dimensions, with means of (0,-4) and (0,4).

6GAUSS consisted of six Gaussian clusters with identity covariance, each with 500 points in two dimensions. The means of these Gaussians were (0,0), (6,0), (10,0), (5,5), (0,-5), and (-5,0). There was some overlap between the clusters in this dataset.

ELONGATED had three elongated clusters in two dimensions. The first elongated cluster made a line from (-0.5,-0.5) to (0.5,0.5). 300 points were spaced evenly along the line. Gaussian noise with a mean of 0 and standard deviation of 0.01 was added to each dimension for each point. The second and third clusters were generated in an identical manner, except the second cluster was shifted by -2 on the x-axis, and the third cluster was shifted by $+2$ on the x-axis.

CUBE was a set of 8 clusters, each had a mean centered on one of the 8 corners of a 10x10x10 cube centered at (0,0,0). 200 points were uniformly distributed within each cluster between $(x-4, y-4, z-4)$ and $(x+4, y+4, z+4)$, where (x, y, z) was the mean of the cluster.

RINGS consisted of 2 ring clusters centered around (0,0), a larger outer ring with radius 2 and a smaller inner ring of radius 1. 400 points were evenly spaced by degrees on the inner ring. A random noise component between 0 and 0.1 was then added to the x and y coordinates of all the points. The outer ring was created in a similar fashion, except 1200 points were used.

6.3.2 Clustering Algorithms

I used eight clustering algorithms in the experiment; *k-means, UPGMA, Repeated Bisecting k-means, Clink, Slink, Agglo-E1, Agglo-I1*, and *Agglo-I2*. The implementation/manner of use of these was exactly as detailed in Section 5.2.2. I did not use every clustering algorithm from Section 5.2.2 for two reasons: 1) some are not meant for use on very low dimensionality datasets such as some of those in this chapter (ex. spectral clustering algorithms); and 2) some of the clustering algorithms were similar (ex. RB-I2 optimizes the same objective function as Agglo-I2).

6.3.3 Competing Evaluation Measures

I compared informativeness against five standard ICQMs and four implementations of its generalized form. The standard ICQMs I compared against were: *Silhouette*, the *Davies-Bouldin Index*, the *C-Index*, *B/W*, and *point-wise margin*. The first four of these measures are exactly as detailed in Chapter 3; Euclidean distance was used whenever these measures required a distance function. My final ICQM, point-wise margin, is defined as:

$$\mathrm{PWM}(X) = \frac{1}{n}\sum_{i=1}^{m}\sum_{o_j \in x_i} \frac{\min\limits_{o_l \in x_i, o_l \neq o_j} d(o_l, o_j)}{\min\limits_{o_l \in D, o_l \notin x_i} d(o_l, o_j)}. \tag{6.15}$$

Point-wise margin is similar to relative margin (Eq. 3.18) but has a time complexity of $O(n^2)$. It is the average over all objects in X of the closest object to them in their cluster divided by the closest object to them that is in another cluster. Intuitively, point-wise margin will function on well separated clusters regardless of their exact structure; it is, however, poorly suited to evaluating overlapping clusters. I used Euclidean distance with point-wise margin.

The four implementations of informativeness' generalized form that I compared it against followed the process detailed in Section 6.2. Each implementation used the same classifier types as informativeness (see the following section), but used a distinct ECQM. In particular, the ECQMs they used were: adjusted Rand Index (*inf-ARI*), purity quality (*inf-PQ*), entropy quality (*inf-EQ*), and F-measure using $\beta = 1$ (*inf-F1*). Each of these ECQMs has been discussed in Section 3.2.

6.3.4 Classification Algorithms

Although it would be ideal to use as many classifier types as possible in computing informativeness (and implementations of its generalized forms), for efficiency reasons I restricted myself to four classifier types in the synthetic dataset experiment. Given the results in Section 6.3.5, I can say in hindsight that this restriction also serves to show that informativeness is useful with even just a few well chosen classifiers.

The four classifier types I used were: a *five nearest neighbors (5NN)* classifier, a *multiclass polynomial kernel SVM (PSVM)*, a *C4.5 decision tree (C4.5)*, and a *nearest centroid (NC)* classifier. With the exception of NC, I used Weka[1] implementations of the classifiers during my experiment; I implemented NC myself.

With respect to classifier parameters, NC used Euclidean distance in classifying. For all the other classifiers, I used their default parameter settings in Weka.

6.3.5 Analysis

I analyzed two aspects of my ICQM results on the clusterings: overall performance of the ICQMs, and using the ICQMs to pick the optimal number of clusters for each dataset. To compare my ICQMs' overall performance I first created a ranking of all 7600 clusterings for each dataset by each ICQM, from best to worst; I also generated a similar ranking for each dataset using NMI, giving ten rankings in total per dataset. The function of the NMI rankings was that of gold standards (i.e., they ranked the clusterings for each dataset by their true quality). This choice was inline with the fact that a large amount of clustering literature (including many works referenced in this thesis) use NMI variants as

[1]`http://www.cs.waikato.ac.nz/ml/weka/`

their measure of true clustering quality. Further, the version of NMI I chose is arguably the most common of the variants.

For each dataset, I computed Kendall's τ [93] (τ-b values specifically) between each of its ICQMs' rankings and NMI's ranking. Table 6.1 gives the τs from these computations. As τ measures the correlation between two rankings, and the NMI rankings were my gold standard rankings, we can say that the higher the τs an ICQM has in Table 6.1, the better it performed in my experiment.

	Dataset					
Evaluation Measure	**2GAUSS**	**6GAUSS**	**ELONGATED**	**CUBE**	**RINGS**	**Average**
Informativeness	0.290	0.267	0.748	0.334	0.388	0.406
inf-ARI	**0.717**	0.329	0.703	0.318	0.415	**0.497**
inf-PQ	0.690	0.297	0.722	0.192	0.367	0.454
inf-EQ	0.550	**0.370**	0.618	**0.492**	0.458	**0.497**
inf-F1	0.653	0.289	0.723	0.194	0.367	0.445
Silhouette	0.159	0.319	0.599	0.469	0.391	0.388
Davies-Bouldin	0.238	0.291	0.322	0.355	0.178	0.277
C-Index	-0.727	0.272	-0.754	0.460	-0.070	-0.163
B/W	-0.635	0.228	-0.744	0.427	-0.101	-0.170
Point-Wise Margin	0.248	0.131	**0.753**	0.005	**0.407**	0.309

Table 6.1: Kendall's τ-b correlations between rankings of the synthetic dataset clusterings by the ICQMs and the rankings made by NMI. τ ranges from -1 (anti-correlated) to 1 (perfectly correlated), 0 is uncorrelated. The higher the τ, the better the measure was performing. All results are significant with $p \sim 0$.

It is immediately noticeable from Table 6.1 that informativeness measures had higher average τ than any of the other ICQMs analyzed. The difference in average is substantial for even the closest pairing, with informativeness at $\tau = 0.406$, roughly 5% higher than the nearest non-informativeness ICQM (Silhouette at $\tau = 0.388$); this suggests that informativeness measures are highly effective ICQMs. In this experiment, inf-ARI and inf-EQ were notably better in average performance than the other informativeness measures.

From the results in Table 6.1, I suggest that the non-informativeness ICQMs I tested overfit particular clustering structures. For example, C-Index performs well on 6GAUSS ($\tau =$

0.272) and CUBE ($\tau = 0.460$), yet it was substantially anti-correlated with NMI on ELONGATED ($\tau = -0.754$) and not substantially correlated with NMI in any way for the RINGS dataset. Silhouette and Davies-Bouldin appeared to do less overfitting, but were still noticeably inferior overall to the informativeness measures. It is extremely important to note that in practice, we do not necessarily know if a dataset fits a certain ICQM well, so we should not use "best-case" behavior (such as point-wise margin on ELONGATED) to justify the use of an ICQM in general. Worst-case and average-case, however, make perfect sense to use, and informativeness measures were superior to all the non-informativeness ICQMs I tested for both of those.

For the ICQMs besides the informativeness measures, one might surmise that their failings were due partly to my specific parameter selections (i.e., Euclidean distance, using centroids in Davies-Bouldin, etc.). However, this only serves to highlight a problem with using them, and other measures, as ICQMs. If the intention of ICQMs is to be able to identify good clusterings without human input, how are their parameters selected?

With respect to picking the optimal number of clusters for each dataset, I grouped my 7600 clustering for each dataset by sample. Then, for each ICQM, I recorded the optimal number of clusters it estimated for each sample of the dataset, where the estimation was simply the number of clusters in the best scoring clustering for that ICQM. Table 6.2 gives the frequency of optimal number of cluster estimations for each ICQM on each dataset.

A naive analysis of Table 6.2 may lead one to believe that informativeness measures picked the right number of clusters only decently. This analysis is somewhat misleading because we do not know what the clusterings selected as containing the optimal number of clusterings for each ICQM actually look like. For example, an ICQM could predict two clusters on a 2GAUSS sample, which is correct, but the clustering from which it derived

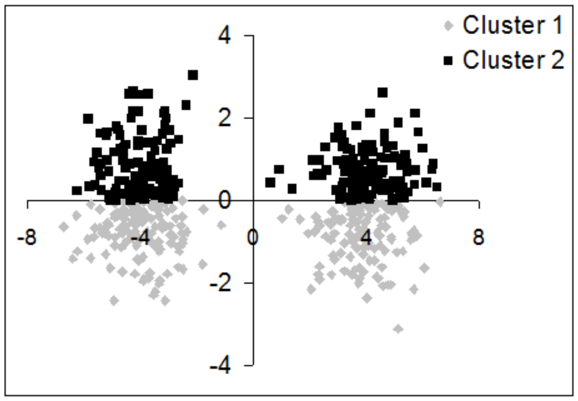


Figure 6.7: A two-clustering of a 2GAUSS sample that has the correct number of clusters (two) but it entirely unrelated to the correct two-clustering.

this could be one where half of each true cluster was in each predicted cluster (see Fig. 6.7), although this particular problem did not occur in the experiment for any clustering algorithm. Given this, I present a more thorough analysis below. This analysis will show that informativeness measures are actually performing better than the other ICQMs at picking numbers of clusters—they almost always selected clusterings related in some meaningful way to the true clustering.

The results for every ICQM tested, except B/W and C-Index, were almost perfect for 2GAUSS.

On the ELONGATED dataset, most of the ICQMs tested performed perfectly. As with 2GAUSS, B/W and C-Index performed very poorly on this dataset. Inf-ARI, inf-EQ, and point-wise margin favored two clusters on this dataset. On investigating this, I found that all the two-clusterings selected the optimal number of clusters by those ICQMs consisted of two true clusters grouped together, with a third true cluster on its own. While not ideal, this choice made sense given the structure of the dataset. Therefore, I argue that those three ICQMs performed decently on ELONGATED.

For the CUBE dataset Davies-Bouldin and Silhouette estimated eight clusters, with the clusters being the true eight. Informativeness estimated two, four, and eight clusters, with eight being the most frequent estimation. On investigating this, I discovered that the two-clusterings informativeness preferred for the CUBE dataset were ones that could be created by cutting the CUBE dataset with a single hyperplane going through the origin, parallel to one of the three axes. This means that the two clusters each contained four complete true clusters. Similarly, the four-clusterings informativeness preferred could be created by cutting the CUBE dataset with two hyperplanes going through the origin, where each one was parallel to a different axes, resulting in each cluster containing two complete true clusters. The results for the other informativeness measures were very similar to the above. Given that the informativeness measures detected multiple meaningful clusterings/number of clusters of the CUBE dataset, one of which was the true clustering/number of clusters, I suggest that they performed as well as Davies-Bouldin or Silhouette, and certainly better than the other ICQMs (which never estimated eight at all).

For the RINGS dataset, point-wise margin worked most effectively at picking the number of clusters. It always used the correct two-clustering in its estimation: two rings, one surrounding the other. Whenever the informativeness measures estimated two clusters, it was the correct two. Further, whenever they estimated too many clusters, the clusters in the estimation were always pure—each cluster contained points from only one ring; Fig. 6.8 gives examples of this. This is reasonable given the structure of the dataset. The remaining ICQMs performed uniformly poorly on the RINGS datasets.

For 6GUASS, Silhouette performed very well and Davies-Bouldin performed relatively well. Informativeness measures seemed to perform poorly, guessing two clusters often, and six only sometimes. The other ICQMs performed very poorly, never estimating six.

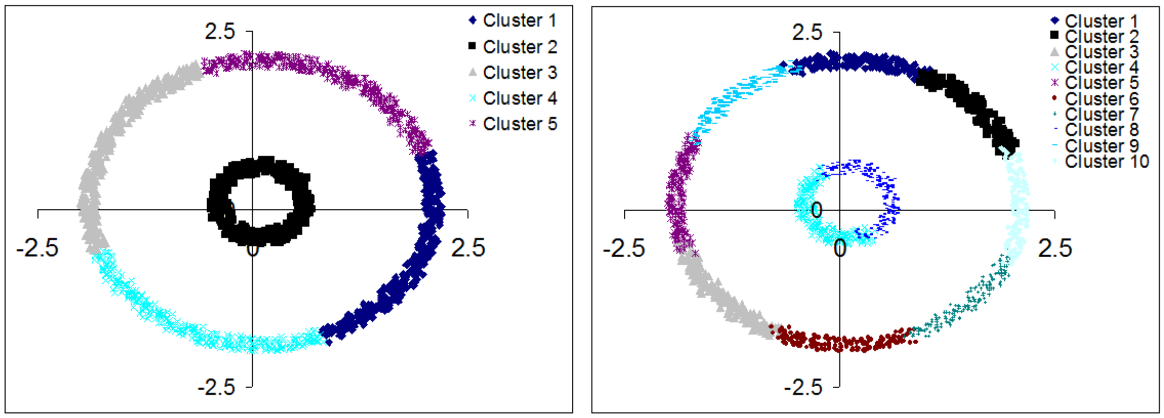


Figure 6.8: A five-clustering and a ten-clustering, each of which was selected by informativeness measures as containing the optimal number of clusters for a different sample of the RINGS dataset.

