Technical University of Crete
Department of Electronic and Computer Engineering

DESIGN AND EVALUATION OF CLUSTERING APPROACHES FOR LARGE DOCUMENT COLLECTIONS, THE "BIC-MEANS" METHOD

by

NIKOLAOS HOURDAKIS

A thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Computer Engineering

Chania, October 2006

Abstract

The availability of large volumes of digital content in modern applications (e.g., digital libraries and organization intranets) and the on internet has generated additional interest in methods and tools for effective management of shared content. Data clustering is a means for achieving better organization of the information by partitioning the data space into groups of entities with similar content. Clustering of large document collections is the problem this thesis is dealing with. State of the art clustering algorithms are reviewed first (e.g. partitional and hierarchical algorithms).

Initially, we focus on partitional clustering methods due to their low time complexity (i.e., linear on the number of documents). Hierarchical clustering methods are considered as well. We examine several variants of the original K-Means algorithm and we propose the so-called "Incremental K-Means" which differs from K-Means in the way the centroids are updated during each clustering iteration. However, both K-Means and its variants produce a flat partition of the data collection. Efficient methods which are able to provide effective organization of information (like hierarchical clustering) are preferred.

We propose a novel hierarchical clustering approach, which we call "BIC-Means". BIC-Means produces a hierarchy of clusters by recursively applying the Incremental K-Means on a document collection. BIC-Means combines the strengths of partitional and hierarchical clustering methods. The main advantage of BIC-Means is that it does not terminate when singleton clusters are reached at the bottom of the hierarchy. To prevent over-splitting of clusters, BIC-Means incorporates the use of the Bayesian Information Criterion (BIC) or Schwarz Criterion for terminating the splitting of the hierarchy when meaningful clusters are reached. We use BIC to perform a splitting test at each leaf cluster in order to decide whether a cluster should be split or not. BIC-Means terminates when there is no separable cluster according to the BIC function.

We run several sets of experiments on two TREC standard document collections (Reuters and OHSUMED). Our experimental results show that the main advantage of BIC-Means is that it requires significantly less time to build a cluster hierarchy than the standard Bisecting K-Means algorithm (BIC-Means does not have to reach singleton clusters at the leafs). In terms of clustering quality, BIC-Means

achieves approximately the same performance as the basic Bisecting technique (the exhaustive approach). Therefore, BIC-Means is more suitable than its competitors for clustering very large document collections effectively. This is not only due to its low computational requirements, but also due to its comparable clustering performance.

We also explore Medical Subject Headings (MeSH) as features for representing medical documents i.e., each document is represented as a vector of MeSH terms (multi-word terms) rather than as vectors of single-word terms. Our evaluation shows that MeSH-based representation of documents improves significantly the performance of BIC-Means in terms of clustering time and clustering quality.

Finally, we examine how hierarchical clustering could be used to improve the effectiveness and efficiency of retrieval from large medical document collections. We propose several cluster-based retrieval strategies using MeSH terms as document representation. Experimental results show that the best proposed cluster-based retrieval strategy is almost as effective as exhaustive searching (i.e. searching without clustering). Cluster-based retrieval not only saves a huge amount of computation but does so without significant loss in precision and recall.

Περίληψη

Ο συνεχώς αυξανόμενος όγκος της ψηφιακής πληροφορίας σε παλιές και νέες εφαρμογές (ψηφιακές βιβλιοθήκες, εσωτερικά δίκτυα οργανισμών κ.α.) και στο διαδίκτυο έχουν αυξήσει σημαντικά το ενδιαφέρον για μεθόδους που επιτυγχάνουν την οργάνωση της πληροφορίας στα μέσα αποθήκευσης όπως είναι μέθοδοι «ομαδοποίησης» (clustering). Τα δεδομένα οργανώνονται σε μικρό αριθμό ομάδων (clusters) όπου κάθε ομάδα περιέχει παρόμοιο πληροφοριακό περιεχόμενο. Η ομαδοποίηση σε πολύ μεγάλες συλλογές κειμένων (document clustering) είναι το βασικό θέμα της συγκεκριμένης εργασίας.

