Chapter 4

A SURVEY OF TEXT CLUSTERING ALGORITHMS

Charu C. Aggarwal
IBM T. J. Watson Research Center
Yorktown Heights, NY
charu@us.ibm.com

ChengXiang Zhai
University of Illinois at Urbana-Champaign
Urbana, IL
czhai@cs.uiuc.edu

Abstract Clustering is a widely studied data mining problem in the text domains. The problem finds numerous applications in customer segmentation, classification, collaborative filtering, visualization, document organization, and indexing. In this chapter, we will provide a detailed survey of the problem of text clustering. We will study the key challenges of the clustering problem, as it applies to the text domain. We will discuss the key methods used for text clustering, and their relative advantages. We will also discuss a number of recent advances in the area in the context of social network and linked data.

Keywords: Text Clustering

1. Introduction

The problem of clustering has been studied widely in the database and statistics literature in the context of a wide variety of data mining tasks [50, 54]. The clustering problem is defined to be that of finding groups of similar objects in the data. The similarity between the objects is measured with the use of a similarity function. The problem of clustering can be very useful in the text domain, where the objects to be clusters can be of different granularities such as documents, paragraphs, sentences or terms. Clustering is especially useful for organizing documents to improve retrieval and support browsing [11, 26].

The study of the clustering problem precedes its applicability to the text domain. Traditional methods for clustering have generally focussed on the case of quantitative data [44, 71, 50, 54, 108], in which the attributes of the data are numeric. The problem has also been studied for the case of categorical data [10, 41, 43], in which the attributes may take on nominal values. A broad overview of clustering (as it relates to generic numerical and categorical data) may be found in [50, 54]. A number of implementations of common text clustering algorithms, as applied to text data, may be found in several toolkits such as *Lemur* [114] and *BOW* toolkit in [64]. The problem of clustering finds applicability for a number of tasks:

- **Document Organization and Browsing:** The hierarchical organization of documents into coherent categories can be very useful for systematic browsing of the document collection. A classical example of this is the *Scatter/Gather* method [25], which provides a systematic browsing technique with the use of clustered organization of the document collection.

- **Corpus Summarization:** Clustering techniques provide a coherent summary of the collection in the form of *cluster-digests* [83] or *word-clusters* [17, 18], which can be used in order to provide summary insights into the overall content of the underlying corpus. Variants of such methods, especially sentence clustering, can also be used for document summarization, a topic, discussed in detail in Chapter 3. The problem of clustering is also closely related to that of dimensionality reduction and topic modeling. Such dimensionality reduction methods are all different ways of summarizing a corpus of documents, and are covered in Chapter 5.

- **Document Classification:** While clustering is inherently an unsupervised learning method, it can be leveraged in order to improve the quality of the results in its supervised variant. In particular, word-clusters [17, 18] and co-training methods [72] can be used in order to improve the classification accuracy of supervised applications with the use of clustering techniques.

We note that many classes of algorithms such as the k-means algorithm, or hierarchical algorithms are general-purpose methods, which can be extended to any kind of data, including text data. A text document can be represented either in the form of binary data, when we use the presence or absence of a word in the document in order to create a binary vector. In such cases, it is possible to directly use a variety of categorical data clustering algorithms [10, 41, 43] on the binary representation. A more enhanced representation would include refined weighting methods based on the frequencies of the individual words in the document as well as frequencies of words in an entire collection (e.g., TF-IDF weighting [82]). Quantitative data clustering algorithms [44, 71, 108] can be used in conjunction with these frequencies in order to determine the most relevant groups of objects in the data.

However, such naive techniques do not typically work well for clustering text data. This is because text data has a number of unique properties which necessitate the design of specialized algorithms for the task. The distinguishing characteristics of the text representation are as follows:

- The dimensionality of the text representation is very large, but the underlying data is sparse. In other words, the lexicon from which the documents are drawn may be of the order of 10^5, but a given document may contain only a few hundred words. This problem is even more serious when the documents to be clustered are very short (e.g., when clustering sentences or tweets).

- While the lexicon of a given corpus of documents may be large, the words are typically correlated with one another. This means that the number of concepts (or principal components) in the data is much smaller than the feature space. This necessitates the careful design of algorithms which can account for word correlations in the clustering process.

- The number of words (or non-zero entries) in the different documents may vary widely. Therefore, it is important to normalize the document representations appropriately during the clustering task.

The sparse and high dimensional representation of the different documents necessitate the design of text-specific algorithms for document representation and processing, a topic heavily studied in the information retrieval literature where many techniques have been proposed to optimize document representation for improving the accuracy of matching a document with a query [82, 13]. Most of these techniques can also be used to improve document representation for clustering.

In order to enable an effective clustering process, the word frequencies need to be normalized in terms of their relative frequency of presence in the document and over the entire collection. In general, a common representation used for text processing is the *vector-space based* TF-IDF representation [81]. In the TF-IDF representation, the term frequency for each word is normalized by the *inverse document frequency*, or IDF. The inverse document frequency normalization reduces the weight of terms which occur more frequently in the collection. This reduces the importance of common terms in the collection, ensuring that the matching of documents be more influenced by that of more discriminative words which have relatively low frequencies in the collection. In addition, a sub-linear transformation function is often applied to the term-frequencies in order to avoid the undesirable dominating effect of any single term that might be very frequent in a document. The work on document-normalization is itself a vast area of research, and a variety of other techniques which discuss different normalization methods may be found in [86, 82].

Text clustering algorithms are divided into a wide variety of different types such as agglomerative clustering algorithms, partitioning algorithms, and standard parametric modeling based methods such as the EM-algorithm. Furthermore, text representations may also be treated as strings (rather than bags of words). These different representations necessitate the design of different classes of clustering algorithms. Different clustering algorithms have different tradeoffs in terms of effectiveness and efficiency. An experimental comparison of different clustering algorithms may be found in [90, 111]. In this chapter we will discuss a wide variety of algorithms which are commonly used for text clustering. We will also discuss text clustering algorithms for related scenarios such as dynamic data, network-based text data and semi-supervised scenarios.

This chapter is organized as follows. In section 2, we will present feature selection and transformation methods for text clustering. Section 3 describes a number of common algorithms which are used for distance-based clustering of text documents. Section 4 contains the description of methods for clustering with the use of word patterns and phrases. Methods for clustering text streams are described in section 5. Section 6 describes methods for probabilistic clustering of text data. Section 7 contains a description of methods for clustering text which naturally occurs in the context of social or web-based networks. Section 8 discusses methods for semi-supervised clustering. Section 9 presents the conclusions and summary.

2. Feature Selection and Transformation Methods for Text Clustering

The quality of any data mining method such as classification and clustering is highly dependent on the noisiness of the features that are used for the clustering process. For example, commonly used words such as *"the"*, may not be very useful in improving the clustering quality. Therefore, it is critical to select the features effectively, so that the noisy words in the corpus are removed before the clustering. In addition to feature *selection*, a number of feature *transformation* methods such as Latent Semantic Indexing (LSI), Probabilistic Latent Semantic Analysis (PLSA), and Non-negative Matrix Factorization (NMF) are available to improve the quality of the document representation and make it more amenable to clustering. In these techniques (often called dimension reduction), the correlations among the words in the lexicon are leveraged in order to create features, which correspond to the concepts or principal components in the data. In this section, we will discuss both classes of methods. A more in-depth discussion of dimension reduction can be found in Chapter 5.

2.1 Feature Selection Methods

Feature selection is more common and easy to apply in the problem of text categorization [99] in which supervision is available for the feature selection process. However, a number of simple unsupervised methods can also be used for feature selection in text clustering. Some examples of such methods are discussed below.

2.1.1 Document Frequency-based Selection. The simplest possible method for feature selection in document clustering is that of the use of *document frequency* to filter out irrelevant features. While the use of inverse document frequencies reduces the importance of such words, this may not alone be sufficient to reduce the noise effects of very frequent words. In other words, words which are too frequent in the corpus can be removed because they are typically common words such as "a", "an", "the", or "of" which are not discriminative from a clustering perspective. Such words are also referred to as *stop words*. A variety of methods are commonly available in the literature [76] for stop-word removal. Typically commonly available stop word lists of about 300 to 400 words are used for the retrieval process. In addition, words which occur extremely infrequently can also be removed from the collection. This is because such words do not add anything to the similarity computations which are used in most clustering methods. In some cases, such words may be misspellings or typographical errors in documents. Noisy text collections which are derived from the web, blogs or social networks are more likely to contain such terms. We note that some lines of research define document frequency based selection purely on the basis of very infrequent terms, because these terms contribute the least to the similarity calculations. However, it should be emphasized that very frequent words should also be removed, especially if they are not discriminative between clusters. Note that the TF-IDF weighting method can also naturally filter out very common words in a "soft" way. Clearly, the standard set of stop words provide a valid set of words to prune. Nevertheless, we would like a way of quantifying the importance of a term directly to the clustering process, which is essential for more aggressive pruning. We will discuss a number of such methods below.

2.1.2 Term Strength. A much more aggressive technique for stop-word removal is proposed in [94]. The core idea of this approach is to extend techniques which are used in supervised learning to the unsupervised case. The term strength is essentially used to measure how informative a word is for identifying two related documents. For example, for two related documents x and y, the term strength $s(t)$ of term t is defined in terms of the following probability:

$$s(t) = P(t \in y | t \in x) \tag{4.1}$$

Clearly, the main issue is how one might define the document x and y as related. One possibility is to use manual (or user) feedback to define when a pair of documents are related. This is essentially equivalent to utilizing supervision in the feature selection process, and may be practical in situations in which predefined categories of documents are available. On the other hand, it is not practical to manually create related pairs in large collections in a comprehensive way. It is therefore desirable to use an automated and purely unsupervised way to define the concept of when a pair of documents is related. It has been shown in [94] that it is possible to use automated similarity functions such as the cosine function [81] to define the relatedness of document pairs. A pair of documents are defined to be related if their cosine similarity is above a user-defined threshold. In such cases, the term strength $s(t)$ can be defined by randomly sampling a number of pairs of such related documents as follows:

$$s(t) = \frac{\text{Number of pairs in which } t \text{ occurs in both}}{\text{Number of pairs in which } t \text{ occurs in the first of the pair}} \tag{4.2}$$

Here, the first document of the pair may simply be picked randomly. In order to prune features, the term strength may be compared to the expected strength of a term which is randomly distributed in the training documents with the same frequency. If the term strength of t is not at least two standard deviations greater than that of the random word, then it is removed from the collection.

One advantage of this approach is that it requires no initial supervision or training data for the feature selection, which is a key requirement in the unsupervised scenario. Of course, the approach can also be used for feature selection in either supervised clustering [4] or categorization [100], when such training data is indeed available. One observation about this approach to feature selection is that it is particularly suited to similarity-based clustering because the discriminative nature of the underlying features is defined on the basis of similarities in the documents themselves.

2.1.3 Entropy-based Ranking. The entropy-based ranking approach was proposed in [27]. In this case, the quality of the term is measured by the entropy reduction when it is removed. Here the entropy $E(t)$ of the term t in a collection of n documents is defined as follows:

$$E(t) = -\sum_{i=1}^{n}\sum_{j=1}^{n}(S_{ij} \cdot \log(S_{ij}) + (1 - S_{ij}) \cdot \log(1 - S_{ij})) \tag{4.3}$$

Here $S_{ij} \in (0, 1)$ is the similarity between the ith and jth document in the collection, after the term t is removed, and is defined as follows:

$$S_{ij} = 2^{-\frac{dist(i,j)}{\overline{dist}}} \tag{4.4}$$

Here $dist(i, j)$ is the distance between the terms i and j after the term t is removed, and $\overline{dist}$ is the average distance between the documents after the term t is removed. We note that the computation of $E(t)$ for each term t requires $O(n^2)$ operations. This is impractical for a very large corpus containing many terms. It has been shown in [27] how this method may be made much more efficient with the use of sampling methods.

2.1.4 Term Contribution. The concept of term contribution [62] is based on the fact that the results of text clustering are highly dependent on document similarity. Therefore, the contribution of a term can be viewed as its contribution to document similarity. For example, in the case of dot-product based similarity, the similarity between two documents is defined as the dot product of their normalized frequencies. Therefore, the contribution of a term of the similarity of two documents is the product of their normalized frequencies in the two documents. This needs to be summed over all pairs of documents in order to determine the term contribution. As in the previous case, this method requires $O(n^2)$ time for each term, and therefore sampling methods may be required to speed up the contribution. A major criticism of this method is that it tends to favor highly frequent words without regard to the specific discriminative power within a clustering process.

In most of these methods, the optimization of term selection is based on some pre-assumed similarity function (e.g., cosine). While this strategy makes these methods unsupervised, there is a concern that the term selection might be biased due to the potential bias of the assumed similarity function. That is, if a different similarity function is assumed, we may end up having different results for term selection. Thus the choice of an appropriate similarity function may be important for these methods.

2.2 LSI-based Methods

In feature selection, we attempt to explicitly select out features from the original data set. Feature transformation is a different method in which the new features are defined as a functional representation of the features in the original data set. The most common class of methods is that of dimensionality reduction [53] in which the documents are transformed to a new feature space of smaller dimensionality in which the features are typically a linear combination of the features in the original data. Methods such as Latent Semantic Indexing (LSI) [28] are based on this common principle. The overall effect is to remove a lot of dimensions in the data which are noisy for similarity based applications such as clustering. The removal of such dimensions also helps magnify the semantic effects in the underlying data.