Again, further investigation showed that the informativeness measures were performing much better than it appeared. The two-clusterings they selected could be created by dividing the dataset into a left/right split, with the (0,0), (-5,0), and (0,-5) true clusters on the left side, and the (6,0), (10,0), and (5,5) on the right side; this is a highly sensible two-clustering when we inspect it visually (see Fig. 6.9), perhaps even more than the true six-way clustering. This was also the split that point-wise margin selected as containing the optimal number of clusters most of the time. Unfortunately, point-wise margin never detected the true six-way clustering. I argue that informativeness measures performed the best of all the ICQMs on 6GAUSS at picking the number of clusters. They selected two choices for the number of clusters using very meaningful clusterings, one of which was the true number of clusters, while the other ICQMs selected either one good number of clusters (Silhouette), often had unclear cluster boundaries in their optimal clusterings (see Fig. 6.10 for an example), or never even estimated six clusters.

My analysis above suggests that informativeness measures are the best of the ICQMs I considered at estimating number of cluster values for datasets. Each informativeness

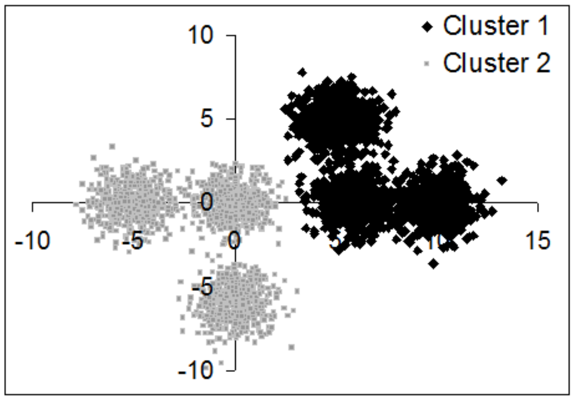


Figure 6.9: An example of the two-clusterings for the 6GAUSS dataset selected by informativeness as containing the optimal number of clusters.

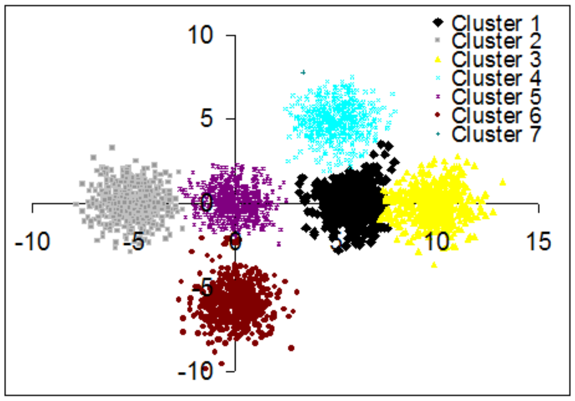


Figure 6.10: A seven-clustering for the 6GAUSS dataset selected by Davies-Bouldin as containing the optimal number of clusters. Cluster 7 is a singleton cluster. Further, the boundaries between clusters 1, 3, and 4 are unclear.

measures seemed to perform roughly as well as the others. It is worth noting that Silhouette performed extremely well on the Gaussian-like datasets (everything except RINGS).

6.4 Real Dataset Experiment

My real dataset experiment was identical to the synthetic dataset experiment except as follows: 1) the datasets I used are detailed in Section 6.4.1; 2) I generated 20 samples (not 50) of each dataset, in each case this was done by randomly sampling 50% the dataset; and 3) informativeness and its generalized forms used only 5NN classifier with leave-one-out crossfold validation. The ICQMs, types of informativeness, and NMI variant used were exactly as detailed in Section 6.3. Section 6.4.2 presents an analysis of the results of the real dataset experiment.

6.4.1 Datasets

I used five datasets in my real dataset experiment: *IMAGE*, *ISOLET*, *IRIS*, *PENDIGITIS*, and *WINE*. Each of these datasets is available from the UCI machine learning repository[2] and has been used in multiple papers. They are summarized in Table 6.3 and are detailed below briefly.

IMAGE is a collection of features for 3x3 pixel regions of 7 outdoors images. The true labels of this dataset are the physical features found in the regions, either brickface, sky, foliage, cement, window, path, or grass. IRIS is a collection of measurements on three different species of iris plants. It is notable that only one species of iris can be linearly

[2] `http://archive.ics.uci.edu/ml/`

separated from the others. ISOLET is a sample of a spoken letter dataset. Each object in the dataset is a collection of features corresponding to the utterance of a single letter of the English alphabet. 150 speakers were used to generate the full dataset, where each speaker uttered each letter twice. The true labels are the letters spoken. I used a 2398 sized sample of this dataset in my experiment. PENDIGITS is a collection of evenly-spaced (x,y) features for hand-written digits from 0 to 9. WINE is a set of chemical analysis results for samples of wine from 3 cultivar in Italy.

6.4.2 Analysis

Table 6.4 gives Kendall's τ value between ICQMs and NMI on the real datasets. One can see that informativeness' behavior on the real datasets was similar to its behavior on the synthetic datasets. Specifically, it was always substantially and significantly correlated with NMI. Its average performance was far superior to its nearest variant, inf-ARI (being nearly 50% better than it), and was superior to the best non-informativeness ICQM I tested as well (C-Index). Informativeness and inf-ARI seem to generalize to the higher dimensionality of real datasets well. It is worth noting that inf-PQ, inf-EQ, and inf-F1 exhibited very poor performance in this particular experiment. This is likely due to the fact that PQ, EQ and F1 do not correct for number of clusters, cluster size, or random chance. The synthetic datasets, having very clear clusterings, may not have required these properties. However, the real datasets were substantially more complicated and most had many more dimensions, likely necessitating such properties. Of the non-informativeness ECQMs, C-Index and B/W performed well. The remaining ICQMs functioned poorly.

On another note, the small correlations in Table 6.4 highlights the fragility of ICQMs, they are often inconsistent with true labelings on real datasets.

Given the results of my synthetic and real dataset experiments, I suggest that informativeness measures are good ICQMs to use in helping humans identify good clusterings. My experiments suggest that the basic version of informativeness is highly effective in general, and particularly useful relative to non-informativeness ICQMs when users have no/minimal biases they want involved during the evaluation of their clusterings (i.e. they want good clustering of any form), and/or they want multiple good clusterings of a single dataset. With respect to which version of informativeness is the best, my results suggest that basic informativeness, or an informativeness variant that uses an ECQM that is corrected for random chance (such as ARI does), is the best choice overall. As a final point, it should be said that the experiments in this chapter represent only a small fraction of those possible. Therefore, much additional experimentation (beyond this chapter and the next) is necessary to truly validate informativeness and its generalized form.

Table 6.2: Estimations of the number of clusters in the synthetic datasets by the ICQMs.

2GAUSS

Evaluation Measure	Number of Clusters									
	2	3	4	5	6	7	8	9	10	11+
Informativeness	50	0	0	0	0	0	0	0	0	0
inf-ARI	50	0	0	0	0	0	0	0	0	0
inf-PQ	50	0	0	0	0	0	0	0	0	0
inf-EQ	49	1	0	0	0	0	0	0	0	0
inf-F1	50	0	0	0	0	0	0	0	0	0
Silhouette	50	0	0	0	0	0	0	0	0	0
Davies-Bouldin	47	3	0	0	0	0	0	0	0	0
C-Index	0	0	0	0	0	0	0	0	0	50
B/W	0	0	0	0	0	0	0	0	0	50
Point-wise Margin	50	0	0	0	0	0	0	0	0	0

6GAUSS

Evaluation Measure	Number of Clusters									
	2	3	4	5	6	7	8	9	10	11+
Informativeness	44	2	0	0	4	0	0	0	0	0
inf-ARI	42	5	0	0	3	0	0	0	0	0
inf-PQ	49	1	0	0	0	0	0	0	0	0
inf-EQ	35	5	1	0	9	0	0	0	0	0
inf-F1	48	2	0	0	0	0	0	0	0	0
Silhouette	0	0	0	0	50	0	0	0	0	0
Davies-Bouldin	0	0	0	6	19	11	5	7	1	1
C-Index	0	0	0	0	0	0	0	0	0	50
B/W	0	0	0	0	0	0	0	0	0	50
Point-wise Margin	50	0	0	0	0	0	0	0	0	0

ELONGATED

Evaluation Measure	Number of Clusters									
	2	3	4	5	6	7	8	9	10	11+
Informativeness	0	50	0	0	0	0	0	0	0	0
inf-ARI	36	14	0	0	0	0	0	0	0	0
inf-PQ	0	50	0	0	0	0	0	0	0	0
inf-EQ	50	0	0	0	0	0	0	0	0	0
inf-F1	0	50	0	0	0	0	0	0	0	0
Silhouette	0	50	0	0	0	0	0	0	0	0
Davies-Bouldin	0	50	0	0	0	0	0	0	0	0
C-Index	0	0	0	0	0	0	0	0	0	50
B/W	0	0	0	0	0	0	0	0	0	50
Point-wise Margin	50	0	0	0	0	0	0	0	0	0

CUBE

Evaluation Measure	Number of Clusters									
	2	3	4	5	6	7	8	9	10	11+
Informativeness	14	1	3	0	0	0	32	0	0	0
inf-ARI	12	3	2	1	0	0	32	0	0	0
inf-PQ	13	3	1	2	0	0	31	0	0	0
inf-EQ	6	1	1	3	2	0	37	0	0	0
inf-F1	21	5	0	0	0	0	24	0	0	0
Silhouette	0	0	0	0	0	0	50	0	0	0
Davies-Bouldin	4	0	0	0	0	0	46	0	0	0
C-Index	0	0	0	0	0	0	0	0	0	50
B/W	0	0	0	0	0	0	0	0	0	50
Point-wise Margin	50	0	0	0	0	0	0	0	0	0

RINGS

Evaluation Measure	Number of Clusters									
	2	3	4	5	6	7	8	9	10	11+
Informativeness	12	7	13	8	4	5	0	0	1	0
inf-ARI	38	0	1	1	5	4	0	1	0	0
inf-PQ	22	6	5	5	3	5	2	2	0	0
inf-EQ	40	1	1	3	2	3	0	0	0	0
inf-F1	25	6	5	5	2	3	2	2	0	0
Silhouette	0	0	0	0	0	0	0	0	0	50
Davies-Bouldin	4	0	0	0	0	0	0	8	2	40
C-Index	0	0	0	0	0	0	0	0	0	50
B/W	0	0	0	0	0	0	0	0	0	50
Point-wise Margin	50	0	0	0	0	0	0	0	0	0

Table 6.3: The datasets used in my real dataset experiment.

Dataset	**# of Objects**	**# of Features**	**# of Classes**	**# of Objects in largest Class**
IMAGE	2100	19	7	300
ISOLET	2398	617	26	52
IRIS	150	4	3	50
PENDIGITS	7494	16	10	780
WINE	178	13	3	71

Table 6.4: Kendall's τ-b correlations between rankings of the real dataset clusterings by the ICQMs and the rankings made by NMI. τ ranges from -1 (anti-correlated) to 1 (perfectly correlated), 0 is uncorrelated. The higher the τ, the better the measure was performing. All results are significant with $p \sim 0$.

	Dataset					
Evaluation Measure	**IMAGE**	**ISOLET**	**IRIS**	**PENDIGITS**	**WINE**	**Average**
Informativeness	**0.468**	0.346	0.457	0.194	0.189	0.331
inf-ARI	0.103	0.173	**0.596**	-0.025	0.282	0.223
inf-PQ	-0.384	-0.088	0.563	-0.124	0.172	0.028
inf-EQ	-0.266	0.067	0.549	-0.016	0.204	0.107
inf-F1	-0.375	-0.084	0.550	-0.115	0.173	0.030
Silhouette	-0.029	-0.043	0.146	0.216	0.075	0.073
Davies-Bouldin	-0.338	-0.253	0.401	0.091	0.085	-0.003
C-Index	**0.659**	**0.595**	-0.238	**0.717**	-0.183	0.309
B/W	0.213	0.516	-0.213	0.694	-0.114	0.215
Point-Wise Margin	0.214	0.106	0.311	-0.013	-0.015	0.120

Chapter 7

A Spam Filtering Application

In the previous chapter I provided theoretical reasons for the use of informativeness in terms of the properties in Chapter 4. I also showed, using synthetic and real datasets, that informativeness detects meaningful clusterings for a wide range of datasets. While this is certainly important, recall that in the introduction of this thesis I noted that it is typically considered that the ideal way to evaluate a clustering is situational human assessment, a clustering is 'good' for someone if it does what they want. Thus, to better show that informativeness can evaluate clusterings well, it should be used in a real application.

In this chapter I show the use of informativeness in a real application: email spam filtering. I first show that clustering, in general, can be a useful tool in email spam filtering. I then present two algorithms that leverage clusterings to produce highly effective email spam filters that require few user labelings of emails. Both algorithms take an ICQM parameter. I will show that using informativeness for this parameter produces more effective spam filters than using several other commonly used ICQMs, proving that informativeness is of practical use.

7.1 Spam Filtering and Clustering

In laboratory experiments, it is not uncommon for email spam filters to yield outstanding results. For example, the top spam filters from the TREC 2005 spam filtering track [32] obtained AUC scores of 0.999 and better. However, field tests of actual spam filters have yielded much poorer results, with overall misclassification rates on the order of 5% being observed, even under high-quality commercial spam filters [34]. At this rate, one in 20 email messages are incorrectly assigned either a spam or ham (i.e., non-spam) label. Part of the disparity between laboratory experiments and field tests may be explained by noting that many laboratory experiments work under the assumption of a 'perfect' user, that is to say, all the training data is labeled, and further that it is labeled correctly.