Αρχικά κάνουμε μια ανασκόπηση των πιο γνωστών μεθόδων ομαδοποίησης που έχουν παρουσιαστεί στη βιβλιογραφία. Εστιάζουμε σε «διαμεριστικούς» (partitional) αλγόριθμούς ομαδοποίησης κειμένων (partitional clustering) εξαιτίας της χαμηλής (γραμμικής) πολυπλοκότητάς τους ως προς τον αριθμό των κειμένων αλλά και της καλής απόδοσης που έχουν επιδείξει. Ωστόσο, για την ομαδοποίηση συλλογών κειμένων προτιμούνται μέθοδοι οι οποίες παρέχουν αποτελεσματική πλοήγηση, οργάνωση και απεικόνιση της πληροφορίας όπως οι ιεραρχικοί μέθοδοι (hierarchical clustering). Οι περισσότερες γνωστές ιεραρχικές μέθοδοι, αν και ακριβείς έχουν τετραγωνική πολυπλοκότητα και για αυτό το λόγο δεν εφαρμόζονται σε μεγάλες συλλογές κειμένων.

Η μέθοδος Κ-μέσων (K-Means) και οι παραλλαγές του παράγουν ένα επίπεδο διαμερισμό των δεδομένων (flat partitioning). Εξετάζουμε διάφορες παραλλαγές του κλασσικού αλγορίθμου των Κ-μέσων και προτείνουμε μία παραλλαγή του, την «Επαυξητική μέθοδο Κ-μέσων» (Incremental K-Means) η οποία ανανεώνει το κέντρο (centroid) μιας ομάδας μόλις ένα κείμενο προστεθεί σε αυτόν. Προτείνουμε επίσης μία νέα ιεραρχική μέθοδο που ονομάζουμε “BIC-Means”. Η μέθοδος BIC-Means παράγει μια ιεραρχία από ομάδες δεδομένων (κειμένων στην περίπτωσή μας) εφαρμόζοντας επαναληπτικά την επαυξητική μέθοδο Κ-μέσων σε μια συλλογή κειμένων. Η μέθοδος συνδυάζει τα πλεονεκτήματα των επίπεδων και ιεραρχικών τεχνικών δηλαδή, είναι ιεραρχική μέθοδος ενώ είναι πιο γρήγορη από την επαυξητική μέθοδο. Αυτό οφείλεται στο ότι η μέθοδος BIC-Means δεν είναι εξαντλητική, δηλαδή δεν χρειάζεται να τερματίσει όταν οι ομάδες περιέχουν ένα μόνο κείμενο (singleton clusters). Για να επιτευχθεί αυτό, ο BIC-Means ενσωματώνει το Bayesian

Information Criterion (BIC) ή Schwarz Criterion. Το κριτήριο αυτό εφαρμόζεται για να σταματήσει τις διασπάσεις των ομάδων σε ανώτερα επίπεδα της ιεραρχίας όταν η περαιτέρω διάσπασή τους δεν θα οδηγήσει σε καλύτερη ομαδοποίηση. Ο BIC-Means τερματίζει όταν έχουν εξεταστεί όλες οι υποψήφιες προς διάσπαση ομάδες (ομάδες που βρίσκονται στα φύλλα της ιεραρχίας) και δεν υπάρχει άλλη υποψήφια ομάδα για διάσπαση.

Για την αξιολόγηση της απόδοσης των αλγορίθμων που αναπτύξαμε κάναμε ένα σύνολο πειραμάτων σε δύο πολύ διαδεδομένες συλλογές κειμένων (Reuters, OHSUMED). Σύμφωνα με τα πειραματικά αποτελέσματα, το βασικό πλεονέκτημα του BIC-Means είναι ότι χρειάζεται πολύ λιγότερο χρόνο (εν σε σχέση με τον βασικό επαυξητικό αλγόριθμο K-μέσων) για να δημιουργήσει μια ιεραρχία από ομάδες ενώ αποδίδει το ίδιο καλά με την επαυξητική μέθοδο που προτείναμε (η οποία είναι ήδη πολύ αποτελεσματική επιτυγχάνοντας ακρίβεια ομαδοποίησης πάνω από 75% εν σχέση με μία ομαδοποίηση που παράγουν ειδικοί χρήστες). Επομένως, ο BIC-Means, συγκρινόμενος με τις γνωστές μεθόδους ομαδοποίησης είναι εξίσου ακριβής και μπορεί να εφαρμοστεί για την ιεραρχική ομαδοποίηση πολύ μεγάλων συλλογών κειμένων. Παράλληλα, εξετάζουμε την χρήση ειδικών ιατρικών όρων (από την MeSH ταξινομική ιεραρχία) για την αναπαράσταση ιατρικών κειμένων. Με αυτόν τον τρόπο κάθε κείμενο αναπαριστάται με διανύσματα πολυ-λεκτικών MeSH όρων (multi-word terms) αντί με διανύσματα απλών μονο-λεκτικών όρων (single-word terms). Οι παραστάσεις αυτές περιγράφουν καλύτερα το ιατρικό περιεχόμενο των κειμένων σε ιατρικές εφαρμογές (π.χ. κείμενα της συλλογής OHSUMED) και είναι πιο συμπαγείς (περιέχουν λιγότερους όρους). Τα αποτελέσματα των πειραμάτων έδειξαν, ότι η παράσταση των κειμένων με MeSH όρους βελτιώνει σημαντικά την απόδοση του BIC-Means, τόσο σε σχέση με την ποιότητα της ομαδοποίησης όσο και με τον χρόνο που απαιτείται.