Since LSI is closely related to problem of *Principal Component Analysis (PCA)* or *Singular Value Decomposition (SVD)*, we will first discuss this method, and its relationship to LSI. For a d-dimensional data set, PCA constructs the symmetric $d \times d$ covariance matrix C of the data, in which the (i, j)th entry is the covariance between dimensions i and j. This matrix is positive semi-definite, and can be diagonalized as follows:

$$C = P \cdot D \cdot P^T \tag{4.5}$$

Here P is a matrix whose columns contain the orthonormal eigenvectors of C and D is a diagonal matrix containing the corresponding eigenvalues. We note that the eigenvectors represent a new orthonormal basis system along which the data can be represented. In this context, the eigenvalues correspond to the variance when the data is projected along this basis system. This basis system is also one in which the second order covariances of the data are removed, and most of variance in the data is captured by preserving the eigenvectors with the largest eigenvalues. Therefore, in order to reduce the dimensionality of the data, a common approach is to represent the data in this new basis system, which is further truncated by ignoring those eigenvectors for which the corresponding eigenvalues are small. This is because the variances along those dimensions are small, and the relative behavior of the data points is not significantly affected by removing them from consideration. In fact, it can be shown that the Euclidian distances between data points are not significantly affected by this transformation and corresponding truncation. The method of PCA is commonly used for similarity search in database retrieval applications.

LSI is quite similar to PCA, except that we use an approximation of the covariance matrix C which is quite appropriate for the sparse and high-dimensional nature of text data. Specifically, let A be the $n \times d$ term-document matrix in which the (i, j)th entry is the normalized frequency for term j in document i. Then, $A^T \cdot A$ is a $d \times d$ matrix which is close (scaled) approximation of the covariance matrix, in which the means have not been subtracted out. In other words, the value of $A^T \cdot A$ would be the same as a scaled version (by factor n) of the covariance matrix, if the data is mean-centered. While text-representations are not mean-centered, the sparsity of text ensures that the use of $A^T \cdot A$ is quite a good approximation of the (scaled) covariances. As in the case of numerical data, we use the eigenvectors of $A^T \cdot A$ with the largest variance in order to represent the text. In typical collections, only about 300 to 400 eigenvectors are required for the representation. One excellent characteristic of LSI [28] is that the truncation of the dimensions removes the noise effects of synonymy and polysemy, and the similarity computations are more closely affected by the semantic concepts in the data. This is particularly useful for a semantic application such as text clustering. However, if finer granularity clustering is needed, such low-dimensional space representation of text may not be sufficiently discriminative; in information retrieval, this problem is often solved by mixing the low-dimensional representation with the original high-dimensional word-based representation (see, e.g., [105]).

A similar technique to LSI, but based on probabilistic modeling is Probabilistic Latent Semantic Analysis (PLSA) [49]. The similarity and equivalence of PLSA and LSI are discussed in [49].

2.2.1 Concept Decomposition using Clustering. One interesting observation is that while feature transformation is often used as a pre-processing technique for clustering, the clustering itself can be used for a novel dimensionality reduction technique known as *concept decomposition* [2, 29]. This of course leads to the issue of circularity in the use of this technique for clustering, especially if clustering is required in order to perform the dimensionality reduction. Nevertheless, it is still possible to use this technique effectively for pre-processing with the use of two separate phases of clustering.

The technique of concept decomposition uses any standard clustering technique [2, 29] on the original representation of the documents. The frequent terms in the centroids of these clusters are used as *basis vectors* which are almost orthogonal to one another. The documents can then be represented in a much more concise way in terms of these basis vectors. We note that this condensed conceptual representation allows for enhanced clustering as well as classification. Therefore, a second phase of clustering can be applied on this reduced representation in order to cluster the documents much more effectively. Such a method has also been tested in [87] by using word-clusters in order to represent documents. We will describe this method in more detail later in this chapter.

2.3 Non-negative Matrix Factorization

The non-negative matrix factorization (NMF) technique is a latent-space method, and is particularly suitable to clustering [97]. As in the case of LSI, the NMF scheme represents the documents in a new axis-system which is based on an analysis of the term-document matrix. However, the NMF method has a number of critical differences from the LSI scheme from a conceptual point of view. In particular, the NMF scheme is a feature transformation method which is particularly suited to clustering. The main conceptual characteristics of the NMF scheme, which are very different from LSI are as follows:

- In LSI, the new basis system consists of a set of orthonormal vectors. This is not the case for NMF.

- In NMF, the vectors in the basis system directly correspond to cluster topics. Therefore, the cluster membership for a document may be determined by examining the largest component of the document along any of the vectors. The coordinate of any document along a vector is always non-negative. The expression of each document as an additive combination of the underlying semantics makes a lot of sense from an intuitive perspective. Therefore, the NMF transformation is particularly suited to clustering, and it also provides an intuitive understanding of the basis system in terms of the clusters.

Let A be the $n \times d$ term document matrix. Let us assume that we wish to create k clusters from the underlying document corpus. Then, the non-negative matrix factorization method attempts to determine the matrices U and V which minimize the following objective function:

$$J = (1/2) \cdot ||A - U \cdot V^T|| \tag{4.6}$$

Here $|| \cdot ||$ represents the sum of the squares of all the elements in the matrix, U is an $n \times k$ non-negative matrix, and V is a $m \times k$ non-negative matrix. We note that the columns of V provide the k basis vectors which correspond to the k different clusters.

What is the significance of the above optimization problem? Note that by minimizing J, we are attempting to factorize A approximately as:

$$A \approx U \cdot V^T \tag{4.7}$$

For each *row* a of A (document vector), we can rewrite the above equation as:

$$a \approx u \cdot V^T \tag{4.8}$$

Here u is the corresponding row of U. Therefore, the document vector a can be rewritten as an approximate linear (non-negative) combination of the basis vector which corresponds to the k columns of V^T. If the value of k is relatively small compared to the corpus, this can only be done if the column vectors of V^T discover the latent structure in the data. Furthermore, the non-negativity of the matrices U and V ensures that the documents are expressed as a non-negative combination of the key concepts (or clustered) regions in the term-based feature space.

Next, we will discuss how the optimization problem for J above is actually solved. The squared norm of any matrix Q can be expressed as the trace of the matrix $Q \cdot Q^T$. Therefore, we can express the objective function above as follows:

$$\begin{aligned} J &= (1/2) \cdot tr((A - U \cdot V^T) \cdot (A - U \cdot V^T)^T) \\ &= (1/2) \cdot tr(A \cdot A^T) - tr(A \cdot U \cdot V^T) + (1/2) \cdot tr(U \cdot V^T \cdot V \cdot U^T) \end{aligned}$$

Thus, we have an optimization problem with respect to the matrices $U = [u_{ij}]$ and $V = [v_{ij}]$, the entries u_{ij} and v_{ij} of which are the variables with respect to which we need to optimize this problem. In addition, since the matrices are non-negative, we have the constraints that $u_{ij} \geq 0$ and $v_{ij} \geq 0$. This is a typical constrained non-linear optimization problem, and can be solved using the Lagrange method. Let $\alpha = [\alpha_{ij}]$ and $\beta = [\beta_{ij}]$ be matrices with the same dimensions as U and V respectively. The elements of the matrices α and β are the corresponding Lagrange multipliers for the non-negativity conditions on the different elements of U and V respectively. We note that $tr(\alpha \cdot U^T)$ is simply equal to $\sum_{i,j} \alpha_{ij} \cdot u_{ij}$ and $tr(\beta \cdot V^T)$ is simply equal to $\sum_{i,j} \beta_{ij} \cdot v_{ij}$. These correspond to the lagrange expressions for the non-negativity constraints. Then, we can express the Lagrangian optimization problem as follows:

$$L = J + tr(\alpha \cdot U^T) + tr(\beta \cdot V^T) \tag{4.9}$$

Then, we can express the partial derivative of L with respect to U and V as follows, and set them to 0:

$$\frac{\delta L}{\delta U} = -A \cdot V + U \cdot V^T \cdot V + \alpha = 0$$
$$\frac{\delta L}{\delta V} = -A^T \cdot U + V \cdot U^T \cdot U + \beta = 0$$

We can then multiply the (i, j)th entry of the above (two matrices of) conditions with u_{ij} and v_{ij} respectively. Using the Kuhn-Tucker conditions $\alpha_{ij} \cdot u_{ij} = 0$ and $\beta_{ij} \cdot v_{ij} = 0$, we get the following:

$$(A \cdot V)_{ij} \cdot u_{ij} - (U \cdot V^T \cdot V)_{ij} \cdot u_{ij} = 0$$
$$(A^T \cdot U)_{ij} \cdot v_{ij} - (V \cdot U^T \cdot U)_{ij} \cdot v_{ij} = 0$$

We note that these conditions are independent of α and β. This leads to the following iterative updating rules for u_{ij} and v_{ij}:

$$u_{ij} = \frac{(A \cdot V)_{ij} \cdot u_{ij}}{(U \cdot V^T \cdot V)_{ij}}$$
$$v_{ij} = \frac{(A^T \cdot U)_{ij} \cdot v_{ij}}{(V \cdot U^T \cdot U)_{ij}}$$

It has been shown in [58] that the objective function continuously improves under these update rules, and converges to an optimal solution.

One interesting observation about the matrix factorization technique is that it can also be used to determine word-clusters instead of document clusters. Just as the columns of V provide a basis which can be used to discover document clusters, we can use the columns of U to discover a basis which correspond to word clusters. As we will see later, document clusters and word clusters are closely related, and it is often useful to discover both simultaneously, as in frameworks such as *co-clustering* [30, 31, 75]. Matrix-factorization provides a natural way of achieving this goal. It has also been shown both theoretically and experimentally [33, 93] that the matrix-factorization technique is equivalent to another graph-structure based document clustering technique known as *spectral clustering.* An analogous technique called *concept factorization* was proposed in [98], which can also be applied to data points with negative values in them.

3. Distance-based Clustering Algorithms

Distance-based clustering algorithms are designed by using a similarity function to measure the closeness between the text objects. The most well known similarity function which is used commonly in the text domain is the cosine similarity function. Let $U = (f(u_1) \dots f(u_k))$ and $V = (f(v_1) \dots f(v_k))$ be the damped and normalized frequency term vector in two different documents U and V. The values $u_1 \dots u_k$ and $v_1 \dots v_k$ represent the (normalized) term frequencies, and the function $f(\cdot)$ represents the damping function. Typical damping functions for $f(\cdot)$ could represent either the square-root or the logarithm [25]. Then, the cosine similarity between the two documents is defined as follows:

$$cosine(U, V) = \frac{\sum_{i=1}^{k} f(u_i) \cdot f(v_i)}{\sqrt{\sum_{i=1}^{k} f(u_i)^2} \cdot \sqrt{\sum_{i=1}^{k} f(v_i)^2}} \quad (4.10)$$

Computation of text similarity is a fundamental problem in information retrieval. Although most of the work in information retrieval has focused on how to assess the similarity of a keyword query and a text document, rather than the similarity between two documents, many weighting heuristics and similarity functions can also be applied to optimize the similarity function for clustering. Effective information retrieval models generally capture three heuristics, i.e., TF weighting, IDF weighting, and document length normalization [36]. One effective way to assign weights to terms when representing a document as a weighted term vector is the BM25 term weighting method [78], where the normalized TF not only addresses length normalization, but also has an upper bound which improves the robustness as it avoids overly rewarding the matching of any particular term. A document can also be represented with a probability distribution over words (i.e., unigram language models), and the similarity can then be measured based an information theoretic measure such as cross entropy or Kullback-Leibler divergencce [105]. For clustering, symmetric variants of such a similarity function may be more appropriate.

One challenge in clustering short segments of text (e.g., tweets or sentences) is that exact keyword matching may not work well. One general strategy for solving this problem is to expand text representation by exploiting related text documents, which is related to smoothing of a document language model in information retrieval [105]. A specific technique, which leverages a search engine to expand text representation, was proposed in [79]. A comparison of several simple measures for computing similarity of short text segments can be found in [66].

These similarity functions can be used in conjunction with a wide variety of traditional clustering algorithms [50, 54]. In the next subsections, we will discuss some of these techniques.

3.1 Agglomerative and Hierarchical Clustering Algorithms

Hierarchical clustering algorithms have been studied extensively in the clustering literature [50, 54] for records of different kinds including multidimensional numerical data, categorical data and text data. An overview of the traditional agglomerative and hierarchical clustering algorithms in the context of text data is provided in [69, 70, 92, 96]. An experimental comparison of different hierarchical clustering algorithms may be found in [110]. The method of agglomerative hierarchical clustering is particularly useful to support a variety of searching methods because it naturally creates a tree-like hierarchy which can be leveraged for the search process. In particular, the effectiveness of this method in improving the search efficiency over a sequential scan has been shown in [51, 77].