As noted by Mojdeh and Cormack [114], the assumption of a perfect user is a highly unrealistic model. Not only are users very unlikely to label all their email messages, they may not label any messages at all, instead relying on their spam filter's default behavior. Further, users make mistakes while labeling. These considerations lead me to examine alternative user models that might be more applicable in practice.

The specific user model I investigate in this chapter is that of a user willing to label some small number of email messages from a much larger collection of unlabeled email, all in advance of the actual spam filtering process (i.e., an offline process) in order to obtain an improved spam filter. We can envision my user model as fitting into the situation where an individual has an existing email account and is changing (or tuning) the spam filter on the account. This alternative user model engenders a semi-supervised learning problem, where both labeled and unlabeled data can be used to train a spam filter. Naturally, other user models are possible, and my work may be adapted to many of them.

Semi-supervised spam filtering is itself not a new research topic. Cormack [33] used dynamic markov compression (DMC) to create improved spam filters by combining labeled and unlabeled training data. The ECML/PKDD Discovery Challenge Workshop [21] investigated the use of semi-supervised spam filtering when the training and test data came from different sources. The results of that challenge suggest that semi-supervised methods may be superior to fully supervised learning methods. However, Mojdeh and Cormack [114] obtained the opposite result when they tested a number of semi-supervised spam filtering techniques, including dynamic markov compression, logistic regression [63], and transductive support vector machines [89]. In a later work, Mojdeh and Cormack [115] developed a method for using unlabeled training data, based on singular value decomposition, that out-performed other semi-supervised methods. Their method also out-performed supervised spam filters when the same number of true labels were available, indicating that the unlabeled training data provided some benefit. Although results with semi-supervised spam filtering have been mixed, there are clear indications that careful use of unlabeled training data can lead to improved spam filtering.

Here I look at using clustering in a semi-supervised spam filtering. The tendency of email datasets to exhibit spam clusters is, as with semi-supervised spam filtering, not a new concept. For example, Li and Hsieh [102] use URLs for clustering spam emails, while Jungsuk et al. [142] investigate feature selection in clustering spam emails. I extend previous notions of clustering and spam filtering by first showing a previously unknown and important result: Almost any reasonable clustering algorithm will naturally produce clusters of almost entirely spam or entirely spam. I then exploit this fact to design two small sample semi-supervised spam filters, based on clustering and ICQMs, the better of which is superior to state-of-the-art small sample semi-supervised spam filters. Further, I show that both my spam filters are most effective when it uses informativeness as its ICQM.

As a starting point, I propose an intuitive rationale for the application of clustering to spam filtering. Since spam filters exhibit very high classification accuracy in laboratory experiments, where large volumes of labeled training data is available, I suggest that the representations and features used for these email messages naturally make ham and spam appear very different to these algorithms. Clustering algorithms are designed to group similar patterns together, thus I posit that a clustering algorithm will generate clusters of mostly spam and mostly ham. Section 7.2 validates this intuition using a number of clustering algorithms and two well-known spam email datasets.

In Section 7.3 I develop two semi-supervised spam filtering methods based on clustering and ICQMs, designed for use in situations where a large amount of unlabeled training data is available, of which only a small portion will be labeled. Both of these methods exploit the nature of email clusters, as demonstrated in Section 7.2, by creating numerous clusterings of the training data as an initial step, using no true labels. An ICQM is then used to select the single best clustering to use in the proceeding steps; again, no true labels are used in this step. The first method then trains a spam filter using the true label of the medoid of each cluster in the best clustering, which are requested from the user. The second method is similar to the first, except that the true label for the medoid is assumed to apply to the email messages in the entire cluster, providing labels for the entire training set, which is then used to train a spam filter.

Section 7.4 presents the results of experiments involving my two semi-supervised spam filtering methods. I compare my two methods using various ICQMs from Chapter 4 and informativeness, as well as comparing all of these against a baseline spam filter trained on a random sample of k email messages and their true labels, where k is equal to the number of clusters (and true labels) used by my two methods. For test datasets, I use the TREC2005 and CEAS2008 spam email datasets. I show that both my methods perform best when using informativeness; in this situation they are much better than random sampling. Of my two methods, I show that the second method is substantially better than the first. Further, when using informativeness, it produces results superior to those previously published for other forms of semi-supervised learning. Section 7.5 summarizes the chapter.

7.2 Clustering Email

Given the discussions in previous chapters, one might be skeptical of using clustering in combination with supervised learning methods. However, spam filtering represents a special case. Visual inspection often trivially reveals when a message is spam or ham. Further, spam filters exhibit exceptionally high classification accuracy. We may posit from these observations, in addition to research on the clustering behaviour of email datasets [142, 102, etc.], that a reasonable clustering algorithm will usually produce clusters of mostly ham or mostly spam with very little crossover.

In the remainder of this section I show that clustering emails with a sufficient number of clusters k (roughly 6 or more) consistently ensure that the above property holds—the resulting clusters are mostly ham or mostly spam, regardless of the exact clustering algorithm.

7.2.1 Datasets

I selected two well-known datasets to use in my experiment: the TREC2005 corpus[1], and the CEAS2008 corpus[2]. To represent these email messages for clustering purposes, I first truncated them all 2500 bytes. I then converted each message into a vector $x_i = (x_{i1}, x_{i2}, \cdots x_{im})$, where

$$x_{ij} = \begin{cases} 1, & \text{if } \mathrm{tf}_{ij} > 0; \\ 0, & \text{otherwise.} \end{cases} \tag{7.1}$$

tf_{ij} is the term frequency of the 4-byte-gram j, obtained from the overlapping 4-byte-grams of email x_i. Following this, I removed all 4-byte gram features from the vectors entirely that did not occur in at least 20 messages. Finally, I Euclidean length normalized each vector:

$$x_{ij} = \frac{x_{ij}}{\sqrt{\sum_{j=1}^{m} x_{ij}^2}} \tag{7.2}$$

I made the above choice of dataset representation/weighting for two reasons. First, while Euclidean length-normalized tf-idf word vectors are commonly used in document clustering (and I showed that Okapi BM25 Euclidean length-normalized vectors are likely even better than that in Chapter 5), spam email datasets are very different from standard document clustering datasets. Secondly, previous research has suggested that 4-byte-grams with binary weighting are excellent features for spam filtering [35]. These two points led to my choice of dataset representation/weighting. Investigating clustering algorithms' performances in spam filtering tasks when using alternative weightings is an avenue of future research.

[1] plg.uwaterloo.ca/ gvcormac/treccorpus/
[2] plg.uwaterloo.ca/ gvcormac/ceascorpus/

Table 7.1: The clustering algorithms used in my spam experiments.

Algorithm	Short	Reference
K-means	K-means	[106]
Repeated Bisecting k-means	RB-K-means	[145]
Random Walk Spectral	Spect-RW	[139]
Principle Component Analysis+K-means	PCA-K	[124]
Unweighted Pair Group Method	UPGMA	[91]

7.2.2 Clustering Algorithms

I selected five well-known clustering algorithms, summarized in Table 7.1. For more details on these algorithms readers should consult Section 5.2.2, as they were implemented/used in the manner described there.

7.2.3 Clustering Quality Measure

My measure of how well a clustering algorithm splits ham and spam was PQ (Eq. 3.46), restated here:

$$\mathrm{PQ}(C,T) = \sum_{c_i \in C} p(c_i)\mathrm{Purity}(c_i,T),$$

where:

$$\mathrm{Purity}(c_i,T) = \max_{t_j \in T} \frac{p(c_i,t_j)}{p(c_i)}.$$

C is a clustering of an email dataset and T is the true labeling of the same email dataset; c_i is the ith cluster in C and t_j is a true label type in T; $p(c_i) = \frac{|c_i|}{n}$ and $p(c_i,t_j) = \frac{|c_i \cap t_j|}{n}$, where n is the number of email messages in the dataset.

If a clustering algorithm was effectively separating ham and spam, I expected $\mathrm{PQ}(C,T) > \frac{max(\text{\#of spam},\text{\# of ham})}{n}$, which is approximately the expected PQ value by random chance.

Table 7.2: Average PQ values obtained by the five clustering algorithms on each dataset as k varies. Random is the expected PQ from creating a clustering by randomly assigning points into clusters.

	TREC2005					**CEAS2008**				
	# of clusters					# of clusters				
Algorithm	2	5	10	25	50	2	5	10	25	50
K-means	0.848	0.854	0.855	0.905	0.921	0.832	0.897	0.947	0.966	0.961
RBK-means	0.848	0.860	0.910	0.949	0.949	0.803	0.933	0.973	0.976	0.979
Spect-RW	0.850	0.877	0.930	0.940	0.940	0.817	0.928	0.971	0.971	0.961
PCA-K	0.853	0.863	0.895	0.951	0.959	0.821	0.870	0.938	0.966	0.980
UPGMA	0.835	0.923	0.950	0.960	0.967	0.803	0.978	0.979	0.983	0.986
Random	0.572					0.803				

7.2.4 Purity Experiment

To show that clustering algorithms naturally separate ham and spam, I first constructed 10 random samples of the TREC2005 and CEAS2008 email datasets. For each sample, each email of the appropriate dataset was included in the sample with some fixed probability (0.015 for TREC2005, 0.01 for CEAS2008). The representation used for the email messages was exactly as detailed in Section 7.2.1, except to increase clustering speed I removed 4-byte-grams on a by-sample basis when the sample did not contain at least 20 messages with that 4-byte-gram. This resulted in approximately 20000 distinct 4-byte-grams per sample.

I ran my five clustering algorithms on each sample of each dataset with the number of clusters k varying from 2 to 50, giving 4900 clusterings in total (49*2*5*10). PQ values were obtained for each clustering. I then averaged the PQ values for each (dataset, clustering algorithm, k) triplet. Average PQ values for some of these triplets are given in Table 7.2. For comparison purposes, a baseline PQ value for each dataset, equal to $\frac{max(\text{\#of spam,\# of ham})}{n}$, is included in the table. As discussed, if a clustering algorithm is doing a good job of separating spam and ham, I expected its PQ values to be higher than the baseline.

It is immediately noticeable from Table 7.2 that for any number of clusters k beyond a very small value ($k > 5$ or so), all the clustering algorithms do better than the baseline. The differences are significant with $p = 0.05$, with the exception of RBK-means and UPGMA at $k = 2$. Moreover, all the significant results substantially outperform the baseline. PQ values rise noticeably with k, with all PQs approaching one (perfect) as k increases. Such strikingly high PQ values suggest that a ham/spam split is a very 'natural' and/or dominant way of partitioning email.

When investigating the relatively poor performance of the clustering algorithms when k was very low (versus higher values for k), I discovered there were simply not enough clusters to represent a spam/ham split properly, possibly due to the skew in the ratio of ham and spam messages. For example, CEAS2008 has more than three times as much spam as ham. Given this ratio, when $k = 2$ we might expect a good clustering algorithm to focus on the differences between spam messages, especially if the algorithm attempts to balance cluster sizes. If we have a approximate understanding of the ratio of spam to ham, we can select k to be large enough that roughly equal-sized clusters could achieve a split between ham and spam.

From Table 7.2 we see that UPGMA provides the best performance overall. Although some of my clustering algorithms perform better than others, all of the clustering algorithms substantially outperform the baseline. From these results, it appears that clustering provides a simple, unsupervised method for separating email spam from ham, but unfortunately without indicating which cluster is which. However, one can take advantage of the structure uncovered by clustering by requesting a label for a selected representative email from each cluster.

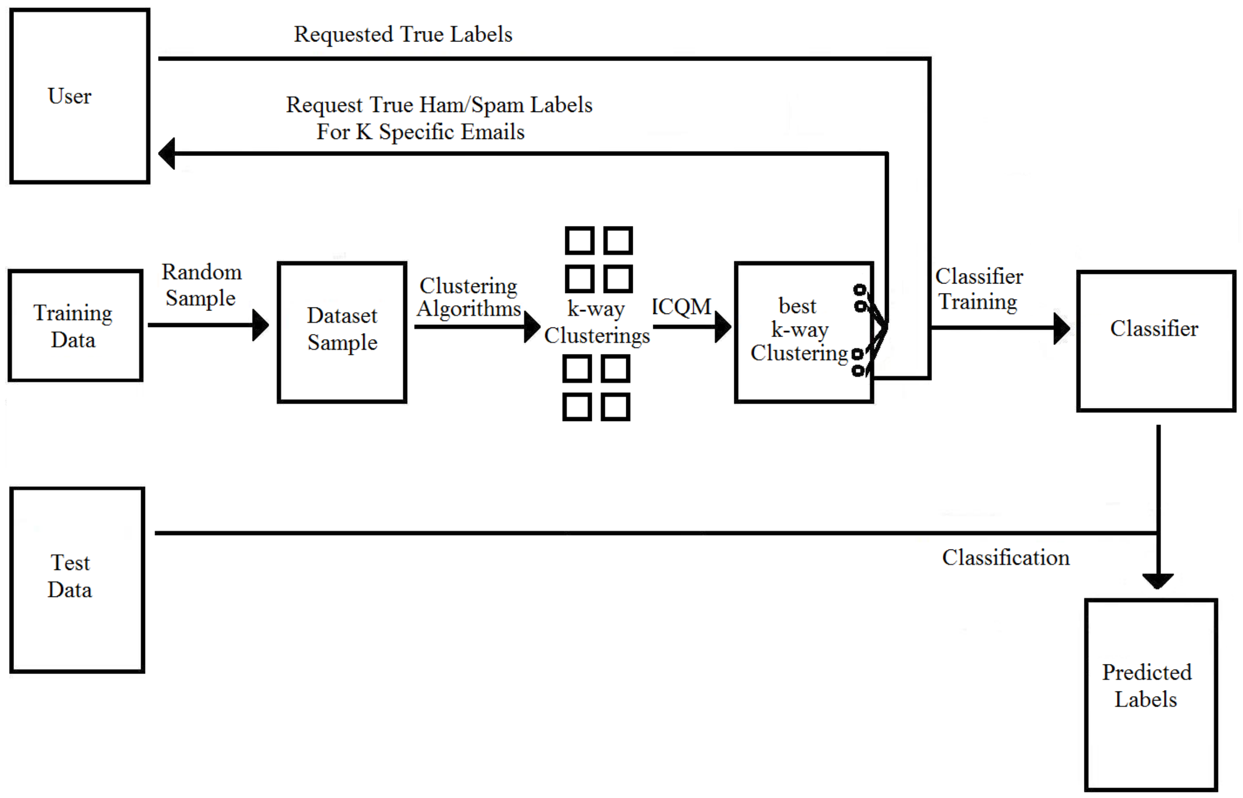


Figure 7.1: A graphical representation of the process my two new spam filtering methods follow.