Ολοκληρώνοντας, εξετάζουμε πώς μία ιεραρχική ομαδοποίηση (που έχει παραχθεί με μία μέθοδος όπως η BIC-Means) μπορεί να χρησιμοποιηθεί για την γρηγορότερη ανάκτηση πληροφορίας (information retrieval) σε μεγάλες ιατρικές συλλογές κειμένων. Προτείνουμε μια σειρά από στρατηγικές ανάκτησης που κάνουν χρήση των δεδομένων της ιεραρχίας. Τα πειραματικά αποτελέσματα έδειξαν ότι η καλύτερη από τις προτεινόμενες στρατηγικές ανάκτησης αποδίδει το ίδιο καλά με την εξαντλητική μέθοδο ανάκτησης (χωρίς χρήση ομαδοποίησης) δηλαδή, μειώνει κατά

πολύ τον απαιτούμενο υπολογιστικό χρόνο, χωρίς να προκαλεί μείωση της απόδοσης της ανάκτησης.

This dissertation is dedicated to my father Giorgos,

my mother Voula and my brother Vaggelis…

Acknowledgements

This dissertation could not have been completed without the help of many people. I am foremost grateful to my supervisor, Professor Euripides Petrakis, for always being available, for his invaluable advice, his tireless support, the encouragement and the confidence he showed to me and to my work. His comments and corrections were always up to the point showing me the way.

I am also very thankful to the rest members of the supervisory committee, Professors Michail Lagoudakis and Vassilis Samoladas for valuable contributions to this work. They got into deep questions and with their comments helped me to improve the content of this dissertation.

Special thanks are given to Professor Evangelos Milios (Computer Science Department, Dalhousie University) for his critical analysis and recommendations and for being an outstanding mentor since my initial research steps.

I feel grateful to my colleagues in the Intelligent Systems Laboratory of Technical University of Crete. Special thanks are given to Angelos Hliaoutakis, Epimenidis Voutsakis, Paraskevi Raftopoulou and Giannis Varelas for their collaboration and valuable comments and the harmonic and enjoyable coexistence they provided. I wish for all of them the best in their life.

I would also like to thank my friends who offered me ungrudgingly their friend-ship. I want to profoundly thank Patra Papadaki for her encouragement and patience during this thesis. She always was there whenever I needed her.

Above all, my deepest thanks go to my parents Giorgos and Voula, and my brother Vaggelis for their endless love and encouragement in all aspects of my life.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

In recent years, we have seen a tremendous explosion of electronic information available on the Internet, digital libraries and organizational intranets. Large collections of documents are becoming increasingly common and widely available to the public. On the other hand, the World Wide Web (WWW) continues to expand at an amazing rate. Due to the huge size of document collections, searching for information in such collections has become a very challenging task.

Document or text clustering plays an important role toward this goal. It refers to the process of automatic grouping of text documents into clusters, so that each cluster consists of similar documents (documents in different clusters are dissimilar). Document clustering is the fundamental tool for enabling efficient document summarization, organization, and navigation in very large data sets. It provides the infrastructure for developing tools supporting navigation and browsing mechanisms by organizing enormous amounts of documents into meaningful clusters.

Document clustering has been widely applied in various scientific fields for supporting search engines, text mining, and Information Retrieval [68]. It has been used also as a post-retrieval tool for organizing query results into thematic topics. These organized results can be interactively browsed, visualized, and explored by the users.

Text Clustering is an unsupervised learning process of grouping documents into clusters. There are no pre-defined classes available in document clustering and this is how text clustering differs from text classification. In text classification, we are provided with a training set of labeled documents and we are asked to assign to one of new, yet unlabeled documents the pre-defined categories [67]. Thus, text categorization is a supervised learning task.