The general concept of agglomerative clustering is to successively merge documents into clusters based on their similarity with one another. Almost all the hierarchical clustering algorithms successively merge groups based on the best pairwise similarity between these groups of documents. The main differences between these classes of methods are in terms of how this pairwise similarity is computed between the different groups of documents. For example, the similarity between a pair of groups may be computed as the best-case similarity, average-case similarity, or worst-case similarity between documents which are drawn from these pairs of groups. Conceptually, the process of agglomerating documents into successively higher levels of clusters creates a cluster hierarchy (or dendogram) for which the leaf nodes correspond to individual documents, and the internal nodes correspond to the merged groups of clusters. When two groups are merged, a new node is created in this tree corresponding to this larger merged group. The two children of this node correspond to the two groups of documents which have been merged to it.

The different methods for merging groups of documents for the different agglomerative methods are as follows:

- **Single Linkage Clustering:** In single linkage clustering, the similarity between two groups of documents is the greatest similarity between any pair of documents from these two groups. In single link clustering we merge the two groups which are such that their closest pair of documents have the highest similarity compared to any other pair of groups. The main advantage of single linkage clustering is that it is extremely efficient to implement in practice. This is because we can first compute all similarity pairs and sort them in order of reducing similarity. These pairs are processed in this pre-defined order and the merge is performed successively if the pairs belong to different groups. It can be easily shown that this approach is equivalent to the single-linkage method. This is essentially equivalent to a spanning tree algorithm on the complete graph of pairwise-distances by processing the edges of the graph in a certain order. It has been shown in [92] how Prim's minimum spanning tree algorithm can be adapted to single-linkage clustering. Another method in [24] designs the single-linkage method in conjunction with the inverted index method in order to avoid computing zero similarities.

  The main drawback of this approach is that it can lead to the phenomenon of *chaining* in which a chain of similar documents lead to disparate documents being grouped into the same clusters. In other words, if A is similar to B and B is similar to C, it does not always imply that A is similar to C, because of lack of transitivity in similarity computations. Single linkage clustering encourages the grouping of documents through such transitivity chains. This can often lead to poor clusters, especially at the higher levels of the agglomeration. Effective methods for implementing single-linkage clustering for the case of document data may be found in [24, 92].

- **Group-Average Linkage Clustering:** In group-average linkage clustering, the similarity between two clusters is the *average* similarity between the pairs of documents in the two clusters. Clearly, the average linkage clustering process is somewhat slower than single-linkage clustering, because we need to determine the average similarity between a large number of pairs in order to determine group-wise similarity. On the other hand, it is much more robust in terms of clustering quality, because it does not exhibit the chaining behavior of single linkage clustering. It is possible to speed up the average linkage clustering algorithm by approximating the average linkage similarity between two clusters C_1 and C_2 by computing the similarity between the mean document of C_1

and the mean document of C_2. While this approach does not work equally well for all data domains, it works particularly well for the case of text data. In this case, the running time can be reduced to $O(n^2)$, where n is the total number of nodes. The method can be implemented quite efficiently in the case of document data, because the centroid of a cluster is simply the concatenation of the documents in that cluster.

- **Complete Linkage Clustering:** In this technique, the similarity between two clusters is the *worst-case* similarity between any pair of documents in the two clusters. Complete-linkage clustering can also avoid chaining because it avoids the placement of any pair of very disparate points in the same cluster. However, like group-average clustering, it is computationally more expensive than the single-linkage method. The complete linkage clustering method requires $O(n^2)$ space and $O(n^3)$ time. The space requirement can however be significantly lower in the case of the text data domain, because a large number of pairwise similarities are zero.

Hierarchical clustering algorithms have also been designed in the context of text data streams. A distributional modeling method for hierarchical clustering of streaming documents has been proposed in [80]. The main idea is to model the frequency of word-presence in documents with the use of a multi-poisson distribution. The parameters of this model are learned in order to assign documents to clusters. The method extends the COBWEB and CLASSIT algorithms [37, 40] to the case of text data. The work in [80] studies the different kinds of distributional assumptions of words in documents. We note that these distributional assumptions are required to adapt these algorithms to the case of text data. The approach essentially changes the distributional assumption so that the method can work effectively for text data.

3.2 Distance-based Partitioning Algorithms

Partitioning algorithms are widely used in the database literature in order to efficiently create clusters of objects. The two most widely used distance-based partitioning algorithms [50, 54] are as follows:

- **k-medoid clustering algorithms:** In k-medoid clustering algorithms, we use a set of points from the original data as the anchors (or medoids) around which the clusters are built. The key aim of the algorithm is to determine an optimal set of representative documents *from the original corpus* around which the clusters are built. Each document is assigned to its closest representative from the collection. This creates a running set of clusters from the corpus which are successively improved by a randomized process.

The algorithm works with an iterative approach in which the set of k representatives are successively improved with the use of randomized inter-changes. Specifically, we use the average similarity of each document in the corpus to its closest representative as the objective function which needs to be improved during this interchange process. In each iteration, we replace a randomly picked representative in the current set of medoids with a randomly picked representative from the collection, if it improves the clustering objective function. This approach is applied until convergence is achieved.

There are two main disadvantages of the use of k-medoids based clustering algorithms, one of which is specific to the case of text data. One general disadvantage of k-medoids clustering algorithms is that they require a large number of iterations in order to achieve convergence and are therefore quite slow. This is because each iteration requires the computation of an objective function whose time requirement is proportional to the size of the underlying corpus.

The second key disadvantage is that k-medoid algorithms do not work very well for sparse data such as text. This is because a large fraction of document pairs do not have many words in common, and the similarities between such document pairs are small (and noisy) values. Therefore, a single document medoid often does not contain all the concepts required in order to effectively build a cluster around it. This characteristic is specific to the case of the information retrieval domain, because of the sparse nature of the underlying text data.

- ***k*-means clustering algorithms:** The k-means clustering algorithm also uses a set of k representatives around which the clusters are built. However, these representatives are not necessarily obtained from the original data and are refined somewhat differently than a k-medoids approach. The simplest form of the k-means approach is to start off with a set of k seeds from the original corpus, and assign documents to these seeds on the basis of closest similarity. In the next iteration, the centroid of the assigned points to each seed is used to replace the seed in the last iteration. In other words, the new seed is defined, so that it is a better central point for this cluster. This approach is continued until convergence. One of the advantages of the k-means method over the k-medoids method is that it requires an extremely small number of iterations in order to converge. Observations from [25, 83] seem to suggest that for many large data sets, it is sufficient to use 5 or less iterations for an effective clustering. The main disadvantage of the k-means method is that it is still quite sensitive to the initial set of seeds picked during the clustering. Secondly, the centroid for a given cluster of documents may contain a large number of words. This will slow down the similarity calculations in the next iteration. A number of methods are used to reduce these effects, which will be discussed later on in this chapter.

The initial choice of seeds affects the quality of k-means clustering, especially in the case of document clustering. Therefore, a number of techniques are used in order to improve the quality of the initial seeds which are picked for the clustering process. For example, another lightweight clustering method such as an agglomerative clustering technique can be used in order to decide the initial set of seeds. This is at the core of the method discussed in [25] for effective document clustering. We will discuss this method in detail in the next subsection.

A second method for improving the initial set of seeds is to use some form of partial supervision in the process of initial seed creation. This form of partial supervision can also be helpful in creating clusters which are designed for particular application-specific criteria. An example of such an approach is discussed in [4] in which we pick the initial set of seeds as the centroids of the documents crawled from a particular category if the $Yahoo$! taxonomy. This also has the effect that the final set of clusters are grouped by the coherence of content within the different $Yahoo$! categories. The approach has been shown to be quite effective for use in a number of applications such as text categorization. Such semi-supervised techniques are particularly useful for information organization in cases where the starting set of categories is somewhat noisy, but contains enough information in order to create clusters which satisfy a pre-defined kind of organization.

3.3 A Hybrid Approach: The Scatter-Gather Method

While hierarchical clustering methods tend to be more robust because of their tendency to compare all pairs of documents, they are generally not very efficient, because of their tendency to require at least $O(n^2)$ time. On the other hand, k-means type algorithms are more efficient than hierarchical algorithms, but may sometimes not be very effective because of their tendency to rely on a small number of seeds.

The method in [25] uses both hierarchical and partitional clustering algorithms to good effect. Specifically, it uses a hierarchical clustering algorithm on a sample of the corpus in order to find a robust initial set of seeds. This robust set of seeds is used in conjunction with a standard k-means clustering algorithm in order to determine good clusters. The size of the sample in the initial phase is carefully tailored so as to provide the best possible effectiveness without this phase becoming a bottleneck in algorithm execution.

There are two possible methods for creating the initial set of seeds, which are referred to as *buckshot* and *fractionation* respectively. These are two alternative methods, and are described as follows:

- **Buckshot:** Let k be the number of clusters to be found and n be the number of documents in the corpus. Instead of picking the k seeds randomly from the collection, the buckshot scheme picks an overestimate $\sqrt{k \cdot n}$ of the seeds, and then agglomerates these to k seeds. Standard agglomerative hierarchical clustering algorithms (requiring quadratic time) are applied to this initial sample of $\sqrt{k \cdot n}$ seeds. Since we use quadratically scalable algorithms in this phase, this approach requires $O(k \cdot n)$ time. We note that this seed set is much more robust than one which simply samples for k seeds, because of the summarization of a large document sample into a robust set of k seeds.

- **Fractionation:** The fractionation algorithm initially breaks up the corpus into n/m buckets of size $m > k$ each. An agglomerative algorithm is applied to each of these buckets to reduce them by a factor of ν. Thus, at the end of the phase, we have a total of $\nu \cdot n$ agglomerated points. The process is repeated by treating each of these agglomerated points as an individual record. This is achieved by merging the different documents within an agglomerated cluster into a single document. The approach terminates when a total of k seeds remain. We note that the the agglomerative clustering of each group of m documents in the first iteration of the fractionation algorithm requires $O(m^2)$ time, which sums to $O(n \cdot m)$ over the n/m different groups. Since, the number of individuals reduces geometrically by a factor of ν in each iteration, the total running time over all iterations is $O(n \cdot m \cdot (1 + \mu + \nu^2 + \ldots))$. For constant $\nu < 1$, the running time over all iterations is still $O(n \cdot m)$. By picking $m = O(k)$, we can still ensure a running time of $O(n \cdot k)$ for the initialization procedure.

The *Buckshot* and *Fractionation* procedures require $O(k \cdot n)$ time which is also equivalent to running time of one iteration of the k means algorithm.

Each iteration of the K-means algorithm also requires $O(k \cdot n)$ time because we need to compute the similarity of the n documents to the k different seeds.

We further note that the fractionation procedure can be applied to a random grouping of the documents into n/m different buckets. Of course, one can also replace the random grouping approach with a more carefully designed procedure for more effective results. One such procedure is to sort the documents by the index of the jth most common word in the document. Here j is chosen to be a small number such as 3, which corresponds to medium frequency words in the data. The documents are then partitioned into groups based on this sort order by segmenting out continuous groups of m documents. This approach ensures that the groups created have at least a few common words in them and are therefore not completely random. This can sometimes provide a better quality of the centers which are determined by the fractionation algorithm.

Once the initial cluster centers have been determined with the use of the *Buckshot* or *Fractionation* algorithms we can apply standard k-means partitioning algorithms. Specifically, we each document is assigned to the nearest of the k cluster centers. The centroid of each such cluster is determined as the concatenation of the different documents in a cluster. These centroids replace the sets of seeds from the last iteration. This process can be repeated in an iterative approach in order to successively refine the centers for the clusters. Typically, only a smaller number of iterations are required, because the greatest improvements occur only in the first few iterations.

It is also possible to use a number of procedures to further improve the quality of the underlying clusters. These procedures are as follows:

- **Split Operation:** The process of splitting can be used in order to further refine the clusters into groups of better granularity. This can be achieved by applying the buckshot procedure on the individual documents in a cluster by using $k = 2$, and then re-clustering around these centers. This entire procedure requires $O(k \cdot n_i)$ time for a cluster containing n_i data points, and therefore splitting all the groups requires $O(k \cdot n)$ time. However, it is not necessary to split *all* the groups. Instead, only a subset of the groups can be split. Those are the groups which are not very coherent and contain documents of a disparate nature. In order to measure the coherence of a group, we compute the self-similarity of a cluster. This self-similarity provides us with an understanding of the underlying coherence. This quantity can be computed both in terms of the similarity of the documents in a cluster to its centroid or in terms of the similarity of the cluster documents to each other. The split criterion can then be applied selectively only to those clusters which have low self similarity. This helps in creating more coherent clusters.

- **Join Operation:** The join operation attempts to merge similar clusters into a single cluster. In order to perform the merge, we compute the *topical* words of each cluster by examining the most frequent words of the centroid. Two clusters are considered similar, if there is significant overlap between the topical words of the two clusters.

We note that the method is often referred to as the *Scatter-Gather* clustering method, but this is more because of how the clustering method has been presented in terms of its use for browsing large collections in the original paper [25]. The scatter-gather approach can be used for organized browsing of large document collections, because it creates a natural hierarchy of similar documents. In particular, a user may wish to browse the hierarchy of clusters in an interactive way in order to understand topics of different levels of granularity in the collection. One possibility is to perform a hierarchical clustering a-priori; however such an approach has the disadvantage that it is unable to merge and re-cluster related branches of the tree hierarchy on-the-fly when a user may need it. A method for constant-interaction time browsing with the use of the scatter-gather approach has been presented in [26]. This approach presents the keywords associated with the different keywords to a user. The user may pick one or more of these keywords, which also corresponds to one or more clusters. The documents in these clusters are merged and re-clustered to a finer-granularity on-the-fly. This finer granularity of clustering is presented to the user for further exploration. The set of documents which is picked by the user for exploration is referred to as the *focus set*. Next we will explain how this focus set is further explored and re-clustered on the fly in constant-time.