7.3 Spam Filtering using Clustering

In this section I present my two new methods for spam filtering. Both methods use clustering, an ICQM, and only a small number of true labels.

7.3.1 Algorithms

The general procedure followed by both my methods is relatively simple (see Fig. 7.1 for a graphical representation). Given a user with some training email messages and some new (test) messages they wish to filter, the first step is to create a random sample of the training messages. This sampling is required only to control the size of the input to clustering algorithms of the following step, and is not required if the clustering algorithms can cluster the entire training set quickly enough. The second step is the production of multiple k-way clusterings of the sample using a set of clustering algorithms. After this, an ICQM, such as those in Chapter 3 or informativeness, is used to select the single best clustering from those with which to proceed.

The user is then asked to label a single message from each of the k clusters in the best clustering. After the user labels these k training messages, a classifier is trained using these k messages, and potentially other training messages that use labels derived from those k messages and the properties of the clustering. Finally, the trained classifier is used to spam filter the test data. It is important to stress that at no point in my experiments are any of test messages, labeled or unlabeled, used in the training of the spam filter (hence the separate boxes for test and training data in Fig. 7.1).

From the experiments in Section 7.2, I assume that the clustering step in Fig. 7.1 will produce clusterings containing clusters of mostly ham and mostly spam, regardless of clustering algorithms I use, but I also assume a little more, specifically that each individual cluster in a clustering represents a distinct subgroup of the larger class (ham or spam) they fall into. This assumption leads to my first spam filtering method, which I call *cluster-medoid spam filtering (CMSF)*.

CMSF exploits my assumption that each cluster in a clustering represents a different subgroup of ham or spam by sampling a single representative from each cluster to use in training a spam filter. This approach intuitively provides a better view of the different groups than exist within the dataset than might be provided by a random sample. For example, a random sampling is highly vulnerable to sampling the more common kinds of spam, while CMSF generally ensures that points used in training are different. Further, CMSF also serves my requirement of using a small number of true labels (k, the number of clusters used for the clustering).

For the representative used by CMSF for each cluster in the best clustering, I opted to use the cluster's medoid. Because the medoid of a cluster is centrally located, one may think of it as exemplar of the cluster, making it perhaps the ideal object to use to represent a cluster if only one object can be selected.

A full definition of how CMSF spam filters is provided below in Algorithm 1. There are five parameters in CMSF: k; C^*; M; A; and p. Note that for M, the ICQM, it is assumed that higher scores are better, for ICQMs where this is not the case we can simply multiple scores by -1 and still use Algorithm 1. For C^* and M, the extra parameters they might require (such as distance functions) are assumed to be fixed, so they are not included in the parameter list. The results in Section 7.2 suggest that a wide range of values for k and C^* will result in good spam filters (assuming k is not a very small number). The classifier type for CMSF (the parameter A) can be selected from those spam filtering classifiers known to work well in previous research (such as SVMs and logistic regression). I will show that CMSF is highly effective when the ICQM is set to informativeness, given the choice of a number of ICQMs. Considering these points, the parameters of CMSF are easy to select. Further, the number of labels required from the user by CMSF is only k.

One potential issue with CMSF is that it may not obtain both ham and spam labels, as it does not request true labels of a specific class. This limitation is shared with any method for unsupervised learning, clustering algorithms have no notion of true labels; however, I found in practice this is not an issue when $k \geq 8$. On testing the clusterings produced in Section 7.2, I found that when $k \geq 8$ there was always at least one cluster that was

Algorithm 3 Cluster-Medoid Spam Filtering

```
Input: int $k$, Clustering Algorithms $C^*$, ICQM M, Classifier Type $A$, Dataset dTrain, Dataset dTest, double $p$ {$k$ is the number of clusters, $p$ is the percentage of dTrain to sample}
$X \leftarrow$ randomSample(dTrain, $p$) {sample the training data}
bestIcqmScore $\leftarrow$ minValue($M$)
bestClustering $\leftarrow$ null {initialize the best clustering results so far}
for all $C \in C^*$ do
    currentClustering $\leftarrow C(X, k, *)$
    if $M$(currentClustering, $X, *$) $\geq$ bestIcqmScore then
        bestIcqmScore $\leftarrow M$(currentClustering, $X, *$)
        bestClustering $\leftarrow$ currentClustering
    end if
end for{find the best clustering}
medoids $\leftarrow \{\phi\}$
for all $c_i \in$ bestClustering do
    currentMedoid $\leftarrow$ getMedoid($c_i$)
    medoids $\leftarrow$ medoids $\cup$ currentMedoid
    userLabel(currentMedoid)
end for{get the user label for the medoid of each cluster in the best clustering}
$T \leftarrow$ trainClassifier($A$, medoids) {train the classifier using those emails/labels}
result $\leftarrow$ classify(dTest, $T$) {test the classifier}
return result
```

primarily spam and had a spam medoid, and at least one cluster that was primarily ham and had a ham medoid (across thousands of clusterings).

While CMSF leverages clustering to obtain a good set of training email messages, it is possible to do more. Table 7.2 indicates that when the label of one object in a cluster is of a certain type, then there is a good chance that the label for any of the other messages in the cluster is the same type. My second spam filtering method, which I call *Full-Cluster Spam Filtering (FCSF)* (detailed fully in Algorithm 2), uses this concept. FCSF is identical to CMSF, except after obtaining the medoids and their labels, the labels of each medoid is generalized to label its entire cluster, resulting in a fully labeled training set with which the spam filter may be trained. Again, I will show FCSF is highly effective when its M parameter is set to informativeness versus the other ICQMs I present here.

Because clustering PQ (Eq. 3.47) values are probably not perfect for any clusterings, FCSF will always guess the incorrect label for some number of messages in the sample, with a minimum percentage of wrong labels equal to $1 - PQ$. This means the k requirement

Algorithm 4 Full-Cluster Spam Filtering

```
Input: int $k$, Clustering Algorithms $C^*$, ICQM M, Classifier Type $A$, Dataset dTrain, Dataset dTest, double $p$ {$k$ is the number of clusters, $p$ is the percentage of dTrain to sample}
$X \leftarrow$ randomSample(dTrain, $p$) {sample the training data}
bestIcqmScore $\leftarrow$ minValue($M$)
bestClustering $\leftarrow$ null {initialize the best clustering results so far}
for all $C \in C^*$ do
    currentClustering $\leftarrow C(X, k, *)$
    if $M$(currentClustering, $X, *) \geq$ bestIcqmScore then
        bestIcqmScore $\leftarrow M$(currentClustering, $X, *)$
        bestClustering $\leftarrow$ currentClustering
    end if
end for{find the best clustering}
labeledTrainingData $\leftarrow \{\phi\}$
for all $c_i \in$ bestClustering do
    currentMedoid $\leftarrow$ getMedoid($c_i$)
    userLabel(currentMedoid) {get and label the $i$th medoid}
    for all $x_j \in c_i$ do
        $setLabel$($x_j$, getLabel(currentMedoid))
        labeledTrainingData $\leftarrow$ labeledTrainingData $\cup\, x_j$
    end for{use the medoid label to label the rest of its cluster}
end for
$T \leftarrow$ trainClassifier($A$, labeledTrainingData) {train the classifier using those emails/labels}
result $\leftarrow$ classify(dTest, $T$) {test the classifier}
return result
```

FCSF is more stringent than CMSF—more clusters, and therefore more user effort in labeling, is required (as PQ keeps increasing with k) to minimize the mislabeling. A further concern is that a medoid does not share the majority label of its cluster, in which case FCSF will label the entire cluster with incorrect labels. However, I found that this rarely occurs. Of the thousands of clusterings produced by the experiments described in Section 7.2, only a very small number had any cluster medoids that did not match the majority label of the entire cluster, and these tended to be very small clusters. There are ways to reduce this problem even further, but they require more work for the user (labeling of multiple objects per cluster, for example), and I do not further discuss them here.

Finally, I note that both CMSF and FCSF can be classified as active learning spam filters [138]. Active learning is a form of learning in which the learner can selectively query the user for their desired labels on specific objects. Typical active learning spam filters [138] use an iterative process whereby the current true labeled objects plus the unlabeled objects are used to select the next object(s) to request user labels for. We can therefore consider CMSF and FCSF to be a single iteration of this active learning process, whereby the algorithms use 0 true labeled objects plus n unlabeled objects to select k objects (the cluster medoids) to request labels for simultaneously.

7.3.2 Efficiency

In order to make use of algorithms in real applications, it is vital that they are efficient enough to handle the amounts of data they will encounter in reasonable amounts of time. With respect to this, I provide the following analysis. Let n_1 be the number of emails in the training data, n_2 be the number of emails in the test data, m be the number of distinct ngrams in the training data, t be the maximum number of ngrams read from a training email (2500 in my case), and k be the number of clusters used in the clustering algorithm.

As a preprocessing step the emails in the training data are converted to vectors. This requires building a dictionary followed by mapping each training email to a vector. Dictionary construction can be done in $O(tn_1)$ time with a hashtable. Mapping each email to a vector using the dictionary is a simple sequence of hashtable lookups ($O(tn_1)$ total for all emails) followed by length normalized binary weighting ($O(mn_1)$ time). As $m \geq t$, preprocessing requires $O(mn_1)$ time total. The preprocessing step in my experiments was extremely fast, requiring a few seconds per training dataset to complete.

For CMSF, let $O(A_{\text{clust}}(n_1))$ be the time complexity of the most time expensive clustering algorithm in C^* on the preprocessed training data; then all clusterings in CMSF take $O(|C^*|A_{\text{clust}}(n_1))$ time together. Let $O(|C^*|A_{\text{icqm}}(n_1))$ be the time complexity of computing M on every clustering of the preprocessed training data. Selecting the clustering with the best ICQM score requires $O(|C^*|)$ time. Computing all the medoids of the best clustering requires $O(n_1 m)$ time. Obtaining true labels for those medoids requires $O(k)$ time. Let $O(A_{\text{train}}(k))$ be the time complexity of training the classifier, and let $O(A_{\text{classify}}(k))$ be the time complexity of labeling a single email using the trained classifier. Then the time complexity of CMSF is $O(|C^*|A_{\text{clust}}(n_1) + |C^*|A_{\text{icqm}}(C) + |C^*| + n_1 m + k + A_{\text{train}}(k) + n_2 A_{\text{classify}}(k))$. It is reasonable to assume that the selection of the clustering with the best ICQM score, computing of medoids, and assigning of true labels to those medoids are dominated by the other four components, giving CMSF a time complexity of $O(|C^*|A_{\text{clust}}(n_1) + |C^*|A_{\text{icqm}}(C) + A_{\text{train}}(k) + n_2 A_{\text{classify}}(k))$.

The analysis of FCSF is almost identical, except we must map the labels of the medoids on to the clusters, requiring $O(n_1)$ time. Assuming this cost is inconsequential, by using the same assumptions as we did for CMSF we obtain $O(|C^*|A_{\text{clust}}(n_1) + |C^*|A_{\text{icqm}}(C) + A_{\text{train}}(n_1) + n_2 A_{\text{classify}}(n_1))$. Note that n_1 is used in the train and classify components, as FCSF trains/classifies using n_1 messages, not k as CMSF does.

In short, both my methods have a time complexity based on the clustering time, plus the time to evaluate the quality of each clustering using the ICQM, plus the time to train the classifier, plus the actual classification process. CMSF is faster than FCSF, as $k << n_1$, though it may have the same time complexity. As I noted earlier in this thesis, the time complexity of clustering algorithms can be very high. This means we cannot simply ignore the cost of clustering in my methods, even when $n_1 << n_2$. Similarly, ICQMs such as informativeness can be expensive. However, in my experiments I found that the process of training CMSF or FCSF spam filters was very fast, taking much less time than the actual filtering of the test data using the spam filters.

In the following section, I will show that both CMSF and FCSF perform best when used with informativeness as their ICQM, as opposed to some other common ICQMs. I also show that when they use informativeness, both my spam filtering methods are significantly and substantially better than using random labeled email messages to train a spam filter, and that FCSF is especially effective at spam filtering despite its potential drawbacks.

7.4 Spam Filtering Experiment

To demonstrate the performance of CMSF and FCSF, and that they performed best when using informativeness as their ICQM, I used the same data as the previous experiment, the TREC2005 and CEAS2008 email corpora. The representations for their emails was identical to the form detailed in Section 7.2.1.

To create training and test data from these datasets, I constructed ten samples of each, following the method detailed in Section 7.2.4, except a sample rate of 0.01 was used for both corpora. Each sample was treated as training data for a single run of my methods. The test data corresponding to that specific sample was the remainder of the dataset. Note that this setup meant that for each individual test run my methods used absolutely no test data in the training process.