The key assumption in order to enable this approach is the *cluster refinement hypothesis*. This hypothesis states that documents which belong to the same cluster in a significantly finer granularity partitioning will also occur together in a partitioning with coarser granularity. The first step is to create a hierarchy of the documents in the clusters. A variety of agglomerative algorithms such as the buckshot method can be used for this purpose. We note that each (internal) node of this tree can be viewed as a meta-document corresponding to the concatenation of all the documents in the leaves of this subtree. The cluster-refinement hypothesis allows us to work with a smaller set of meta-documents rather than the entire set of documents in a particular subtree. The idea is to pick a constant M which represents the maximum number of meta-documents that we are willing to re-cluster with the use of the interactive approach. The tree nodes in the focus set are then expanded (with priority to the branches with largest degree), to a maximum of M nodes. These M nodes are then re-clustered on-the-fly with the scatter-gather approach. This requires constant time because of the use of a constant number M of meta-documents in the clustering process. Thus, by working with the meta-documents for M. we assume the cluster-refinement hypothesis of all nodes of the subtree at the lower level. Clearly, a larger value of M does not assume the cluster-refinement hypothesis quite as strongly, but also comes at a higher cost. The details of the algorithm are described in [26]. Some extensions of this approach are also presented in [85], in which it has been shown how this approach can be used to cluster arbitrary corpus subsets of the documents in constant time. Another recent online clustering algorithm called *LAIR2* [55] provides constant-interaction time for Scatter/Gather browsing. The parallelization of this algorithm is significantly faster than a corresponding version of the Buckshot algorithm. It has also been suggested that the *LAIR2* algorithm leads to better quality clusters in the data.

3.3.1 Projections for Efficient Document Clustering.

One of the challenges of the scatter-gather algorithm is that even though the algorithm is designed to balance the running times of the agglomerative and partitioning phases quite well, it sometimes suffer a slowdown in large document collections because of the massive number of distinct terms that a given cluster centroid may contain. Recall that a cluster centroid in the scatter-gather algorithm is defined as the concatenation of all the documents in that collection. When the number of documents in the cluster is large, this will also lead to a large number of distinct terms in the centroid. This will also lead to a slow down of a number of critical computations such as similarity calculations between documents and cluster centroids.

An interesting solution to this problem has been proposed in [83]. The idea is to use the concept of *projection* in order to reduce the dimensionality of the document representation. Such a reduction in dimensionality will lead to significant speedups, because the similarity computations will be made much more efficient. The work in [83] proposes three kinds of projections:

- **Global Projection:** In global projection, the dimensionality of the original data set is reduced in order to remove the least important (weighted) terms from the data. The weight of a term is defined as the aggregate of the (normalized and damped) frequencies of the terms in the documents.

- **Local Projection:** In local projection, the dimensionality of the documents in each cluster are reduced with a *locally specific approach* for that cluster. Thus, the terms in each cluster centroid are truncated separately. Specifically, the least weight terms in the different cluster centroids are removed. Thus, the terms removed from each document may be different, depending upon their local importance.

- **Latent Semantic Indexing:** In this case, the document-space is transformed with an LSI technique, and the clustering is applied to the transformed document space. We note that the LSI technique can also be applied either globally to the whole document collection, or locally to each cluster if desired.

It has been shown in [83] that the projection approaches provide competitive results in terms of effectiveness while retaining an extremely high level of efficiency with respect to all the competing approaches. In this sense, the clustering methods are different from similarity search because they show little degradation in quality, when projections are performed. One of the reasons for this is that clustering is a much less fine grained application as compared to similarity search, and therefore there is no perceptible difference in quality even when we work with a truncated feature space.

4. Word and Phrase-based Clustering

Since text documents are drawn from an inherently high-dimensional domain, it can be useful to view the problem in a dual way, in which important clusters of words may be found and utilized for finding clusters of documents. In a corpus containing d terms and n documents, one may view a term-document matrix as an $n \times d$ matrix, in which the (i, j)th entry is the frequency of the jth term in the ith document. We note that this matrix is extremely sparse since a given document contains an extremely small fraction of the universe of words. We note that the problem of clustering *rows* in this matrix is that of clustering documents, whereas that of clustering *columns* in this matrix is that of clustering words. In reality, the two problems are closely related, as good clusters of words may be leveraged in order to find good clusters of documents and vice-versa. For example, the work in [16] determines frequent itemsets of words in the document collection, and uses them to determine compact clusters of documents. This is somewhat analogous to the use of clusters of words [87] for determining clusters of documents. The most general technique for simultaneous word and document clustering is referred to as *co-clustering* [30, 31]. This approach simultaneous clusters the rows and columns of the term-document matrix, in order to create such clusters. This can also be considered to be equivalent to the problem of re-ordering the rows and columns of the term-document matrix so as to create dense rectangular blocks of non-zero entries in this matrix. In some cases, the ordering information among words may be used in order to determine good clusters. The work in [103] determines the frequent phrases in the collection and leverages them in order to determine document clusters.

It is important to understand that the problem of word clusters and document clusters are essentially dual problems which are closely related to one another. The former is related to dimensionality reduction, whereas the latter is related to traditional clustering. The boundary between the two problems is quite fluid, because good word clusters provide hints for finding good document clusters and vice-versa. For example, a more general probabilistic framework which determines word clusters and document clusters simultaneously is referred to as *topic modeling* [49]. Topic modeling is a more general framework than either clustering or dimensionality reduction. We will introduce the method of topic modeling in a later section of this chapter. A more detailed treatment is also provided in the next chapter in this book, which is on dimensionality reduction, and in Chapter 8 where a more general discussion of probabilistic models for text mining is given.

4.1 Clustering with Frequent Word Patterns

Frequent pattern mining [8] is a technique which has been widely used in the data mining literature in order to determine the most relevant patterns in transactional data. The clustering approach in [16] is designed on the basis of such frequent pattern mining algorithms. A frequent itemset in the context of text data is also referred to as a *frequent term set*, because we are dealing with documents rather than transactions. The main idea of the approach is to not cluster the high dimensional document data set, but consider the low dimensional frequent term sets as cluster candidates. This essentially means that a frequent terms set is a description of a cluster which corresponds to all the documents containing that frequent term set. Since a frequent term set can be considered a description of a cluster, a set of carefully chosen frequent terms sets can be considered a clustering. The appropriate choice of this set of frequent term sets is defined on the basis of the overlaps between the supporting documents of the different frequent term sets.

The notion of clustering defined in [16] does not necessarily use a strict partitioning in order to define the clusters of documents, but it allows a certain level of overlap. This is a natural property of many term- and phrase-based clustering algorithms because one does not directly control the assignment of documents to clusters during the algorithm execution. Allowing some level of overlap between clusters may sometimes be more appropriate, because it recognizes the fact that documents are complex objects and it is impossible to cleanly partition documents into specific clusters, especially when some of the clusters are partially related to one another. The clustering definition of [16] assumes that each document is covered by at least one frequent term set.

Let R be the set of chosen frequent term sets which define the clustering. Let f_i be the number of frequent term sets in R which are contained in the ith document. The value of f_i is at least one in order to ensure complete coverage, but we would otherwise like it to be as low as possible in order to minimize overlap. Therefore, we would like the average value of $(f_i - 1)$ for the documents in a given cluster to be as low as possible. We can compute the average value of $(f_i - 1)$ for the documents in the cluster and try to pick frequent term sets such that this value is as low as possible. However, such an approach would tend to favor frequent term sets containing very few terms. This is because if a term set contains m terms, then all subsets of it would also be covered by the document, as a result of which the standard overlap would be increased. The entropy overlap of a given term is essentially the sum of the values of $-(1/f_i) \cdot \log(1/f_i)$ over all documents in the cluster. This value is 0, when each document has $f_i = 1$, and increases monotonically with increasing f_i values.

It then remains to describe how the frequent term sets are selected from the collection. Two algorithms are described in [16], one of which corresponds to a flat clustering, and the other corresponds to a hierarchical clustering. We will first describe the method for flat clustering. Clearly, the search space of frequent terms is exponential, and therefore a reasonable solution is to utilize a greedy algorithm to select the frequent terms sets. In each iteration of the greedy algorithm, we pick the frequent term set with a cover having the minimum overlap with other cluster candidates. The documents covered by the selected frequent term are removed from the database, and the overlap in the next iteration is computed with respect to the remaining documents.

The hierarchical version of the algorithm is similar to the broad idea in flat clustering, with the main difference that each level of the clustering is applied to a set of term sets containing a fixed number k of terms. In other words, we are working only with frequent patterns of length k for the selection process. The resulting clusters are then further partitioned by applying the approach for $(k+1)$-term sets. For further partitioning a given cluster, we use only those $(k+1)$-term sets which contain the frequent k-term set defining that cluster. More details of the approach may be found in [16].

4.2 Leveraging Word Clusters for Document Clusters

A two phase clustering procedure is discussed in [87], which uses the following steps to perform document clustering:

- In the first phase, we determine word-clusters from the documents in such a way that most of mutual information between words and documents is preserved when we represent the documents in terms of word clusters rather than words.

- In the second phase, we use the condensed representation of the documents in terms of word-clusters in order to perform the final document clustering. Specifically, we replace the word occurrences in documents with word-cluster occurrences in order to perform the document clustering. One advantage of this two-phase procedure is the significant reduction in the noise in the representation.

Let $X = x_1 \dots x_n$ be the random variables corresponding to the rows (documents), and let $Y = y_1 \dots y_d$ be the random variables corresponding to the columns (words). We would like to partition X into k clusters, and Y into l clusters. Let the clusters be denoted by $\hat{X} = \hat{x_1} \dots \hat{x_k}$ and $\hat{Y} = \hat{y_1} \dots \hat{y_l}$. In other words, we wish to find the maps C_X and C_Y, which define the clustering:

$$C_X : x_1 \dots x_n \Rightarrow \hat{x_1} \dots \hat{x_k}$$
$$C_Y : y_1 \dots y_d \Rightarrow \hat{y_1} \dots \hat{y_l}$$

In the first phase of the procedure we cluster Y to $\hat{Y}$, so that most of the information in $I(X, Y)$ is preserved in $I(X, \hat{Y})$. In the second phase, we perform the clustering again from X to $\hat{X}$ using exactly the same procedure so that as much information as possible from $I(X, \hat{Y})$ is preserved in $I(\hat{X}, \hat{Y})$. Details of how each phase of the clustering is performed is provided in [87].

How to discover interesting word clusters (which can be leveraged for document clustering) has itself attracted attention in the natural language processing research community, with particular interests in discovering word clusters that can characterize word senses [34] or a semantic concept [21]. In [34], for example, the Markov clustering algorithm was applied to discover corpus-specific word senses in an unsupervised way. Specifically, a word association graph is first constructed in which related words would be connected with an edge. For a given word that potentially has multiple senses, we can then isolate the subgraph representing its neighbors. These neighbors are expected to form clusters according to different senses of the target word, thus by grouping together neighbors that are well connected with each other, we can discover word clusters that characterize different senses of the target word. In [21], an n-gram class language model was proposed to cluster words based on minimizing the loss of mutual information between adjacent words, which can achieve the effect of grouping together words that share similar context in natural language text.

4.3 Co-clustering Words and Documents

In many cases, it is desirable to simultaneously cluster the rows and columns of the contingency table, and explore the interplay between word clusters and document clusters during the clustering process. Since the clusters among words and documents are clearly related, it is often desirable to cluster both simultaneously when when it is desirable to find clusters along one of the two dimensions. Such an approach is referred to as *co-clustering* [30, 31]. Co-clustering is defined as a pair of maps from rows to row-cluster indices and columns to column-cluster indices. These maps are determined *simultaneously* by the algorithm in order to optimize the corresponding cluster representations.

We further note that the matrix factorization approach [58] discussed earlier in this chapter can be naturally used for co-clustering because it discovers word clusters and document clusters simultaneously. In that section, we have also discussed how matrix factorization can be viewed as a co-clustering technique. While matrix factorization has not widely been used as a technique for co-clustering, we point out this natural connection, as possible exploration for future comparison with other co-clustering methods. Some recent work [60] has shown how matrix factorization can be used in order to transform knowledge from word space to document space in the context of document clustering techniques.

The problem of co-clustering is also closely related to the problem of *subspace clustering* [7] or *projected clustering* [5] in quantitative data in the database literature. In this problem, the data is clustered by simultaneously associating it with a set of points and subspaces in multidimensional space. The concept of co-clustering is a natural application of this broad idea to data domains which can be represented as **sparse** high dimensional matrices in which most of the entries are 0. Therefore, traditional methods for subspace clustering can also be extended to the problem of co-clustering. For example, an adaptive iterative subspace clustering method for documents was proposed in [59].

We note that subspace clustering or co-clustering can be considered a form of *local feature selection*, in which the features selected are specific to each cluster. A natural question arises, as to whether the features can be selected as a linear combination of dimensions as in the case of traditional dimensionality reduction techniques such as PCA [53]. This is also known as *local dimensionality reduction* [22] or *generalized projected clustering* [6] in the traditional database literature. In this method, PCA-based techniques are used in order to generate subspace representations which are *specific to each cluster*, and are leveraged in order to achieve a better clustering process. In particular, such an approach has recently been designed [32], which has been shown to work well with document data.

In this section, we will study two well known methods for document co-clustering, which are commonly used in the document clustering literature. One of these methods uses graph-based term-document representations [30] and the other uses information theory [31]. We will discuss both of these methods below.

4.3.1 Co-clustering with graph partitioning. The core idea in this approach [30] is to represent the term-document matrix as a bipartite graph $G = (V_1 \cup V_2, E)$, where V_1 and V_2 represent the vertex sets in the two bipartite portions of this graph, and E represents the edge set. Each node in V_1 corresponds to one of the n documents, and each node in V_2 corresponds to one of the d terms. An undirected edge exists between node $i \in V_1$ and node $j \in V_2$ if document i contains the term j. We note that there are no edges in E directly between terms, or directly between documents. Therefore, the graph is bipartite. The weight of each edge is the corresponding normalized term-frequency.

We note that a word partitioning in this bipartite graph induces a document partitioning and vice-versa. Given a partitioning of the documents in this graph, we can associate each word with the document cluster to which it is connected with the most weight of edges. Note that this criterion also minimizes the weight of the edges across the partitions. Similarly, given a word partitioning, we can associate each document with the word partition to which it is connected with the greatest weight of edges. Therefore, a natural solution to this problem would be *simultaneously* perform the k-way partitioning of this graph which minimizes the total weight of the edges across the partitions. This is of course a classical problem in the graph partitioning literature. In [30], it has been shown how a spectral partitioning algorithm can be used effectively for this purpose. Another method discussed in [75] uses an isometric bipartite graph-partitioning approach for the clustering process.

4.3.2 Information-Theoretic Co-clustering. In [31], the optimal clustering has been defined to be one which maximizes the mutual information between the clustered random variables. The normalized non-negative contingency table is treated as a joint probability distribution between two discrete random variables which take values over rows and columns. Let $X = x_1 \dots x_n$ be the random variables corresponding to the rows, and let $Y = y_1 \dots y_d$ be the random variables corresponding to the columns. We would like to partition X into k clusters, and Y into l clusters. Let the clusters be denoted by $\hat{X} = \hat{x_1} \dots \hat{x_k}$ and $\hat{Y} = \hat{y_1} \dots \hat{y_l}$. In other words, we wish to find the maps C_X and C_Y, which define the clustering:

$$C_X : x_1 \dots x_n \Rightarrow \hat{x_1} \dots \hat{x_k}$$
$$C_Y : y_1 \dots y_d \Rightarrow \hat{y_1} \dots \hat{y_l}$$

The partition functions C_X and C_Y are allowed to depend on the joint probability distribution $p(X, Y)$. We note that since $\hat{X}$ and $\hat{Y}$ are higher level clusters of X and Y, there is loss in mutual information in the higher level representations. In other words, the distribution $p(\hat{X}, \hat{Y})$ contains less information than $p(X, Y)$, and the mutual information $I(\hat{X}, \hat{Y})$ is lower than the mutual information $I(X, Y)$. Therefore, the optimal co-clustering problem is to determine the mapping which minimizes the loss in mutual information. In other words, we wish to find a co-clustering for which $I(X, Y) - I(\hat{X}, \hat{Y})$ is as small as possible. An iterative algorithm for finding a co-clustering which minimizes mutual information loss is proposed in [29].

4.4 Clustering with Frequent Phrases

One of the key differences of this method from other text clustering methods is that it treats a document as a string as opposed to a bag of words. Specifically, each document is treated as a string of *words*, rather than characters. The main difference between the string representation and the bag-of-words representation is that the former also retains ordering information for the clustering process. As is the case with many clustering methods, it uses an indexing method in order to organize the phrases in the document collection, and then uses this organization to create the clusters [103, 104]. Several steps are used in order to create the clusters:

(1) The first step is to perform the cleaning of the strings representing the documents. A light stemming algorithm is used by deleting word prefixes and suffixes and reducing plural to singular. Sentence boundaries are marked and non-word tokens are stripped.

(2) The second step is the identification of base clusters. These are defined by the frequent phases in the collection which are represented in the form of a *suffix tree*. A suffix tree [45] is essentially a trie which contains all the suffixes of the entire collection. Each node of the suffix tree represents a group of documents, and a phrase which is common to all these documents. Since each node of the suffix-tree also corresponds to a group of documents, it also corresponds to a base clustering. Each base cluster is given a score which is essentially the product of the number of documents in that cluster and a non-decreasing function of the length of the underlying phrase. Therefore, clusters containing a large number of documents, and which are defined by a relatively long phrase are more desirable.

(3) An important characteristic of the base clusters created by the suffix tree is that they do not define a strict partitioning and have overlaps with one another. For example, the same document may contain multiple phrases in different parts of the suffix tree, and will therefore be included in the corresponding document groups. The third step of the algorithm merges the clusters based on the similarity of their underlying document sets. Let P and Q be the document sets corresponding to two clusters. The base similarity $BS(P, Q)$ is defined as follows:

$$BS(P, Q) = \left\lfloor \frac{|P \cap Q|}{\max\{|P|, |Q|\}} + 0.5 \right\rfloor \tag{4.11}$$

This base similarity is either 0 or 1, depending upon whether the two groups have at least 50% of their documents in common. Then, we construct a graph structure in which the nodes represent the base clusters, and an edge exists between two cluster nodes, if the corresponding base similarity between that pair of groups is 1. The connected components in this graph define the final clusters. Specifically, the union of the groups of documents in each connected component is used as the final set of clusters. We note that the final set of clusters have much less overlap with one another, but they still do not define a strict partitioning. This is sometimes the case with clustering algorithms in which modest overlaps are allowed to enable better clustering quality.

5. Probabilistic Document Clustering and Topic Models

A popular method for probabilistic document clustering is that of *topic modeling*. The idea of topic modeling is to create a *probabilistic generative model* for the text documents in the corpus. The main approach is to represent a corpus as a function of hidden random variables, the parameters of which are estimated using a particular document collection. The primary assumptions in any topic modeling approach (together with the corresponding random variables) are as follows:

- The n documents in the corpus are assumed to have a probability of belonging to one of k topics. Thus, a given document may have a probability of belonging to multiple topics, and this reflects the fact that the same document may contain a multitude of subjects. For a given document D_i, and a set of topics $T_1 \dots T_k$, the probability that the document D_i belongs to the topic T_j is given by $P(T_j|D_i)$. We note that the the topics are essentially analogous to clusters, and the value of $P(T_j|D_i)$ provides a probability of cluster membership of the ith document to the jth cluster. In non-probabilistic clustering methods, the membership of documents to clusters is deterministic in nature, and therefore the clustering is typically a clean partitioning of the document collection. However, this often creates challenges, when there are overlaps in document subject matter across multiple clusters. The use of a *soft cluster membership in terms of probabilities* is an elegant solution to this dilemma. In this scenario, the determination of the membership of the documents to clusters is a secondary goal to that of finding the *latent topical clusters* in the underlying text collection. Therefore, this area of research is referred to as *topic modeling*, and while it is related to the clustering problem, it is often studied as a distinct area of research from clustering.

  The value of $P(T_j|D_i)$ is estimated using the topic modeling approach, and is one of the primary outputs of the algorithm. The value of k is one of the inputs to the algorithm and is analogous to the number of clusters.

- Each topic is associated with a probability vector, which quantifies the probability of the different terms in the lexicon for that topic. Let $t_1 \dots t_d$ be the d terms in the lexicon. Then, for a document that belongs completely to topic T_j, the probability that the term t_l occurs in it is given by $P(t_l|T_j)$. The value of $P(t_l|T_j)$ is another important parameter which needs to be estimated by the topic modeling approach.

Note that the number of documents is denoted by n, topics by k and lexicon size (terms) by d. Most topic modeling methods attempt to learn the above parameters using maximum likelihood methods, so that the probabilistic fit to the given corpus of documents is as large as possible. There are two basic methods which are used for topic modeling, which are *Probabilistic Latent Semantic Indexing (PLSI)* [49] and *Latent Dirichlet Allocation (LDA)*[20] respectively.

In this section, we will focus on the probabilistic latent semantic indexing method. Note that the above set of random variables $P(T_j|D_i)$ and $P(t_l|T_j)$ allow us to model the probability of a term t_l occurring in any document D_i. Specifically, the probability $P(t_l|D_i)$ of the term t_l occurring document D_i can be expressed in terms of afore-mentioned parameters as follows:

$$P(t_l|D_i) = \sum_{j=1}^{k} p(t_l|T_j) \cdot P(T_j|D_i) \tag{4.12}$$

Thus, for each term t_l and document D_i, we can generate a $n \times d$ matrix of probabilities in terms of these parameters, where n is the number of documents and d is the number of terms. For a given corpus, we also have the $n \times d$ term-document occurrence matrix X, which tells us which term *actually* occurs in each document, and how many times the term occurs in the document. In other words, $X(i, l)$ is the number of times that term t_l occurs in document D_i. Therefore, we can use a maximum likelihood estimation algorithm which maximizes the product of the probabilities of terms that are observed in each document in the entire collection. The logarithm of this can be expressed as a weighted sum of the logarithm of the terms in Equation 4.12, where the weight of the (i, l)th term is its frequency count $X(i, l)$. This is a constrained optimization problem which optimizes the value of the log likelihood probability $\sum_{i,l} X(i, l) \cdot \log(P(t_l|D_i))$ subject to the constraints that the probability values over each of the topic-document and term-topic spaces must sum to 1:

$$\sum_{l} P(t_l|T_j) = 1 \quad \forall T_j \tag{4.13}$$

$$\sum_{j} P(T_j|D_i) = 1 \quad \forall D_i \tag{4.14}$$

The value of $P(t_l|D_i)$ in the objective function is expanded and expressed in terms of the model parameters with the use of Equation 4.12. We note that a Lagrangian method can be used to solve this constrained problem. This is quite similar to the approach that we discussed for the non-negative matrix factorization problem in this chapter. The Lagrangian solution essentially leads to a set of iterative update equations for the corresponding parameters which need to be estimated. It can be shown that these parameters can be estimated [49] with the iterative update of two matrices $[P_1]_{k\times n}$ and $[P_2]_{d\times k}$ containing the topic-document probabilities and term-topic probabilities respectively. We start off by initializing these matrices randomly, and normalize each of them so that the probability values in their columns sum to one. Then, we iteratively perform the following steps on each of P_1 and P_2 respectively:

```
for each entry $(j,i)$ in $P_1$ do update
  $P_1(j,i) \leftarrow P_1(j,i) \cdot \sum_{r=1}^{d} P_2(r,j) \cdot \frac{X(i,r)}{\sum_{v=1}^{k} P_1(v,i) \cdot P_2(r,v)}$
Normalize each column of $P_1$ to sum to 1;
for each entry $(l,j)$ in $P_2$ do update
  $P_2(l,j) \leftarrow P_2(l,j) \cdot \sum_{q=1}^{n} P_1(j,q) \cdot \frac{X(q,l)}{\sum_{v=1}^{k} P_1(v,q) \cdot P_2(l,v)}$
Normalize each column of $P_2$ to sum to 1;
```

The process is iterated to convergence. The output of this approach are the two matrices P_1 and P_2, the entries of which provide the topic-document and term-topic probabilities respectively.

The second well known method for topic modeling is that of *Latent Dirichlet Allocation*. In this method, the term-topic probabilities and topic-document probabilities are modeled with a Dirichlet distribution as a prior. Thus, the LDA method is the Bayesian version of the PLSI technique. It can also be shown the the PLSI method is equivalent to the LDA technique, when applied with a uniform Dirichlet prior [42].

The method of LDA was first introduced in [20]. Subsequently, it has generally been used much more extensively as compared to the PLSI method. Its main advantage over the PLSI method is that it is not quite as susceptible to overfitting. This is generally true of Bayesian methods which reduce the number of model parameters to be estimated, and therefore work much better for smaller data sets. Even for larger data sets, PLSI has the disadvantage that the number of model parameters grows linearly with the size of the collection. It has been argued [20] that the PLSI model is not a fully generative model, because there is no accurate way to model the topical distribution of a document which is not included in the current data set. For example, one can use the current set of topical distributions to perform the modeling of a new document, but it is likely to be much more inaccurate because of the overfitting inherent in PLSI. A Bayesian model, which uses a small number of parameters in the form of a well-chosen prior distribution, such as a *Dirichlet*, is likely to be much more robust in modeling new documents. Thus, the LDA method can also be used in order to model the topic distribution of a new document more robustly, even if it is not present in the original data set. Despite the theoretical advantages of LDA over PLSA, a recent study has shown that their task performances in clustering, categorization and retrieval tend to be similar [63]. The area of topic models is quite vast, and will be treated in more depth in Chapter 5 and Chapter 8 of this book; the purpose of this section is to simply acquaint the reader with the basics of this area and its natural connection to clustering.

We note that the EM-concepts which are used for topic modeling are quite general, and can be used for different variations on the text clustering tasks, such as text classification [72] or incorporating user feedback into clustering [46]. For example, the work in [72] uses an EM-approach in order to perform supervised clustering (and classification) of the documents, when a mixture of labeled and unlabeled data is available. A more detailed discussion is provided in Chapter 6 on text classification.