For the parameters required by my methods, I continued to use the five clustering algorithms detailed previously, and the 2 to 50 range of k as well. For the ICQMs required by my methods I used *informativeness* (Eq. 6.13, its base version), as well as the *Davies-Bouldin Index*, the *C-Index*, *B/W*, and *point-wise margin*.

The exact form of the ICQMs besides informativeness was identical to what I used in the synthetic dataset experiments in Section 6.3. For informativeness, I used leave-one-out crossfold validation with a special five-nearest neighbor classifier. To classify each email, this classifier first computed the ten nearest neighbors of the email by Euclidean distance. For each possible cluster, the average Euclidean distance to emails in that cluster from the ten nearest neighbors was computed. Clusters without representation in the ten nearest neighbors were assigned an average Euclidean distance of ∞. The email was then assigned to the cluster which had the smallest average Euclidean distance. I used this classifier because it allowed very fast computation of leave-one-out crossfold validation and with many classes at once.

The classifier used by the spam filter itself (as opposed to that used by informativeness) was a simple logistic regression (LR) method as detailed by Cormack et al. [35], except no hash function was used on the 4-byte-grams. This classifier was selected for two reasons: 1) it is extremely fast, and 2) it has performance competitive with other state-of-the-art spam filters. During the classification process, an unknown 4-byte-gram in the test data (one that was not in the sample, or was stripped from it by my filtering process) was treated as if it did not occur. The p parameter was set to 1 as the size of each training sample was made small enough so as to ensure it could be clustered relatively quickly.

I computed the average one minus the area under the curve percentage ((1-AUC)%) obtained over the 10 samples per (ICQM, dataset, k) tuple for each of my two methods. I refer to these averages as b_k values. I also computed the average (1-AUC)% obtained from training a spam filter on k random email messages and classifying the rest of the emails using that classifier for each (dataset, k) tuple for $k = 2$ to 50. 50 trails were used to compute each of these tuples, and the classifier used was the same logistic regression classifier used by my methods. I refer to these averages as s_k values.

Fig. 7.2 gives plots of $\ln(s_k/b_k)$ on TREC2005 for the CMSF and FCSF when they use the various ICQM parameters. Fig. 7.3 gives similar plots for the CEAS2008 dataset. As

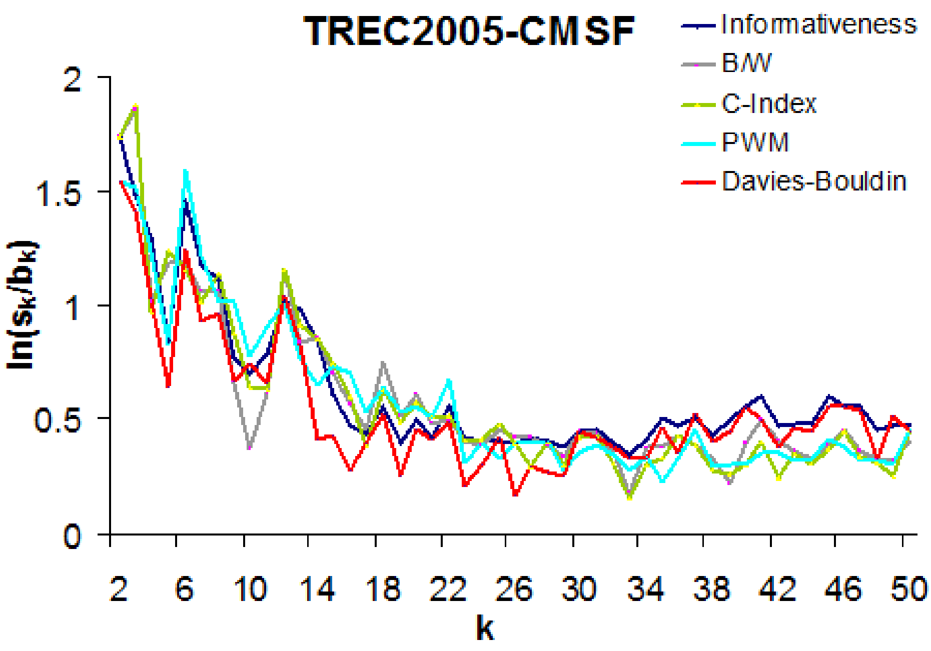
Table 7.3: Wilcoxon signed-rank test results between ICQMs' paired $\ln(s_k/b_k)$ values for TREC2005. $-$ is not significant, $>$ and $<$ indicate a significance at $p = 0.05$, where $>$ indicates the row name had a higher (better) mean, and $<$ indicates the opposite. $<<$ and $>>$ are similar, but with $p = 0.005$.

TREC2005										
	CMSF					**FCSF**				
	Inf.	**B/W**	**C-Index**	**PWM**	**DB**	**Inf.**	**B/W**	**C-Index**	**PWM**	**DB**
Inf.	-	$>>$	$>>$	$>$	$>>$	-	$>>$	$>>$	$>>$	$>>$
B/W	$<<$	-	-	-	-	$<<$	-	-	-	-
C-Index	$<<$	-	-	-	-	$<<$	-	-	-	-
PWM	$<$	-	-	-	-	$<<$	-	-	-	-
DB	$<<$	-	-	-	-	$<<$	-	-	-	-

low (1-AUC)% is better, the higher $\ln(s_k/b_k)$, the better a method was performing. When $\ln(s_k/b_k) > 0$, my methods were outperforming the baseline.

$\ln(s_k/b_k)$ was greater than 0 for all but approximately 10 of the data points in Fig. 7.2 and Fig. 7.3. Averaging over all the results, s_k was 95% worse than b_k for FCSF on TREC2005, 65% worse than b_k for CMSF on TREC2005, 397% worse than b_k for FCSF on CEAS2008, and 81% worse than b_k for CMSF on CEAS2008. The differences between s_k and b_k for fixed ks were significant with $p = 0.05$ in all cases. It is clear that both CMSF and FCSF are substantially superior to the baseline spam filter. This reinforces my suggestion that most clustering algorithms produce clusters that reflect the different kinds of ham and spam present in email datasets (beyond just being mostly ham or mostly spam), and, just as importantly, provides motivation to use my methods in practice. Perhaps the only failing of CMSF and FCSF was on CEAS2008 when k was low; every (ICQMs, method) pairing had poor results in this situation. I believe this is a product of their being much more spam than ham in CEAS2008. As discussed, in a field application of my methods one could pick the minimum k to use based on some understanding of the approximate ratio of ham to spam in the emails, avoiding this problem entirely.

With respect to comparing how the five ICQMs performed in conjunction with my two



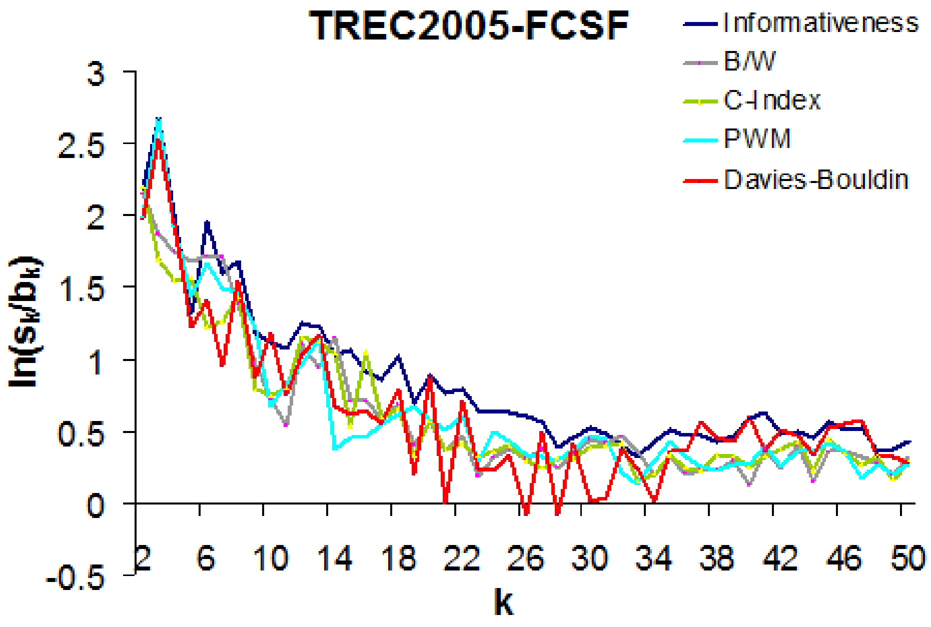
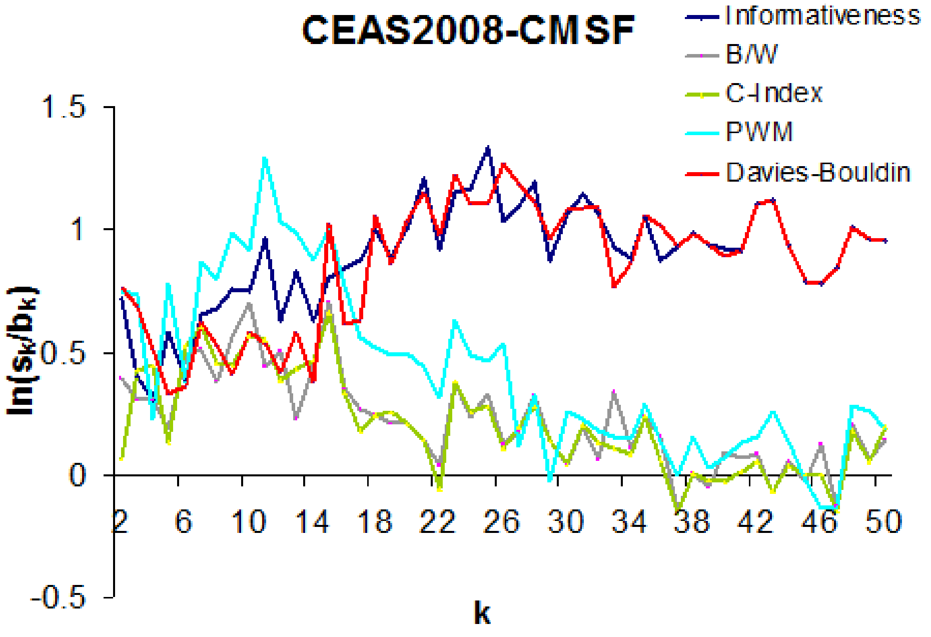


Figure 7.2: CMSF and FCSF results for TREC2005.



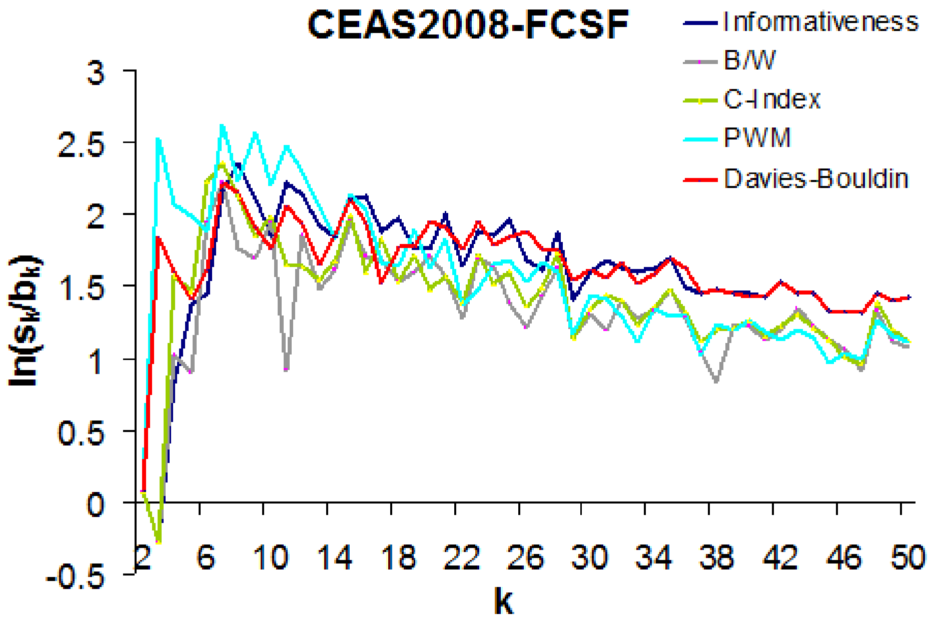


Figure 7.3: CMSF and FCSF results for CEAS2008.

Table 7.4: Wilcoxon signed-rank test results between ICQMs' paired $\ln(s_k/b_k)$ values for CEAS2005. See Table 7.3 for the notation used.

CEAS2008										
	CMSF					**FCSF**				
	Inf.	**B/W**	**C-Index**	**PWM**	**DB**	**Inf.**	**B/W**	**C-Index**	**PWM**	**DB**
Inf.	-	>>	>>	>>	-	-	>>	>>	-	-
B/W	<<	-	-	<<	<<	<<	-	<<	<<	<<
C-Index	<<	-	-	<<	<<	<<	>>	-	<<	<<
PWM	<<	>>	>>	-	-	-	>>	>>	-	-
DB	-	>>	>>	>>	-	-	>>	>>	-	-

spam filterings, one can see a few interesting trends from Fig. 7.2 and 7.3. Point-wise margin produces excellent spam filters when few clusters/user labels are used by either method; this is especially notable for CMSF on CEAS2008. Equally interesting is that its performance degrades relative to the other ICQMs as k increases, being rather poor when $k \geq 20$. B/W and C-Index exhibit uniformly poor across all the results. Davies-Bouldin fairs well on CEAS2008 (Fig. 7.3), but poorly on TREC2005 (Fig. 7.2). On the other hand, we can see that informativeness performed well in all the situations that I tested.

In order to establish the general significance of the results in Fig. 7.2 and Fig. 7.3, for each dataset and method, I computed a Wilcoxon signed-rank test between $\ln(s_k/b_k)$ values for all possible ICQM pairings. Table 7.3 gives the results of these tests for TREC2005, Table 7.4 gives the results for CEAS2008. For TREC2005 we can see that informativeness is significantly better, with $p = 0.005$, than all the other ICQMs when using CMSF or FCSF, except point-wise margin for CMSF; that result is still significant in favor of informativeness, but only with $p = 0.05$. None of the other ICQM pairings had significant differences on TREC2005. This indicates that on TREC2005 informativeness was superior to its four competitors, the four of which were not notably better than each other.

For CEAS2008, Table 7.4 shows that informativeness, Davies-Bouldin, and point-wise margin are significantly better than B/W and C-Index. The only other significant result is that informativeness is better than point-wise margin for CMSF. This suggests that informativeness and Davies-Bouldin were best for CEAS2008, followed by point-wise margin, then B/W and C-Index.

Overall, the significance tests suggest that informativeness was the best ICQM. It was significantly better than all the other ICQMs in at least two of four of the results in Table 7.3 and Table 7.4. Further, no competitor was ever significantly better than it. Davies-Bouldin and point-wise margin were next in line. B/W and C-Index were the worst, neither was significantly better than another ICQM even once, and they were frequently significantly worse than the other three ICQMs.

I now consider two final aspects of my results; 1) Which was superior, CMSF or FCSF?; and 2) How did my methods' results compare against other semi-supervised spam filtering approaches? To answer the first question, I performed more Wilcoxon signed-rank tests, this time between each ICQM's results when using CMSF versus when using FCSF for each dataset; Table 7.5 shows the results of this. From the table, it seems that CMSF was better on TREC2005, and FCSF was better on TREC2008, so one might argue that there is little evidence to conclude one method is better than the other. However, informativeness was better on both datasets with FCSF. With the previous results showing that informativeness is likely the best ICQM to use in either of my methods, we can argue that FCSF is the better of my two spam filtering methods. That being said, the difference between CMSF and FCSF diminishes as k increases. I offer the following explanation for this: as the number of clusters/user labels increase, CMSF obtains a sufficient view of the different kinds of ham/spam. Once this "saturation" point is reached, adding in additional labels, as FCSF does, does not help anymore. It is worth noting that CEAS2008 does not reach my supposed saturation point by the time $k = 50$, although TREC2005 seems to reach

Table 7.5: Wilcoxon signed-rank test results between CMSF and FCSF's paired $\ln(s_k/b_k)$ values for TREC2005 and CEAS2008. The name entry at (row,column) indicates which of the two methods, if any, was significantly better than the other with $p = 0.05$.

	TREC2005	**CEAS2008**
Inf.	FCSF	FCSF
B/W	CMSF	FCSF
C-Index	CMSF	FCSF
PWM	CMSF	FCSF
DB	CMSF	FCSF

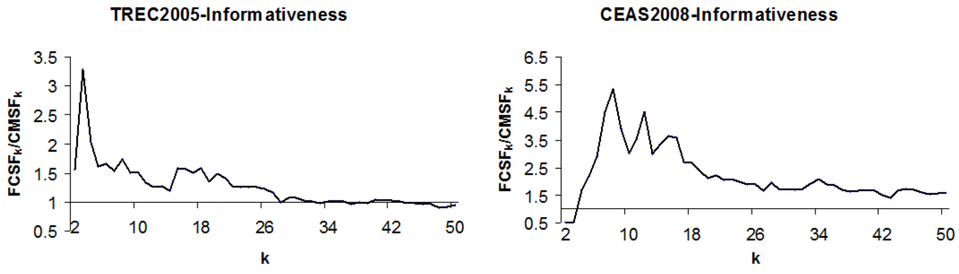


Figure 7.4: Comparison of CMSF and FCSF when using informativeness for TREC2005 and CEAS2008.

it around $k = 30$ (see Fig. 7.4 for an example of this with informativeness). It would be interesting to look at the specific clusterings in order to analyze this behavior, but I do not consider this further here.

The second question from the previous paragraph is the most important of my experiment. While informativeness was the best for both CMSF and FCSF, if it was not outperforming other semi-supervised spam filtering approaches (as opposed to a simple random baseline I used), then it was still not practical to use it in a real spam filter. The results I obtained for FCSF with informativeness are better than those obtained in a recent paper on semi-supervised spam filtering using small numbers of true labels [115]. In that work the authors investigated semi-supervised spam filtering methods using the CEAS2008 and TREC2007 corpora. Their experimental methodology was different from ours, and the results they presented used overall misclassification rate (as opposed to (1-

AUC)%) so definitive comparisons are difficult between their work and mine. However, for CEAS2008 they obtained higher misclassification rates than FCSF with informativeness obtained, on average, with k beyond a small threshold. For example, the average misclassification rate of FCSF with informativeness for CEAS2008, when $k = 32$, was roughly 2%. An average misclassification rate of greater than 3% for the new semi-supervised spam filtering method presented by Mojdeh and Cormack [115] was observed for CEAS2008 when $k = 32$. It is important to note that that method was shown in the paper to be superior to previous semi-supervised spam filtering approaching, so FCSF is at least competitive with the state-of-the-art small sample semi-supervised spam filter.

Overall, I can conclude from my experiment that small sample spam filtering, in my framework, is improved by incorporating unlabeled training data information via clustering. Both my methods are significantly and substantially better than a simple baseline spam filter, justifying their increased time complexity over it. I showed that FCSF with informativeness was the best pairing to use in my methods. Further, that pairing is at least competitive with a state-of-the-art semi-supervised spam filter which uses a small amount of training data.

7.5 Discussion

In this chapter I investigated the use of clustering algorithms and ICQMs in semi-supervised spam filtering. I showed that, given suitable representations for email messages, many clustering algorithms partition email datasets into mostly ham and mostly spam clusters. This result is surprising as clusterings are not always closely related to true labelings, especially when dealing with text datasets. Because a ham/spam split is a natural clustering for an email dataset, clustering can and should be investigated further as a tool for augmenting spam filters.

Along these lines, I presented two spam filtering methods, which illustrate that clustering can be an effective tool for spam filtering. The specific context in these methods can be applied is a setting where there exists a large amount of unlabeled training emails from which we need request only a small number of true labels from the user.

Both my methods exploit clustering and ICQMs. First, my two methods use an ICQM to select the best candidate clustering to use. My first method uses the medoids of clusters in the best clustering to train a spam filter, the second maps the medoid labels onto the entire cluster to obtain a much larger labeled training set. Despite the potential issues I discussed in this chapter, my experiments verify that my two approaches outperform spam filters trained on random labelings.

Informativeness was the best ICQM to use in my two methods among those I tested, with FCSF performing better than CMSF when informativeness was used. The quality of the spam filters produced by this pairing was competitive with state of the art semi-supervised spam filters that used an equivalent number of user labels. This, combined with the fact that my experiment showed that my methods can be applied on large amounts of data, even with slower clustering algorithms and slower ICQMs such as informativeness, proves that informativeness can be of use in real applications.

It is worth noting that spam filtering is classification based, aligning with informativeness' basis well. There are many other real applications that use classification, such as document routing and protein structure prediction. Given my results here, it is highly possible that clustering, combined with informativeness, would be especially effective in other real applications such as those.

Chapter 8

Conclusion

During the time I worked on this thesis a myriad of new clustering research arose. In fact, there was so much that it was not feasible to discuss many of the new works out there in any depth in this thesis. A good amount of newer clustering research is on clustering evaluation, but this has scarcely kept pace with the volume of research on other clustering topics. It seems that for every new clustering evaluation work there are dozens on other clustering topics. This has had the effect of leaving clustering in its precarious poorly understood state. In view of the above the content of this thesis is particularly important.

I began this thesis with a literature review of clustering in general, describing features, similarity/distance measures, and clustering output structures. I covered many notable clustering algorithms and discussed the major points one is most likely to run across in clustering. Beyond providing a review, I aimed at showing just how massive the scope of clustering research is. Despite my thoroughness, I feel the review is only a scratch at the surface of clustering in general. There is a huge amount of literature out there, and more than a few good works can be found outside of the major journals and conferences that have clustering as one of their focuses.

I followed my general clustering literature review with a more narrow review focused on CQMs. I discussed specific CQMs, focusing on those that are most widely used, and extensively analyzed important properties of CQMs in general. While some of the analysis was taken from previous works, a major portion of it was my own research. In particular, I analyzed how numerous CQMs behave with respect to a large body of properties they may have, suggested refinements to previously purposed/analyzed properties, and designed/discussed some new properties. Some of the properties I discussed can be viewed as must-haves for reasonable CQMs, but my analysis also served to highlight many properties that meaningfully differentiate CQMs without necessarily making them better or worse. I believe that such properties aid users in selecting CQMs at least as much, if not more, than must-have properties. In general, more research needs to be dedicated to the study of these and must-have CQM properties. A good aim of such research is the creation of clusterings of CQMs that have high clarity.

The document clustering experiments that followed my CQM discussions showed that the *de facto* standard weighting function in text clustering is inferior to Okapi BM25 weighting. While this is the main result to take away from these experiments, the results nonetheless tie in very well with my CQM research. I showed that there was a disparity between how ECQMs ranked clustering algorithms/clusterings, leading to the question: “Can we say that a certain clustering algorithm is better in general?”. Most authors presenting a new clustering algorithm attempt to answer this question, invariably concluding in favor of the new clustering algorithm they are presenting. However, I argue that it might be better to focus on what makes the new clustering algorithm different from other clustering algorithms using meaningful properties. This analysis could be augmented by demonstrating the new clustering algorithm working in real applications. Showing efficacy in a specific real application cannot prove general superiority, but it is highly sensible if three things hold: 1) the application is meaningful; 2) we agree that we cannot definitively show superiority of clustering algorithms in general; and 3) we agree that situational human assessment is the best way to evaluate a clustering.

Two direct follow-ups to experiments in Chapter 5 that I am currently investigating are the effect of the b parameter in the BM25 weightings, and, more notably, the design of better document clustering similarity/distance measure.

In Chapter 6, I presented the principal work of this thesis: informativeness, my ICQM for estimating the clarity of a clustering. My aim in designing informativeness was to keep it as general an ICQM as possible while also providing good results in practice. I used classification accuracy as its basis, and gave a generalized form. I showed that informativeness behaved favorably with respect to the properties for ICQMs I discussed in Chapter 4. Interestingly, informativeness measures exhibited the best overall performance in my experiments, while not being optimal for every dataset. The non-informativeness ICQMs fluctuated dramatically in the quality of their results by dataset. These facts suggest that I achieved my goal of designing a useful highly general ICQM. It is important to stress that basic informativeness did poorly on no datasets. One often has no idea what clustering structures are actually present in a dataset prior to clustering, necessitating a CQM that will detect as broad a set of clustering structures as possible. Informativeness seems to be such an ICQM.

With respect to future work with informativeness, investigating how it behaves with different classifiers and ECQMs, as well as using it in real applications, is of interest to me. Informativeness might be particularly suited for use in real applications that use classification directly. In Chapter 7, I showed the use of informativeness in one such application: spam filtering. I was able to design highly effective spam filters using minimal user labelings by leveraging clustering and informativeness.

Another interesting avenue to explore is using informativeness in a clustering algorithm. For example, it could be used in a greedy agglomerative clustering algorithm where clusters are progressively merged such that informativeness is maximized for the clustering at each iteration. This process could be repeated until the desired number of clusters was reached. Alternative clustering algorithms using informativeness could be created using other objective function maximization techniques such as those given in Zhao and Karypis [171, 172, 173]. Intuitively, we might expect clusterings generated using informativeness, or classification in general, to be of more use than other clusterings in classification related tasks. Whether this occurs in practice though, remains an open question.

Finally, finding distinct high quality clusterings of a single dataset is a task of some interest in clustering [27, 156, etc.]. It would be worth investigating how informativeness, and other ICQMs, might be used in this task. An example of how this could be done is as follows. We could cluster a dataset many times, then cluster the clusterings to produce clusters representing the various types of groupings possible for the dataset. After this, we could use informativeness (or another ICQM) to select which clustering to use as a representative for each grouping type. This procedure would ensure that the representatives are distinct, and of high quality (with respect to the ICQM used).

As a closing point, I note that given the quality of informativeness' results, one might be curious as to why classification has not been used extensively already in clustering evaluation—it seems like a very sensible thing to try. Given my results, classification can and should be investigated further as a tool for evaluating clusterings.

References

[1] Margareta Ackerman and Shai Ben-David. Measures of clustering quality: A working set of axioms for clustering. In *Neural Information Processing Systems*, pages 121–128, 2008.

[2] Margareta Ackerman and Shai Ben-David. Clusterability: A theoretical study. In *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics*, pages 1–8, 2009.

[3] Margareta Ackerman and Shai Ben-David. Axioms of clustering via quality measures. In journal submission, 2012.

[4] Margareta Ackerman, Shai Ben-David, and David Loker. Characterization of linkage-based clustering. In *Proceedings of the 23rd International Conference on Learning Theory*, 2010.

[5] Charu C. Aggarwal, Joel L. Wolf, Philip S. Yu, Cecilia Procopiuc, and Jong S. Park. Fast algorithms for projected clustering. In *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*, pages 61–72, 1999.

[6] Charu C. Aggarwal and Philip S. Yu. Finding generalized projected clusters in high dimensional spaces. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, pages 70–81, 2000.

[7] Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos, and Prabhakar Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, pages 94–105, 1998.

[8] Bader Aljaber, Nicola Stokes, James Bailey, and Jian Pei. Document clustering of scientific texts using citation contexts. *Information Retrieval*, 13:101–131, 2010.

[9] Enrique Amigó, Julio Gonzalo, Javier Artiles, and Felisa Verdejo. A comparison of extrinsic clustering evaluation metrics based on formal constraints. *Information Retrieval*, 12:461–486, 2009.