6. Online Clustering with Text Streams

The problem of streaming text clustering is particularly challenging in the context of text data because of the fact that the clusters need to be continuously maintained in real time. One of the earliest methods for streaming text clustering was proposed in [112]. This technique is referred to as the *Online Spherical k-Means Algorithm (OSKM)*, which reflects the broad approach used by the methodology. This technique divides up the incoming stream into small segments, each of which can be processed effectively in main memory. A set of k-means iterations are applied to each such data segment in order to cluster them. The advantage of using a segment-wise approach for clustering is that since each segment can be held in main memory, we can process each data point multiple times as long as it is held in main memory. In addition, the centroids from the previous segment are used in the next iteration for clustering purposes. A decay factor is introduced in order to age-out the old documents, so that the new documents are considered more important from a clustering perspective. This approach has been shown to be extremely effective in clustering massive text streams in [112].

A different method for clustering massive text and categorical data streams is discussed in [3]. The method discussed in [3] uses an approach which examines the relationship between outliers, emerging trends, and clusters in the underlying data. Old clusters may become inactive, and eventually get replaced by new clusters. Similarly, when newly arriving data points do not naturally fit in any particular cluster, these need to be initially classified as outliers. However, as time progresses, these new points may create a distinctive pattern of activity which can be recognized as a new cluster. The temporal locality of the data stream is manifested by these new clusters. For example, the first web page belonging to a particular category in a crawl may be recognized as an outlier, but may later form a cluster of documents of its own. On the other hand, the new outliers may not necessarily result in the formation of new clusters. Such outliers are true short-term abnormalities in the data since they do not result in the emergence of sustainable patterns. The approach discussed in [3] recognizes new clusters by first recognizing them as outliers. This approach works with the use of a summarization methodology, in which we use the concept of *condensed droplets* [3] in order to create concise representations of the underlying clusters.

As in the case of the OSKM algorithm, we ensure that recent data points are given greater importance than older data points. This is achieved by creating a time-sensitive weight for each data point. It is assumed that each data point has a time-dependent weight defined by the function $f(t)$. The function $f(t)$ is also referred to as the *fading function*. The fading function $f(t)$ is a non-monotonic decreasing function which decays uniformly with time t. The aim of defining a half life is to quantify the rate of decay of the importance of each data point in the stream clustering process. The *decay-rate* is defined as the inverse of the half life of the data stream. We denote the decay rate by $\lambda = 1/t_0$. We denote the weight function of each point in the data stream by $f(t) = 2^{-\lambda \cdot t}$. From the perspective of the clustering process, the weight of each data point is $f(t)$. It is easy to see that this decay function creates a half life of $1/\lambda$. It is also evident that by changing the value of λ, it is possible to change the rate at which the importance of the historical information in the data stream decays.

When a cluster is created during the streaming process by a newly arriving data point, it is allowed to remain as a trend-setting outlier for at least one half-life. During that period, if at least one more data point arrives, then the cluster becomes an active and mature cluster. On the other hand, if no new points arrive during a half-life, then the trend-setting outlier is recognized as a true anomaly in the data stream. At this point, this anomaly is removed from the list of current clusters. We refer to the process of removal as *cluster death*. Thus, a new cluster containing one data point dies when the (weighted) number of points in the cluster is 0.5. The same criterion is used to define the death of mature clusters. A necessary condition for this criterion to be met is that the inactivity period in the cluster has exceeded the half life $1/\lambda$. The greater the number of points in the cluster, the greater the level by which the inactivity period would need to exceed its half life in order to meet the criterion. This is a natural solution, since it is intuitively desirable to have stronger requirements (a longer inactivity period) for the death of a cluster containing a larger number of points.

The statistics of the data points are captured in summary statistics, which are referred to as *condensed droplets*. These represent the word distributions within a cluster, and can be used in order to compute the similarity of an incoming data point to the cluster. The overall algorithm proceeds as follows. At the beginning of algorithmic execution, we start with an empty set of clusters. As new data points arrive, unit clusters containing individual data points are created. Once a maximum number k of such clusters have been created, we can begin the process of online cluster maintenance. Thus, we initially start off with a trivial set of k clusters. These clusters are updated over time with the arrival of new data points.

When a new data point $\overline{X}$ arrives, its similarity to each cluster droplet is computed. In the case of text data sets, the cosine similarity measure between $\overline{DF1}$ and $\overline{X}$ is used. The similarity value $S(\overline{X}, \mathcal{C}_j)$ is computed from the incoming document $\overline{X}$ to every cluster $\mathcal{C}_j$. The cluster with the maximum value of $S(\overline{X}, \mathcal{C}_j)$ is chosen as the relevant cluster for data insertion. Let us assume that this cluster is $\mathcal{C}_{mindex}$. We use a threshold denoted by *thresh* in order to determine whether the incoming data point is an outlier. If the value of $S(\overline{X}, \mathcal{C}_{mindex})$ is larger than the threshold *thresh*, then the point $\overline{X}$ is assigned to the cluster $\mathcal{C}_{mindex}$. Otherwise, we check if some inactive cluster exists in the current set of cluster droplets. If no such inactive cluster exists, then the data point $\overline{X}$ is added to $\mathcal{C}_{mindex}$. On the other hand, when an inactive cluster does exist, a new cluster is created containing the solitary data point $\overline{X}$. This newly created cluster replaces the inactive cluster. We note that this new cluster is a potential true outlier or the beginning of a new trend of data points. Further understanding of this new cluster may only be obtained with the progress of the data stream.

In the event that $\overline{X}$ is inserted into the cluster $\mathcal{C}_{mindex}$, we update the statistics of the cluster in order to reflect the insertion of the data point and temporal decay statistics. Otherwise, we replace the most inactive cluster by a new cluster containing the solitary data point $\overline{X}$. In particular, the replaced cluster is the least recently updated cluster among all inactive clusters. This process is continuously performed over the life of the data stream, as new documents arrive over time. The work in [3] also presents a variety of other applications of the stream clustering technique such as evolution and correlation analysis.

A different way of utilizing the temporal evolution of text documents in the clustering process is described in [48]. Specifically, the work in [48] uses *bursty features* as markers of new topic occurrences in the data stream. This is because the semantics of an up-and-coming topic are often reflected in the frequent presence of a few distinctive words in the text stream. At a given period in time, the nature of relevant topics could lead to bursts in specific features of the data stream. Clearly, such features are extremely important from a clustering perspective. Therefore, the method discussed in [48] uses a new representation, which is referred to as the *bursty feature representation* for mining text streams. In this representation, a time-varying weight is associated with the features depending upon its burstiness. This also reflects the varying importance of the feature to the clustering process. Thus, it is important to remember that a particular document representation is dependent upon the particular instant in time at which it is constructed.

Another issue which is handled effectively in this approach is an implicit reduction in dimensionality of the underlying collection. Text is inherently a high dimensional data domain, and the pre-selection of some of the features on the basis of their burstiness can be a natural way to reduce the dimensionality of document representation. This can help in both the effectiveness and efficiency of the underlying algorithm.

The first step in the process is to identify the bursty features in the data stream. In order to achieve this goal, the approach uses Kleinberg's 2-state finite automaton model [57]. Once these features have been identified, the bursty features are associated with weights which depend upon their level of burstiness. Subsequently, a bursty feature representation is defined in order to reflect the underlying weight of the feature. Both the identification and the weight of the bursty feature are dependent upon its underlying frequency. A standard k-means approach is applied to the new representation in order to construct the clustering. It was shown in [48] that the approach of using burstiness improves the cluster quality. Once criticism of the work in [48] is that it is mostly focused on the issue of improving effectiveness with the use of temporal characteristics of the data stream, and does not address the issue of efficient clustering of the underlying data stream.

In general, it is evident that feature extraction is important for all clustering algorithms. While the work in [48] focuses on using temporal characteristics of the stream for feature extraction, the work in [61] focuses on using *phrase extraction* for effective feature selection. This work is also related to the concept of topic-modeling, which will be discussed in detail in the next section. This is because the different topics in a collection can be related to the clusters in a collection. The work in [61] uses topic-modeling techniques for clustering. The core idea in the work of [61] is that individual words are not very effective for a clustering algorithm because they miss the context in which the word is used. For example, the word "star" may either refer to a celestial body or to an entertainer. On the other hand, when the phrase "fixed star" is used, it becomes evident that the word "star" refers to a celestial body. The phrases which are extracted from the collection are also referred to as *topic signatures*.

The use of such phrasal clarification for improving the quality of the clustering is referred to as *semantic smoothing* because it reduces the noise which is associated with semantic ambiguity. Therefore, a key part of the approach is to extract phrases from the underlying data stream. After phrase extraction, the training process determines a translation probability of the phrase to terms in the vocabulary. For example, the word "planet" may have high probability of association with the phrase "fixed star", because both refer to celestial bodies. Therefore, for a given document, a rational probability count may also be assigned to all terms. For each document, it is assumed that all terms in it are generated either by a topic-signature model, or a background collection model.

The approach in [61] works by modeling the soft probability $p(w|C_j)$ for word w and cluster C_j. The probability $p(w|C_j)$ is modeled as a linear combination of two factors; (a) A maximum likelihood model which computes the probabilities of generating specific words for each cluster (b) An indirect (translated) word-membership probability which first determines the maximum likelihood probability for each topic-signature, and then multiplying with the conditional probability of each word, given the topic-signature. We note that we can use $p(w|C_j)$ in order to estimate $p(d|C_j)$ by using the product of the constituent words in the document. For this purpose, we use the frequency $f(w,d)$ of word w in document d.

$$p(d|C_j) = \prod_{w \in d} p(w|C_j)^{f(w,d)} \tag{4.15}$$

We note that in the static case, it is also possible to add a background model in order to improve the robustness of the estimation process. This is however not possible in a data stream because of the fact that the background collection model may require multiple passes in order to build effectively. The work in [61] maintains these probabilities in online fashion with the use of a *cluster profile*, that weights the probabilities with the use of a fading function. We note that the concept of cluster profile is analogous to the concept of condensed droplet introduced in [3]. The key algorithm (denoted by OCTS) is to maintain a dynamic set of clusters into which documents are progressively assigned with the use of similarity computations. It has been shown in [61] how the cluster profile can be used in order to efficiently compute $p(d|C_j)$ for each incoming document. This value is then used in order to determine the similarity of the documents to the different clusters. This is used in order to assign the documents to their closest cluster. We note that the methods in [3, 61] share a number of similarities in terms of (a) maintenance of cluster profiles, (b) use of cluster profiles (or condensed droplets) to compute similarity and assignment of documents to most similar clusters, and (c) the rules used to decide when a new singleton cluster should be created, or one of the older clusters should be replaced.

The main difference between the two algorithms is the technique which is used in order to compute cluster similarity. The OCTS algorithm uses the probabilistic computation $p(d|C_j)$ to compute cluster similarity, which takes the phrasal information into account during the computation process. One observation about OCTS is that it may allow for very similar clusters to co-exist in the current set. This reduces the space available for distinct cluster profiles. A second algorithm called OCTSM is also proposed in [61], which allows for merging of very similar clusters. Before each assignment, it checks whether pairs of similar clusters can be merged on the basis of similarity. If this is the case, then we allow the merging of the similar clusters and their corresponding cluster profiles. Detailed experimental results on the different clustering algorithms and their effectiveness are presented in [61].

A closely related area to clustering is that of topic modeling, which we discussed in an earlier section. Recently, the topic modeling method has also been extended to the *dynamic* case which is helpful for topic modeling of text streams [107].

7. Clustering Text in Networks

Many social networks contain both text content in the nodes, as well as links between the different nodes. Clearly, the links provide useful cues in understanding the related nodes in the network. The impact of different link types on the quality of the clustering has been studied in [109], and it has been shown that many forms of implicit and explicit links improve clustering quality, because they encode human knowledge. Therefore, a natural choice is to combine these two factors in the process of clustering the different nodes. In this section, we will discuss a number of such techniques.

In general, links may be considered as a kind of side-information, which can be represented in the form of attributes. A general approach for incorporating side attributes into the clustering process has been proposed in [1]. This algorithm uses a combination of a k-means approach on the text attributes, and Bayesian probability estimations on the side attributes for the clustering process. The idea is to identify those attributes, which are helpful for the clustering process, and use them in order to enhance the quality of the clustering. However, this approach is really designed for general attributes of any kind, rather than link-based attributes, in which an underlying graph structure is implied by the document-to-document linkages. In spite of this, it has been shown in [1], that it is possible to significantly enhance the quality of clustering by treating linkage information as side-attributes. Many other techniques, which will be discussed in this section, have been proposed specifically for the case of text documents, which are linked together in a network structure.

The earliest methods for combining text and link information for the clustering process are proposed in [12]. Two different methods were proposed in this paper for the clustering process. The first method uses the link information in the neighbors of a node in order to bias the term weights in a document. Term weights which are common between a document and its neighbors are given more importance in the clustering process. One advantage of such an approach is that we can use any of the existing clustering algorithms for this purpose, because the link information is implicitly encoded in the modified term weights. The second method proposed in [12] is a graph-based approach which directly uses the links in the clustering process. In this case, the approach attempts to model the probability that a particular document belongs to a given cluster for a particular set of links and content. This is essentially a *soft-clustering*, in which a probability of assignment is determined for each cluster. The cluster with the largest probability of assignment is considered the most relevant cluster. A Markov Random Field (MRF) technique is used in order to perform the clustering. An iterative technique called relaxation labeling is used in order to compute the maximum likelihood parameters of this MRF. More details of this approach may be found in [12].