[10] Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, and Jörg Sander. OPTICS: Ordering points to identify the clustering structure. *SIGMOD Record*, 28(2):49–60, 1999.

[11] David Arthur and Sergei Vassilvitskii. K-means++: The advantages of careful seeding. In *Proceedings of the 18th ACM-SIAM Symposium on Discrete Algorithms*, pages 1027–1035, 2007.

[12] Arindam Banerjee, Inderjit Dhillon, Joydeep Ghosh, Srujana Merugu, and Dharmendra S. Modha. A generalized maximum entropy approach to Bregman co-clustering and matrix approximation. *Journal of Machine Learning Research*, 8:1919–1986, 2007.

[13] Lei Bao, Sheng Tang, Jintao Li, Yongdong Zhang, and Wei-Ping Ye. Document clustering based on spectral clustering and non-negative matrix factorization. In *Proceedings of the 21st International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems: New Frontiers in Applied Artificial Intelligence*, pages 149–158, 2008.

[14] Daniel Barbará, Yi Li, and Julia Couto. COOLCAT: an entropy-based algorithm for categorical clustering. In *Proceedings of the 11th ACM International Conference on Information and Knowledge Management*, pages 582–589, 2002.

[15] Shariq Bashier and Andreas Rauber. Improving retrievability of patents with cluster-based pseudo-relevance feedback documents selection. In *Proceedings of the 18th ACM International Conference on Information and Knowledge Management*, pages 1863–1866, 2009.

[16] Florian Beil, Martin Ester, and Xiaowei Xu. Frequent term-based text clustering. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 436–442, 2002.

[17] Ron Bekkerman, Ran El-Yaniv, and Andres McCallum. Multi-way distributional clustering via pairwise interactions. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 41–48, 2005.

[18] Asa Ben-Hur, David Horn, Hava T. Siegelmann, and Vladimir Vapnik. Support vector clustering. *Journal of Machine Learning Research*, 2:125–137, 2002.

[19] Kevin Beyer, Jonathan Goldstein, Raghu Ramakrishan, and Uri Shaft. When is "nearest neighbor" meaningful? In *Proceedings of the 7th International Conference on Database Theory*, pages 217–235, 1999.

[20] James C. Bezdek. *Pattern recognition with fuzzy objective function algorithms*. New York: Plenum Press.

[21] Steffen Bickel. Overview. In *European Conference on Machine Learning/Principles and Practice of Knowledge Discovery in Databases Discovery Challenge Workshop*, 2006.

[22] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[23] Christian Böhm, Christos Faloutsos, and Claudia Plant. Outlier-robust clustering using independent components. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pages 185–198, 2008.

[24] Daniel Boley, Maria Gini, Robert Gross, Eui-Hong Han, Kyle Hastings, George Karypis, Vipin Kumar, Bamshad Mobasher, and Jerome Moore. Document categorization and query generation on the World Wide Web using WebACE. *Artificial Intelligence Review*, 11:365–391, 1999.

[25] Roelof Brouwer. Extending the Rand, Adjusted Rand and Jaccard indices to fuzzy partitions. *Journal of Intelligent Information Systems*, 32(3):213–235, 2009.

[26] Ricardo Campello. A fuzzy extension of the rand index and other related indexes for clustering and classification assessment. *Pattern Recognition Letters*, 28(7):833–841, 2007.

[27] Rich Caruana, Mohamed Elhawary, Nam Nguyen, and Casey Smith. Meta clustering. In *Proceedings of the 6th IEEE International Conference on Data Mining*, pages 107–118, 2006.

[28] Ning Chen, An Chen, and Longxiang Zhou. Fuzzy k-prototypes algorithm for clustering mixed numeric and categorical valued data. *Journal of Software*, 23(8):1107–1119, 2001.

[29] Chun-Hung Cheng, Ada W. Fu, and Yi Zhang. Entropy-based subspace clustering for mining numerical data. In *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 84–93, 1999.

[30] Hyuk Cho, Inderjit S. Dhillon, Yuqiang Guan, and Suvrit Sra. Minimum sum-squared residue co-clustering of gene expression data. In *Proceedings of the 4th SIAM International Conference on Data Mining*, 2004.

[31] Pierre Comon. Independent component analysis: A new concept? *Signal Processing*, 36(3):287–314, 1994.

[32] Gordon Cormack and Thomas Lynam. Spam track overview. In *The 14th Text REtrieval Conference*, 2005.

[33] Gordon V. Cormack. Harnessing unlabeled examples through iterative application of dynamic markov modeling. In *European Conference on Machine Learning/Principles and Practice of Knowledge Discovery in Databases Discovery Challenge Workshop*, 2006.

[34] Gordon V. Cormack and Mona Mojdeh. Autonomous personal filtering improves global spam filter performance. In *Proceedings of the 6th Conference on Email and Anti-Spam*, 2009.

[35] Gordon V. Cormack, Mark D. Smucker, and Charles L. A. Clarke. Efficient and effective spam filtering and re-ranking for large web datasets. *Information Retrieval*, pages 1–25, 2011.

[36] Manoranjan Dash, Kiseok Choi, Peter Scheuermann, and Huan Liu. Feature selection for clustering - A filter solution. In *Proceedings of the 2nd IEEE International Conference on Data Mining*, pages 115–122, 2002.

[37] David L. Davies and Donald W. Bouldin. A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1:224–227, 1979.

[38] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41:391–407, 1990.

[39] Arthur Dempster, Nan Laird, and Donald Rubin. Likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society of Britian, Series B*, 39:1–38, 1977.

[40] Inderjit S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 269–274, 2001.

[41] Inderjit S. Dhillon, Yuqiang Guan, and Brian Kulis. Kernel k-means: Spectral clustering and normalized cuts. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 551–556, 2004.

[42] Inderjit S. Dhillon, Subramanyam Mallela, and Rahul Kumar. A divisive information-theoretic feature clustering algorithm for text classification. *Journal of Machine Learning Research*, 3:1265–1287, 2003.

[43] Inderjit S. Dhillon, Subramanyam Mallela, and Dharmendra S. Modha. Information-theoretic co-clustering. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 89–98, 2003.

[44] Inderjit S. Dhillon and Suvrit Sra. Generalized nonnegative matrix approximations with Bregman divergences. In *Neural Information Processing Systems*, pages 283–290, 2005.

[45] Joris D'hondt, Joris Vertommena, Paul Armand Verhaegena, Dirk Cattryssea, and Joost R. Dufloua. Pairwise-adaptive dissimilarity measure for document clustering. *Information Sciences*, 180:2341–2358, 2010.

[46] Lee R. Dice. Measures of the amount of ecologic association between species. *Ecology*, 26:297–302, 1945.

[47] Chris Ding and Xiaofeng He. K-means clustering via principal component analysis. In *Proceedings of the 21st International Conference on Machine Learning*, pages 225–232, 2004.

[48] Chris Ding, Xiaofeng He, Hongyuan Zha, Ming Gu, and Horst Simon. Spectral min-max cut for graph partitioning and data clustering. Technical report, Lawrence Berkeley National Laboratory, 2001.

[49] Chris Ding, Xiaofeng He, Hongyuan Zha, and Horst D. Simon. Adaptive dimension reduction for clustering high dimensional data. In *Proceedings of the 2nd IEEE International Conference on Data Mining*, pages 147–154, 2002.

[50] Chris Ding and Tao Li. Adaptive dimension reduction using discriminant analysis and k-means clustering. In *International Conference on Machine Learning*, pages 521–528, 2007.

[51] Chris Ding, Tao Li, and Wei Peng. On the equivalence between non-negative matrix factorization and probabilistic latent semantic indexing. *Computational Statistic and Data Analysis*, 52(8):3913–3927, 2008.

[52] J. Doak. An evaluation of feature selection methods and their application to computer security. Technical report, University of California, 1992.

[53] John C. Dunn. A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters. *Cybernetics and Systems*, 3:32–57, 1973.

[54] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the 2nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 226–231, 1996.

[55] Yu Feng and Greg Hamerly. PG-means: Learning the number of clusters in data. In *Neural Information Processing Systems*, pages 393–400, 2006.

[56] Charless Fowlkes, Serge Belongie, Fan Chung, and Jitendra Malik. Spectral grouping using the Nyström method. *IEEE Transactions on Pattern Analysis and Machine Learning*, 26:907–916, 2004.

[57] E. Fowlkes and C. Mallows. A method of comparing two hieararchical clusterings. *Journal of the American Statistical Society*, 78(383):553–569, 1983.

[58] Jerome H. Friedman and Jacqueline J. Meulman. Clustering objects on subsets of attributes. *Journal of the Royal Statistical Society of Britian. Series B*, 66:825–849, 2004.

[59] Benjamin C. M. Fung, Ke Wang, and Martin Ester. Hierarchical document clustering using frequent itemsets. In *Proceedings of the 3rd SIAM International Conference on Data Mining*, pages 59–70, 2003.

[60] Eric Gaussier, Cyril Goutte, K. Popat, and F. Chen. A hierarchical model for clustering and categorising documents. In *Advances in Information Retrieval*, pages 121–125, 2002.

[61] Mark Girolami and Ata Kabán. On an equivalence between PLSI and LDA. In *Proceedings of the 26th ACM SIGIR International Conference on Information Retrieval*, pages 433–434, 2003.

[62] Sanjay Goil, Harsha Nagesh, and Alok Choudhary. Efficient and scalable subspace clustering for very large data sets. Technical report, Northwestern University, 1999.

[63] Joshua Goodman and Wen tau Yih. Online discriminative spam filter training. In *Proceedings of the 3rd Conference on Email and Anti-Spam*, 2006.

[64] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. CURE: an efficient clustering algorithm for large databases. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, pages 73–84, 1998.

[65] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. ROCK: A robust clustering algorithm for categorical attributes. In *Proceedings of the 15th International Conference on Data Engineering*, pages 512–521, 1999.

[66] Lars Hagen and Andrew B. Kahng. New spectral methods for ratio cut partitioning and clustering. *IEEE Transactions on CAD*, 11:1074–1085, 1992.

[67] Maria Halkidi, Yannis Batistakis, and Michalis Vazirgiannis. On clustering validation techniques. *Journal of Intelligent Information Systems*, 17:107–145, 2001.

[68] Maria Halkidi, Yannis Batistakis, and Michalis Vazirgiannis. Cluster validity methods: Part I. *SIGMOD Record*, 31(2):40–45, 2002.

[69] Maria Halkidi, Yannis Batistakis, and Michalis Vazirgiannis. Clustering validity checking methods: Part II. *SIGMOD Record*, 31(3):19–27, 2002.

[70] Greg Hamerly and Charles Elkan. Learning the k in k-means. In *Neural Information Processing Systems*, pages 281–288, 2003.

[71] Richard W. Hamming. Error detecting and error correcting codes. *Bell Systems Technical Journal*, 29(2):147–160, 1950.

[72] John Hartigan. Direct clustering of a data matrix. *Journal of the American Statistical Association*, 67:123–132, 1972.

[73] Alexander Hinneburg and Hans-Henning Gabriel. DENCLUE 2.0: Fast clustering based on kernel density estimation. In *Advances in Intelligent Data Analysis*, pages 70–80, 2007.

[74] Alexander Hinneburg and Daniel A. Keim. An efficient approach to clustering in large multimedia databases with noises with noise. In *Proceedings of the 4th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 58–65, 1998.

[75] Akaike Hirotugu. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19:716–723, 1974.

[76] Thomas Hofmann. Probabilistic latent semantic analysis. In *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence*, pages 289–296, 1999.

[77] Thomas Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd ACM SIGIR International Conference on Informationa Retrieval*, pages 50–57, 1999.

[78] Michael E. Houle, Hans-Peter Kriegel, Peer Kröger, Erich Schubert, and Arthur Zimek. Can shared-neighbor distances defeat the curse of dimensionality? In *Proceedings of the 22nd International Conference on Scientific and Statistical Database Management*, pages 217–235, 2010.

[79] Eduardo R. Hruschka, Thiago F. Covoes, Estevam R. Jr. Hruschka, and Nelson F.F. Ebecken. Feature selection for clustering problems: A hybrid algorithm that iterates between k-means and a bayesian filter. pages 405–410, 2005.

[80] Xiaohua Hu, Xiaodan Zhang, Caimei Lu, Eun K. Park, and Xiaohua Zhou. Exploiting Wikipedia as external knowledge for document clustering. In *Proceedings of the ACM*

SIGKDD 15th International Conference on Knowledge Discovery and Data Mining, pages 389–396, 2009.

[81] Zhexue Huang. Extensions to the k-means algorithm for clustering large datasets with categorical values. *Data Mining and Knowledge Discovery*, 2:283–304, 1998.

[82] Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of Classification*, 2:193–218, 1985.

[83] Lawrence Hubert and James Schultz. Quadratic assignment as a general data-analysis strategy. *British Journal of Mathematical and Statistical Psychology*, 29:190–241, 1976.

[84] Mary Inaba, Naoki Katoh, and Hiroshi Imai. Applications of weighted Voronoi diagrams and randomization to variance-based k-clustering. In *Proceedings of the 10th Symposium on Computational Geometry*, pages 332–339, 1994.

[85] Paul Jaccard. Étude comparative de la distribution florale dans une portion des alpes et des jura. *Bulletin de la Socit Vaudoise des Sciences Naturelles*, 37:547–579, 1901.

[86] Anil K. Jain. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31:651–666, 2010.

[87] Anil K. Jain and Richard C. Dubes. *Algorithms for Clustering Data.* Prentice-Hall, Inc., 1988.

[88] Anil K. Jain, M. N. Murty, and Patrick J. Flynn. Data clustering: A review. *ACM Computing Reviews*, 31:264–323, 1999.

[89] Thorsten Joachims. *Making large-scale support vector machine learning practical.* 1998.