A recent method to perform clustering with both structural and attribute similarities is proposed in [113]. The techniques of this paper can be applied to both relational and text attributes. This paper integrates structural and attribute-based clustering by adding attribute vertices to the network in addition to the original structural vertices. In the context of text data, this implies that a vertex exists for each word in the lexicon. Therefore, in addition to the original set of vertices V in the graph $G = (V, E)$, we now have the augmented vertex set $V \cup V_1$, such that V_1 contains one vertex for each nodes. We also augment the edge set, in order to add to the original set of structural edges E. We add an edge between a structural vertex $i \in V$ and an attribute vertex $j \in V_1$, if word j is contained in the node i. This new set of edges added is denoted by E_1. Therefore, we now have an augmented graph $G_1 = (V \cup V_1, E \cup E_1)$ which is semi-bipartite. A neighborhood random walk model is used in order to determine the closeness of vertices. This closeness measure is used in order to perform the clustering. The main challenge in the algorithm is to determine the relative importance of structural and attribute components in the clustering process. In the context of the random walk model, this translates to determining the appropriate weights of different edges during the random walk process. A learning model has been proposed in [113] in order to learn these weights, and leverage them for an effective clustering process.

The problem of clustering network content is often encountered in the context of community detection in social networks. The text content in the social network graph may be attached to either the nodes [101] of the network, or to the edges [74]. The node-based approach is generally more common, and most of the afore-mentioned techniques in this paper can be modeled in terms of content attached to the nodes. In the method proposed in [101], the following link-based and content-based steps are combined for effective community detection:

- A conditional model is proposed for link analysis, in which the conditional probability for the destination of given link is modeled. A hidden variable is introduced in order to capture the popularity of a node in terms of the likelihood of that node being cited by other nodes.

- A discriminative content model is introduced in order to reduce the impact of noisy content attributes. In this model, the attributes are weighed by their ability to discriminate between the different communities.

- The two models are combined into a unified framework with the use of a two-stage optimization algorithm for maximum likelihood inference. One interesting characteristic of this broad framework is that it can also be used in the context of other complementary approaches.

The details of the algorithm are discussed in [101].

For the case of edge-based community detection, it is assumed that the text content in the network is attached to the edges [74]. This is common in applications which involve extensive communication between the different nodes. For example, in email networks, or online chat networks, the text in the network is associated with the communications between the different entities. In such cases, the text is associated with an edge in the underlying network. The presence of content associated with edges allows for a much more nuanced approach in community detection, because a given node may participate in communities of different kinds. The presence of content associated with edges helps in separating out these different associations of the same individual to different communities. The work in [74] uses a matrix-factorization methodology in order to jointly model the content and structure for the community detection process. The matrix factorization method is used to transform the representation into multi-dimensional representation, which can be easily clustered by a simple algorithm such as the k-means algorithm. It was shown in [74], that the use of such an approach can provide much more effective results than a pure content- or link-based clustering methodology.

A closely related area to clustering is that of *topic modeling*, in which we attempt to model the probability of a document belonging to a particular cluster. A natural approach to network-based topic modeling is to add a network-based regularization constraint to traditional topic models such as NetPLSA [65]. The relational topic model (RTM) proposed in [23] tries to model the generation of documents and links sequentially. The first step for generating the documents is the same as LDA. Subsequently, the model predicts links based on the similarity of the topic mixture used in two documents. Thus, this method can be used both for topic modeling and predicting missing links. A more unified model is proposed in the *iTopicModel* [91] framework which creates a Markov Random Field model in order to create a generative model which simultaneously captures both text and links. Experimental results have shown this approach to be more general and superior to previously existing methods. A number of other methods for incorporating network information into topic modeling are discussed in the next chapter on dimensionality reduction.

8. Semi-Supervised Clustering

In some applications, prior knowledge may be available about the kinds of clusters that are available in the underlying data. This prior knowledge may take on the form of labels attached with the documents, which indicate its underlying topic. For example, if we wish to use the broad distribution of topics in the $Yahoo!$ taxonomy in order to supervise the clustering process of a new web collection, one way to performing supervision would be add some labeled pages from the $Yahoo!$ taxonomy to the collection. Typically such pages would contain labels of the form @*Science*@*Astronomy* or @*Arts*@*Painting*, which indicate the subject area of the added pages. Such knowledge can be very useful in creating significantly more coherent clusters, especially when the total number of clusters is large. The process of using such labels to guide the clustering process is referred to as *semi-supervised clustering*. This form of learning is a bridge between the clustering and classification problem, because it uses the underlying class structure, but it is not completely tied down by the specific structure. As a result, such an approach finds applicability both to the clustering and classification scenarios.

The most natural method for incorporating supervision into the clustering process is to do so in partitional clustering methods such as k-means. This is because the supervision can be easily incorporated by changing the seeds in the clustering process. For example, the work in [4] uses the initial seeds in the k-means clustering process as the centroids of the original classes in the underlying data. A similar approach has also been used in [15], except a wider variation of how the seeds may be selected has been explored.

A number of probabilistic frameworks have also been designed for semi-supervised clustering [72, 14]. The work in [72] uses an iterative EM-approach in which the unlabeled documents are assigned labels using a naive Bayes approach on the currently labeled documents. These newly labeled documents are then again used for re-training a Bayes classifier. This process is iterated to convergence. The iterative labeling approach in [72] can be considered a partially supervised approach for clustering the unlabeled documents. The work in [14] uses a Heterogeneous Markov Random Field (HMRF) model for the clustering process.

A graph-based method for incorporating prior knowledge into the clustering process has been proposed in [52]. In this method, the documents are modeled as a graph, in which nodes represent documents and edges represent the similarity among them. New edges may also be added to this graph, which correspond to the prior knowledge. Specifically, an edge is added to the graph, when it is known on the basis of prior knowledge that these two documents are similar. A normalized cut algorithm [84] is then applied to this graph in order to create the final clustering. This approach implicitly uses the prior knowledge because of the augmented graph representation which is used for the clustering.

Since semi-supervised clustering forms a natural bridge between the clustering and classification problems, it is natural that semi-supervised methods can be used for classification as well [68]. This is also referred to as *co-training*, because it involves the use of unsupervised document clustering in order to assist the training process. Since semi-supervised methods use both the clustering structure in the feature space and the class information, they are sometimes more robust in classification scenarios, especially in cases where the amount of available labeled data is small. It has been shown in [72], how a partially supervised co-training approach which mixes supervised and unsupervised data may yield more effective classification results, when the amount of training data available is small. The work in [72] uses a partially supervised EM-algorithm which iteratively assigns labels to the unlabeled documents and refines them over time as convergence is achieved. A number of similar methods along this spirit are proposed in [4, 14, 35, 47, 89] with varying levels of supervision in the clustering process. Partially supervised clustering methods are also used feature transformation in classification using the methods as discussed in [17, 18, 88]. The idea is that the clustering structure provides a compressed feature space, which capture the relevant classification structure very well, and can therefore be helpful for classification.

Partially supervised methods can also be used in conjunction with pre-existing categorical hierarchies (or prototype hierarchies) [4, 56, 67]. A typical example of a prototype hierarchy would be the *Yahoo!* taxonomy, the *Open Directory Project*, or the *Reuters collection*. The idea is that such hierarchies provide a good general idea of the clustering structure, but also have considerable noise and overlaps in them because of their typical manual origins. The partial supervision is able to correct the noise and overlaps, and this results in a relatively clean and coherent clustering structure.

An unusual kind of supervision for document clustering is the method of use of a *universum* of documents which are known *not* to belong to a cluster [106]. This is essentially, the background distribution which cannot be naturally clustered into any particular group. The intuition is that the universum of examples provide an effective way of avoiding mistakes in the clustering process, since it provides a background of examples to compare a cluster with.

9. Conclusions and Summary

In this chapter, we presented a survey of clustering algorithms for text data. A good clustering of text requires effective feature selection and a proper choice of the algorithm for the task at hand. Among the different classes of algorithms, the distance-based methods are among the most popular in a wide variety of applications.

In recent years, the main trend in research in this area has been in the context of two kinds of text data:

- **Dynamic Applications:** The large amounts of text data being created by dynamic applications such as social networks or online chat applications has created an immense need for streaming text clustering applications. Such streaming applications need to be applicable in the case of text which is not very clean, as is often the case for applications such as social networks.
- **Heterogeneous Applications:** Text applications increasingly arise in heterogeneous applications in which the text is available in the context of links, and other heterogeneous multimedia data. For example, in social networks such as *Flickr* the clustering often needs to be applied in such scenario. Therefore, it is critical to effectively adapt text-based algorithms to heterogeneous multimedia scenarios.

We note that the field of text clustering is too vast to cover comprehensively in a single chapter. Some methods such as *committee-based clustering* [73] cannot even be neatly incorporated into any class of methods, since they use a combination of the different clustering methods in order to create a final clustering result. The main purpose of this chapter is to provide a comprehensive overview of the main algorithms which are often used in the area, as a starting point for further study.

References

[1] C. C. Aggarwal, Y. Zhao, P. S. Yu. On Text Clustering with Side Information, *ICDE Conference*, 2012.

[2] C. C. Aggarwal, P. S. Yu. On Effective Conceptual Indexing and Similarity Search in Text, *ICDM Conference*, 2001.

[3] C. C. Aggarwal, P. S. Yu. A Framework for Clustering Massive Text and Categorical Data Streams, *SIAM Conference on Data Mining*, 2006.

[4] C. C. Aggarwal, S. C. Gates, P. S. Yu. On Using Partial Supervision for Text Categorization, *IEEE Transactions on Knowledge and Data Engineering*, 16(2), 245–255, 2004.

[5] C. C. Aggarwal, C. Procopiuc, J. Wolf, P. S. Yu, J.-S. Park. Fast Algorithms for Projected Clustering, *ACM SIGMOD Conference*, 1999.

[6] C. C. Aggarwal, P. S. Yu. Finding Generalized Projected Clusters in High Dimensional Spaces, *ACM SIGMOD Conference*, 2000.

[7] R. Agrawal, J. Gehrke, P. Raghavan. D. Gunopulos. Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications, *ACM SIGMOD Conference*, 1999.

[8] R. Agrawal, R. Srikant. Fast Algorithms for Mining Association Rules in Large Databases, *VLDB Conference*, 1994.

[9] J. Allan, R. Papka, V. Lavrenko. Online new event detection and tracking. *ACM SIGIR Conference*, 1998.

[10] P. Andritsos, P. Tsaparas, R. Miller, K. Sevcik. LIMBO: Scalable Clustering of Categorical Data. *EDBT Conference*, 2004.

[11] P. Anick, S. Vaithyanathan. Exploiting Clustering and Phrases for Context-Based Information Retrieval. *ACM SIGIR Conference*, 1997.

[12] R. Angelova, S. Siersdorfer. A neighborhood-based approach for clustering of linked document collections. *CIKM Conference*, 2006.

[13] R. A. Baeza-Yates, B. A. Ribeiro-Neto, *Modern Information Retrieval - the concepts and technology behind search, Second edition*, Pearson Education Ltd., Harlow, England, 2011.

[14] S. Basu, M. Bilenko, R. J. Mooney. A probabilistic framework for semi-supervised clustering. *ACM KDD Conference*, 2004.

[15] S. Basu, A. Banerjee, R. J. Mooney. Semi-supervised Clustering by Seeding. *ICML Conference*, 2002.

[16] F. Beil, M. Ester, X. Xu. Frequent term-based text clustering, *ACM KDD Conference*, 2002.

[17] L. Baker, A. McCallum. Distributional Clustering of Words for Text Classification, *ACM SIGIR Conference*, 1998.

[18] R. Bekkerman, R. El-Yaniv, Y. Winter, N. Tishby. On Feature Distributional Clustering for Text Categorization. *ACM SIGIR Conference*, 2001.

[19] D. Blei, J. Lafferty. Dynamic topic models. *ICML Conference*, 2006.

[20] D. Blei, A. Ng, M. Jordan. Latent Dirichlet allocation, *Journal of Machine Learning Research*, 3: pp. 993–1022, 2003.

[21] P. F. Brown, P. V. deSouza, R. L. Mercer, V. J. Della Pietra, and J/ C. Lai. Class-based n-gram models of natural language, *Computational Linguistics*, 18, 4 (December 1992), 467-479.

[22] K. Chakrabarti, S. Mehrotra. Local Dimension reduction: A new Approach to Indexing High Dimensional Spaces, *VLDB Conference*, 2000.

[23] J. Chang, D. Blei. Topic Models for Document Networks. *AISTASIS*, 2009.

[24] W. B. Croft. Clustering large files of documents using the single-link method. *Journal of the American Society of Information Science*, 28: pp. 341–344, 1977.

[25] D. Cutting, D. Karger, J. Pedersen, J. Tukey. Scatter/Gather: A Cluster-based Approach to Browsing Large Document Collections. *ACM SIGIR Conference*, 1992.

[26] D. Cutting, D. Karger, J. Pederson. Constant Interaction-time Scatter/Gather Browsing of Large Document Collections, *ACM SIGIR Conference*, 1993.

[27] M. Dash, H. Liu. Feature Selection for Clustering, *PAKDD Conference*, pp. 110–121, 1997.

[28] S. Deerwester, S. Dumais, T. Landauer, G. Furnas, R. Harshman. Indexing by Latent Semantic Analysis. *JASIS*, 41(6), pp. 391–407, 1990.

[29] I. Dhillon, D. Modha. Concept Decompositions for Large Sparse Data using Clustering, 42(1), pp. 143–175, 2001.

[30] I. Dhillon. Co-clustering Documents and Words using bipartite spectral graph partitioning, *ACM KDD Conference*, 2001.

[31] I. Dhillon, S. Mallela, D. Modha. Information-theoretic Co-Clustering, *ACM KDD Conference*, 2003.

[32] C. Ding, X. He, H. Zha, H. D. Simon. Adaptive Dimension Reduction for Clustering High Dimensional Data, *ICDM Conference*, 2002.

[33] C. Ding, X. He, H. Simon. On the equivalence of nonnegative matrix factorization and spectral clustering. *SDM Conference*, 2005.

[34] B. Dorow, D. Widdows. Discovering corpus-specific word senses, *Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics - Volume 2 (EACL '03)*, pages 79-82, 2003.

[35] R. El-Yaniv, O. Souroujon. Iterative Double Clustering for Unsupervised and Semi-supervised Learning. *NIPS Conference*, 2002.

[36] H. Fang, T. Tao, C. Zhai, A formal study of information retrieval heuristics, *Proceedings of ACM SIGIR 2004*, 2004.

[37] D. Fisher. Knowledge Acquisition via incremental conceptual clustering. *Machine Learning*, 2: pp. 139–172, 1987.

[38] M. Franz, T. Ward, J. McCarley, W.-J. Zhu. Unsupervised and supervised clustering for topic tracking. *ACM SIGIR Conference*, 2001.

[39] G. P. C. Fung, J. X. Yu, P. Yu, H. Lu. Parameter Free Bursty Events Detection in Text Streams, *VLDB Conference*, 2005.

[40] J. H. Gennari, P. Langley, D. Fisher. Models of incremental concept formation. *Journal of Artificial Intelligence*, 40 pp. 11–61, 1989.

[41] D. Gibson, J. Kleinberg, P. Raghavan. Clustering Categorical Data: An Approach Based on Dynamical Systems, *VLDB Conference*, 1998.

[42] M. Girolami, A Kaban. On the Equivalance between PLSI and LDA, *SIGIR Conference*, pp. 433–434, 2003.

[43] S. Guha, R. Rastogi, K. Shim. ROCK: a robust clustering algorithm for categorical attributes, *International Conference on Data Engineering*, 1999.

[44] S. Guha, R. Rastogi, K. Shim. CURE: An Efficient Clustering Algorithm for Large Databases. *ACM SIGMOD Conference*, 1998.

[45] D. Gusfield. Algorithms for strings, trees and sequences, *Cambridge University Press*, 1997.

[46] Y. Huang, T. Mitchell. Text clustering with extended user feedback. *ACM SIGIR Conference*, 2006.

[47] H. Li, K. Yamanishi. Document classification using a finite mixture model. *Annual Meeting of the Association for Computational Linguistics*, 1997.

[48] Q. He, K. Chang, E.-P. Lim, J. Zhang. Bursty feature representation for clustering text streams. *SDM Conference*, 2007.

[49] T. Hofmann. Probabilistic Latent Semantic Indexing. *ACM SIGIR Conference*, 1999.

[50] A. Jain, R. C. Dubes. Algorithms for Clustering Data, *Prentice Hall*, Englewood Cliffs, NJ, 1998.

[51] N. Jardine, C. J.van Rijsbergen. The use of hierarchical clustering in information retrieval, *Information Storage and Retrieval*, 7: pp. 217–240, 1971.

[52] X. Ji, W. Xu. Document clustering with prior knowledge. *ACM SIGIR Conference*, 2006.

[53] I. T. Jolliffee. Principal Component Analysis. *Springer*, 2002.

[54] L. Kaufman, P. J. Rousseeuw. Finding Groups in Data: An Introduction to Cluster Analysis, *Wiley Interscience*, 1990.

[55] W. Ke, C. Sugimoto, J. Mostafa. Dynamicity vs. effectiveness: studying online clustering for scatter/gather. *ACM SIGIR Conference*, 2009.

[56] H. Kim, S. Lee. A Semi-supervised document clustering technique for information organization, *CIKM Conference*, 2000.

[57] J. Kleinberg, Bursty and hierarchical structure in streams, *ACM KDD Conference*, pp. 91–101, 2002.

[58] D. D. Lee, H. S. Seung. Learning the parts of objects by non-negative matrix factorization, *Nature*, 401: pp. 788–791, 1999.

[59] T. Li, S. Ma, M. Ogihara, Document Clustering via Adaptive Subspace Iteration, *ACM SIGIR Conference*, 2004.

[60] T. Li, C. Ding, Y. Zhang, B. Shao. Knowledge transformation from word space to document space. *ACM SIGIR Conference*, 2008.

[61] Y.-B. Liu, J.-R. Cai, J. Yin, A. W.-C. Fu. Clustering Text Data Streams, *Journal of Computer Science and Technology*, Vol. 23(1), pp. 112–128, 2008.

[62] T. Liu, S. Lin, Z. Chen, W.-Y. Ma. An Evaluation on Feature Selection for Text Clustering, *ICML Conference*, 2003.

[63] Y. Lu, Q. Mei, C. Zhai. Investigating task performance of probabilistic topic models: an empirical study of PLSA and LDA, *Information Retrieval*, 14(2): 178-203 (2011).

[64] A. McCallum. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. `http://www.cs.cmu.edu/~mccallum/bow`, 1996.

[65] Q. Mei, D. Cai, D. Zhang, C.-X. Zhai. Topic Modeling with Network Regularization. *WWW Conference*, 2008.

[66] D. Metzler, S. T. Dumais, C. Meek, Similarity Measures for Short Segments of Text, *Proceedings of ECIR 2007*, 2007.

[67] Z. Ming, K. Wang, T.-S. Chua. Prototype hierarchy-based clustering for the categorization and navigation of web collections. *ACM SIGIR Conference*, 2010.

[68] T. M. Mitchell. The role of unlabeled data in supervised learning. *Proceedings of the Sixth International Colloquium on Cognitive Science*, 1999.

[69] F. Murtagh. A Survey of Recent Advances in Hierarchical Clustering Algorithms, *The Computer Journal*, 26(4), pp. 354–359, 1983.

[70] F. Murtagh. Complexities of Hierarchical Clustering Algorithms: State of the Art, *Computational Statistics Quarterly*, 1(2), pp. 101–113, 1984.

[71] R. Ng, J. Han. Efficient and Effective Clustering Methods for Spatial Data Mining. *VLDB Conference*, 1994.

[72] K. Nigam, A. McCallum, S. Thrun, T. Mitchell. Learning to classify text from labeled and unlabeled documents. *AAAI Conference*, 1998.

[73] P. Pantel, D. Lin. Document Clustering with Committees, *ACM SIGIR Conference*, 2002.

[74] G. Qi, C. Aggarwal, T. Huang. Community Detection with Edge Content in Social Media Networks, *ICDE Conference*, 2012.

[75] M. Rege, M. Dong, F. Fotouhi. Co-clustering Documents and Words Using Bipartite Isoperimetric Graph Partitioning. *ICDM Conference*, pp. 532–541, 2006.

[76] C. J. van Rijsbergen. *Information Retrieval*, Butterworths, 1975.

[77] C. J.van Rijsbergen, W. B. Croft. Document Clustering: An Evaluation of some experiments with the Cranfield 1400 collection, *Information Processing and Management*, 11, pp. 171–182, 1975.

[78] S. E. Robertson and S. Walker. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *SIGIR*, pages 232–241, 1994.

[79] M. Sahami, T. D. Heilman, A web-based kernel function for measuring the similarity of short text snippets, *Proceedings of WWW 2006*, pages 377-386, 2006.

[80] N. Sahoo, J. Callan, R. Krishnan, G. Duncan, R. Padman. Incremental Hierarchical Clustering of Text Documents, *ACM CIKM Conference*, 2006.

[81] G. Salton. An Introduction to Modern Information Retrieval, *Mc Graw Hill*, 1983.

[82] G. Salton, C. Buckley. Term Weighting Approaches in Automatic Text Retrieval, *Information Processing and Management*, 24(5), pp. 513–523, 1988.

[83] H. Schutze, C. Silverstein. Projections for Efficient Document Clustering, *ACM SIGIR Conference*, 1997.

[84] J. Shi, J. Malik. Normalized cuts and image segmentation. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 2000.

[85] C. Silverstein, J. Pedersen. Almost-constant time clustering of arbitrary corpus subsets. *ACM SIGIR Conference*, pp. 60–66, 1997.

[86] A. Singhal, C. Buckley, M. Mitra. Pivoted Document Length Normalization. *ACM SIGIR Conference*, pp. 21–29, 1996.

[87] N. Slonim, N. Tishby. Document Clustering using word clusters via the information bottleneck method, *ACM SIGIR Conference*, 2000.

[88] N. Slonim, N. Tishby. The power of word clusters for text classification. *European Colloquium on Information Retrieval Research (ECIR)*, 2001.

[89] N. Slonim, N. Friedman, N. Tishby. Unsupervised document classification using sequential information maximization. *ACM SIGIR Conference*, 2002.

[90] M. Steinbach, G. Karypis, V. Kumar. A Comparison of Document Clustering Techniques, *KDD Workshop on text mining*, 2000.

[91] Y. Sun, J. Han, J. Gao, Y. Yu. iTopicModel: Information Network Integrated Topic Modeling, *ICDM Conference*, 2009.

[92] E. M. Voorhees. Implementing Agglomerative Hierarchical Clustering for use in Information Retrieval,*Technical Report TR86–765, Cornell University, Ithaca, NY*, July 1986.

[93] F. Wang, C. Zhang, T. Li. Regularized clustering for documents. *ACM SIGIR Conference*, 2007.

[94] J. Wilbur, K. Sirotkin. The automatic identification of stopwords, *J. Inf. Sci.*, 18: pp. 45–55, 1992.

[95] P. Willett. Document Clustering using an inverted file approach. *Journal of Information Sciences*, 2: pp. 223–231, 1980.

[96] P. Willett. Recent Trends in Hierarchical Document Clustering: A Critical Review. *Information Processing and Management*, 24(5): pp. 577–597, 1988.

[97] W. Xu, X. Liu, Y. Gong. Document Clustering based on non-negative matrix factorization, *ACM SIGIR Conference*, 2003.

[98] W. Xu, Y. Gong. Document clustering by concept factorization. *ACM SIGIR Conference*, 2004.

[99] Y. Yang, J. O. Pederson. A comparative study on feature selection in text categorization, *ACM SIGIR Conference*, 1995.

[100] Y. Yang. Noise Reduction in a Statistical Approach to Text Categorization, *ACM SIGIR Conference*, 1995.

[101] T. Yang, R. Jin, Y. Chi, S. Zhu. Combining link and content for community detection: a discriminative approach. *ACM KDD Conference*, 2009.

[102] L. Yao, D. Mimno, A. McCallum. Efficient methods for topic model inference on streaming document collections, *ACM KDD Conference*, 2009.

[103] O. Zamir, O. Etzioni. Web Document Clustering: A Feasibility Demonstration, *ACM SIGIR Conference*, 1998.

[104] O. Zamir, O. Etzioni, O. Madani, R. M. Karp. Fast and Intuitive Clustering of Web Documents, *ACM KDD Conference*, 1997.

[105] C. Zhai, *Statistical Language Models for Information Retrieval (Synthesis Lectures on Human Language Technologies)*, Morgan & Claypool Publishers, 2008.

[106] D. Zhang, J. Wang, L. Si. Document clustering with universum. *ACM SIGIR Conference*, 2011.

[107] J. Zhang, Z. Ghahramani, Y. Yang. A probabilistic model for online document clustering with application to novelty detection. In *Saul L., Weiss Y., Bottou L. (eds) Advances in Neural Information Processing Letters*, 17, 2005.

[108] T. Zhang, R. Ramakrishnan, M. Livny. BIRCH: An Efficient Data Clustering Method for Very Large Databases. *ACM SIGMOD Conference,* 1996.

[109] X. Zhang, X. Hu, X. Zhou. A comparative evaluation of different link types on enhancing document clustering. *ACM SIGIR Conference*, 2008.

[110] Y. Zhao, G. Karypis. Evaluation of hierarchical clustering algorithms for document data set, *CIKM Conference*, 2002.

[111] Y. Zhao, G. Karypis. Empirical and Theoretical comparisons of selected criterion functions for document clustering, *Machine Learning*, 55(3), pp. 311–331, 2004.

[112] S. Zhong. Efficient Streaming Text Clustering. *Neural Networks*, Volume 18, Issue 5–6, 2005.

[113] Y. Zhou, H. Cheng, J. X. Yu. Graph Clustering based on Structural/Attribute Similarities, *VLDB Conference*, 2009.

[114] `http://www.lemurproject.org/`