[90] Thorsten Joachims. *Learning To Classify Text Using Support Vector Machines: Methods, Theory, and Algorithms.* Kluwer Academic Publishing, 2002.

[91] Leonard Kafuman and Peter Rousseeuw. *Finding groups in data: An introduction to cluster analysis.* Wiley (New York), 1990.

[92] Karin Kailing, Hans-Peter Kriel, and Peer Kröger. Density-connected subspace clustering for high-dimensional data. In *Proceedings of the 4th SIAM International Conference on Data Mining*, pages 246–257, 2004.

[93] Maurice Kendall. A new measure of rank correlation. *Biometrika*, 30(1-2):81–89, 1938.

[94] YongSeog Kim, W. Nick Street, and Filippo Menczer. Feature selection for unsupervised learning via evolutionary search. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 365–369, 2000.

[95] Jon Kleinberg. An impossibility theorem for clustering. In *Neural Information Processing Systems*, pages 446–453, 2002.

[96] Hans-Peter Kriegel, Peer Kröger, and Arthur Zimek. Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *ACM Transactions on Knowledge Discovery from Data*, 3:1–58.

[97] Joseph B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. In *Proceedings of the American Mathematical Society*, volume 7, pages 48–50, 1956.

[98] Harold W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logist Quarterly*, 2:83–97, 1955.

[99] Amit Kumar, Yogish Sabbarwal, and Sandeep Sen. A simple linear time $(1+\epsilon)$-approximation algorithm for k-means clustering in any dimensions. In *Proceedings of the 45th IEEE Symposium on Foundations of Computer Science*, pages 454–452, 2004.

[100] Kenichi Kurihara and Max Welling. Bayesian K-means as a "Maximization-Expectation" algorithm. *Neural Computation*, 21(4):1145–1172, 2009.

[101] Sangeetha Kutty, Richi Nayak, and Yuefeng Li. Utilising semantic tags in XML clustering. In *Focused Retrieval and Evaluation*, pages 416–425, 2010.

[102] Fulu Li and Mo Han Hsieh. An empirical study of clustering behavior of spammers and group-based anti-spam strategies. In *Proceedings of the 3rd Conference on Email and Anti-Spam*, 2006.

[103] Tao Li, Sheng Ma, and Mitsunori Ogihara. Document clustering via adaptive subspace iteration. In *Proceedings of the 27th ACM SIGIR International Conference on Information Retrieval*, pages 218–225, 2004.

[104] Guimei Liu, Jinyan Li, Kelvin Sim, and Limsoon Wong. Distance based subspace clustering with flexible dimension partitioning. In *Proceedings of the 2007 IEEE International Conference on Data Engineering*, pages 1250–1254, 2007.

[105] Huan Liu and Lei Yu. Toward integrating feature selection algorithms for classification and clustering. *IEEE Transactions on Knowledge and Data Engineering*, 17:491–502, 2005.

[106] Stuart Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28:129–137, 1982.

[107] Prasanta C. Mahalanobis. On the generalised distance in statistics. *Proceedings of the National Institude of Sciences of India*, 2(1):49–55, 1936.

[108] Jiri Matousek. On approximate geometric k-clustering. *Discrete and Computational Geometry*, pages 61–84, 2000.

[109] Marina Meilă. Comparing clusterings: an axiomatic view. In *Proceedings of the 22nd Internation Conference on Machine Learning*, pages 577–584, 2005.

[110] Marina Meilă. Comparing clusteringsan information based distance. *Journal of Multivariate Analysis*, 98:873–895, 2007.

[111] Glenn W. Milligan. A monte carlo study of thirty internal criterion measures for cluster analysis. *Psychometrika*, 46:187–199, 1981.

[112] Glenn W. Milligan and Martha C. Cooper. An examination of procedures for determining the number of clusters. *Psychometrika*, 50:159–179, 1985.

[113] Pabitra Mitra, C. A. Murthy, and Sankar K. Pal. Unsupervised feature selection using feature similarity. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(3):301–312, 2002.

[114] Mona Mojdeh and Gordon V. Cormack. Semi-supervised spam filtering: Does it work? In *Proceedings of the 31th ACM SIGIR International Conference on Information Retrieval*, pages 745–746, 2008.

[115] Mona Mojdeh and Gordon V. Cormack. Semi-supervised spam filtering using aggressive consistency learning. In *Proceedings of the 33th ACM SIGIR International Conference on Information Retrieval*, pages 751–752, 2010.

[116] Sandrine Mouysset, Joseph Noailles, and Daniel Ruiz. Using a global parameter for gaussian affinity matrices in spectral clustering. In *High Performance Computing for Computational Science*, pages 378–390. 2008.

[117] Patrenahalli M. Nardendra and Keinosuke Funkunaga. A brand and bound algorithm for feature subset selection. *IEEE Transactions on Computer*, 26(9):917–922, 1977.

[118] Radford M. Neal and Geoffery E. Hinton. A view of the EM algorithm that justifies incremental, sparse and other variants. Technical report, University of Toronto, 1993.

[119] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *Neural Information Processing Systems*, pages 849–856, 2001.

[120] Alessandro Di Nuovo and Vincenzo Catania. On external measures for validation of fuzzy partitions. In *Foundations of Fuzzy Logic and Soft Computing*, pages 491–501, 2007.

[121] Carlos Ordonez and Edward Omiecinski. FREM: fast and robust EM clustering for large data sets. In *Proceedings of the 11th ACM International Conference on Information and Knowledge Management*, pages 590–599, 2002.

[122] Malay K. Pakhira, Sanghamitra Bandyopadhyay, and Ujjwal Maulik. Validty index for crisp and fuzzy clusters. *Pattern Recognition Letters*, 37:487–501, 2004.

[123] Lance Parsons, Ehtesham Haque, and Huan Liu. Subspace clustering for high dimensional data: A review. *KDD Explorations*, 6:90–105, 2004.

[124] Karl Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2:559–572, 1901.

[125] Dan Pelleg and Andrew W. Moore. X-means: Extending k-means with an efficient estimation of the number of clusters. In *Proceedings of the 17th International Conference on Machine Learning*, 2000.

[126] Martin F. Porter. An algorithm for suffix stripping. *Program*, 14:130–137, 1980.

[127] Henri Ralambondrainy. A conceptual version of the k-means algorithm. *Pattern Recognition Letters*, pages 1147–1157, 1995.

[128] Willam M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66:846–850, 1971.

[129] Stephen E. Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. Okapi at TREC-3. In *TREC '94: The Third Text REtrieval Conference*, 1994.

[130] Joseph Lee Rodger and W. Alan Nicewander. Thirteen ways to look at the correlation coefficient. *The American Statistican*, 42(1):59–66, 1988.

[131] Andrew Roseberg and Julia Hirschberg. V-measure: A conditional entropy-based external cluster evaluation measure. In *the 2007 Joint Conference on Empirical Meth-*

ods in Natural Language Processing and Computational Natural Language Learning, pages 410–420, 2007.

[132] Volker Roth and Tilman Lange. Feature selection in clustering problems. In *Neural Information Processing Systems*, 2004.

[133] Peter J. Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987.

[134] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.

[135] Jörg Sander, Martin Ester, Hans-Peter Kriegel, and Xiaowei Xu. Density-based clustering in spatial databases: The algorithm GDBSCAN and its applications. *Data Mining and Knowledge Discovery*, pages 169–194, 1998.

[136] Bernhard Schölkopf, Alexander J. Smola, and Klaus-Robert Müller. Kernel principal component analysis. In *Proceedings of the 7th International Conference on Artificial Neural Networks*, pages 583–588, 1997.

[137] Gideon E. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6:461–464, 1978.

[138] D Sculley. Online active learning methods for fast label-efficient spam filtering. 2007.

[139] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[140] Noam Slonim and Naftali Tishby. Document clustering using word clusters via the information bottleneck method. In *Proceedings of the 23rd ACM SIGIR International Conference on Informationa Retrieval*, pages 208–215, 2000.

[141] Robert R. Snokal and P. H. Sneath. *Numerical Taxonomy*, pages 230–234. W. H. Freeman and Company, San Francisco, 1973.

[142] Jungsuk Song, Masashi Eto, Hyung C. Kim, Daisuke Inoue, and Koji Nakao. A heuristic-based feature selection method for clustering spam emails. In *Neural Information Processing. Theory and Algorithms*, pages 290–297, 2010.

[143] Wei Song and Soon C. Park. A novel document clustering model based on latent semantic analysis. In *Proceedings of the 2007 International Conference on Semantics, Knowledge and Grid*, pages 539–542, 2007.

[144] Charles Spearman. The proof and measurement of association between two things. *The American Journal of Psychology*, 15(1):72–101.

[145] Michael Steinbach, George Karypis, and Vipin Kumar. A comparison of document clustering techniques. In *KDD Text Mining Workshop*, 2000.

[146] Alexander Strehl and Joydeep Ghosh. Cluster ensembles – a knowledge reuse framework for combining multipe partitions. *Journal of Machine Learning Research*, 3:583–617, 2002.

[147] Robert Tibshirani, Guenther Walther, and Trevor Hastie. Estimating the number of clusters in a dataset via the gap statistic. *Journal of the Royal Statistical Society of Britian*, 32:411–423, 2001.

[148] Naftali Tishby, Fernando C. Pereira, and William Bialek. The information bottleneck method. In *Proceedings of the 1999 Allerton Conference on Communication, Control, and Computing*, pages 368–377, 1999.

[149] Hamed Valizadegan and Rong Jin. Generalized maximum margin clustering and unsupervised kernel learning. In *Neural Information Processing Systems*, pages 1417–1424, 2007.

[150] Cornelis J. van Rijsbergen. *Information Retrieval.* Butterworth, 2nd edition, 1979.

[151] Nguyen X. Vinh, Julien Epps, and James Bailey. Information theoretic measures for clusterings comparison: Is a correction for chance necessary? In *Proceedings of the 26th International Conference on Machine Learning*, 2009.

[152] Ulrike von Luxberg. A tutorial on spectral clustering. *Statistics and Computing*, 17:395–416, 2007.

[153] Christopher M. De Vries and Shlomo Geva. Document Clustering with K-tree. In *INEX*, pages 420–431, 2008.

[154] Stanley Wasserman and Katherine Faust. *Social Network Analysis: Methods and Applications.* Cambridge University Press, 1994.

[155] John S. Whissell and Charles L. A. Clarke. Improving document clustering using Okapi BM25 feature weighting. *Information Retrieval*, 14:466–487, 2011.

[156] John S. Whissell, Charles L. A. Clarke, and Azin Ashkan. Clustering web queries. In *Proceedings of the 18th ACM International Conference on Information and Knowledge Management*, pages 899–908, 2009.

[157] John Wilbur and Won Kim. The ineffectiveness of within-document term frequency in text classification. *Information Retrieval*, 12(5):509–525, 2009.

[158] Kyoung-Gu Woo, Jeong-Hoon Lee, Myoung-Ho Kim, and Yoon-Joon Lee. FINDIT: A fast and intelligent subspace clustering algorithm using dimension voting. *Information and Software Technology*, 46(4):255–271, 2004.

[159] Rui Wu, Jianhua Huang, Xianglong Tang, and Jiafeng Liu. A text image segmentation method based on spectral clustering. *Computer and Information Science*, 1(4):9–15, 2008.

[160] Xuanli L. Xie and Gerardo A. Beni. Validity measure for fuzzy clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 3:841–846, 1991.

[161] Linli Xu, James Neufeld, Bryce Larson, and Dale Schuurmans. Maximum margin clustering. In *Neural Information Processing Systems*, pages 1537–1544, 2005.

[162] Wei Xu, Xin Liu, and Yihong Gong. Document clustering based on non-negative matrix factorization. In *Proceedings of the 26th ACM SIGIR International Conference on Information Retrieval*, pages 267–273, 2003.

[163] Gui-Rong Xue, Wenyuan Dai, Qiang Yang, and Yong Yu. Topic-bridged PLSA for cross-domain text classification. In *Proceedings of the 31st ACM SIGIR International Conference on Information Retrieval*, pages 627–634, 2008.

[164] Mingjin Yan and Keying Ye. Determining the number of clusters using the weighted gap statistic. *Biometrics*, 63:1031–1037, 2007.

[165] Yu-Jiu Yang and Bao-Gang Hu. Pairwise constraints-guided non-negative matrix factorization for document clustering. In *Proceedings of the International Conference on Web Intelligence*, pages 250–256, 2007.

[166] Reza Bosagh Zadeh and Shai Ben-David. A uniqueness theorem for clustering. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*, 2009.

[167] Bin Zhang, Meichun Hsu, and Umeshwar Dayal. K-Harmonic means - a data clustering algorithm. Technical report, Hewlett Packard, 1999.

[168] Kai Zhang, Ivor W. Tsang, and James T. Kwok. Maximum margin clustering made practical. *IEEE Transactions on Neural Networks*, 20:583–596, 2009.

[169] Bin Zhao, Fei Wang, and Changshui Zhang. Efficient maximum margin clustering via cutting plane algorithm. In *Proceedings of the 8th SIAM International Conference on Data Mining*, pages 751–762, 2008.

[170] Bin Zhao, Fei Wang, and Changshui Zhang. Efficient multiclass maximum margin clustering. In *Proceedings of the 25th International Conference on Machine Learning*, pages 1248–1255, 2008.

[171] Ying Zhao and George Karypis. Criterion functions for document clustering: Experiments and analysis. Technical Report 01-40, University of Minnesota, Department of Computer Science / Army HPC Research Center, 2001.

[172] Ying Zhao and George Karypis. Evaluation of hierarchical clustering algorithms for document datasets. *Data Mining and Knowledge Discovery*, pages 515–524, 2002.

[173] Ying Zhao and George Karypis. Empirical and theoretical comparisons of selected criterion functions for document clustering. *Machine Learning*, 55:311–331, 2004.