1.1. Motivation

Document clustering has been extensively studied in the literature and a variety of algorithms have been proposed [17], [23], [24], [30], [34]. These algorithms can be categorized along different dimensions. First, clustering can be either static or dynamic. Static clustering algorithms usually refer to static document collections. On the other hand, dynamic clustering is applied on data sets that change dynamically. Consider for example the flow of information that arrives continuously on news wires message systems such as Reuters, Marketwatch, etc. In this case the clusters must adapt to the incoming flow or deletions of documents. Dynamic clustering has not been widely studied, while static clustering methods can be further improved.

Based on the nature of the membership function the clustering can be either hard or soft (fuzzy). Hard clustering algorithms produce hard clusters (i.e., each document is assigned to a single cluster) while in soft clustering, documents may be instances of more than one cluster. Notice that a fuzzy clustering can be converted to a hard clustering by assigning each data object to the closest cluster. In this thesis, we focus on static, hard clustering algorithms.

Based on the underlying algorithmic methodology, the standard clustering algorithms can be categorized into hierarchical [28], [31], [54], [55], [68] and partitional [9], [20], [24], [40]. Hierarchical clustering algorithms proceeds either bottom-up (agglomerative), or top-down (divisive). Hierarchical Agglomerative Clustering (HAC) starts with all documents as individual clusters and works by merging the most similar ones iteratively until a single cluster with all documents is produced at the root of the hierarchy. Divisive algorithm approaches start with all documents in the same root cluster and work by iteratively splitting each cluster into a number of smaller ones until clusters with one document (singleton clusters) are produced at the leafs of the hierarchy. Both types of methods produce a tree hierarchy of clusters called a “dendrogram”. Contrary to hierarchical clustering techniques, partitional algorithms create a flat (un-nested) partitioning of documents. K-Means is a widely used partitional clustering method. It partitions the entire collection into K clusters, where K stands for the desired number of output clusters and must be known in advanced.

In recent years, various experimental results [1], [6], [32], [55] have indicated that partitional clustering algorithms are well-suited for clustering large data sets due

to their low computational requirements (linear in the number of documents) [1], [6]. The time complexity of most hierarchical clustering methods is quadratic in the number of documents. Due to the large size of document collections in modern applications and the explosion of the WWW there is an increased need for effective clustering, which scales-up well for very large data sets. Hierarchical clustering provides the infrastructure for developing effective navigation, organization, and visualization tools for such large amounts of information. For this reason, hierarchical clustering solutions are preferred. However, traditional hierarchical clustering algorithms have limited applicability in large document collections due to their quadratic time complexity. Furthermore, partitional techniques usually lead to better clustering solutions than agglomerative algorithms [55], [66].

Partitional clustering methods can also be used to obtain a hierarchical clustering solution via a sequence of repeated application of the K-Means algorithm. Bisecting K-Means method is such an approach. The method starts with all documents in a single cluster. Initially, this cluster is partitioned into two clusters by applying K-Means. The algorithm continues by splitting similarly each produced cluster until singleton clusters are obtained at the leafs or until K clusters have been produced. The so-obtained clusters are structured as a hierarchical binary tree. The overall hierarchy is built in $O(n \log n)$ time (in case of a balanced hierarchy), where n is the number of documents.

An important issue in divisive clustering approaches is to determine a strategy to terminate the divisive procedure. Without prior knowledge on the number of clusters the algorithm executes exhaustively. In the case of large document collections the efficiency of clustering decreases significantly. For this reason, additional termination criteria must be introduced to increase the efficiency of the algorithm and prevent it from over-partitioning. This is exactly one of the problems this work is dealing with. Notice that, without a termination criterion, even meaningful clusters (i.e., clusters corresponding to real classes) are further split until singleton clusters are reached at the leafs of the tree hierarchy.

1.2. Contributions

In this thesis, our main objective is to develop a highly efficient algorithm for clustering very large document collections. We focus on partitional clustering methods due to their low time complexity. We implemented several partitional clustering algorithms and we studied their performance.

Initially, we examined the standard K-Means clustering approach. We developed several variants of the original K-Means method and we proposed the so-called "Incremental K-Means". Incremental K-Means differs from basic K-Means in the way the centroids are updated during each clustering iteration. In Incremental K-Means each cluster centroid is updated after each document is assigned to a cluster (rather than re-computing each cluster centroid after each iteration when all documents have been assigned to clusters).

We investigate how Incremental K-Means can be used effectively to build a hierarchy of clusters. A hierarchical solution is obtained by recursively applying the Incremental K-Means on a document collection. All documents are initially partitioned into two clusters. Then, the least cohesive leaf cluster is selected for further splitting. This process of selecting and bisecting a leaf cluster continues until all clusters at the bottom of the hierarchy contain a single document. We call the proposed algorithm "Bisecting Incremental K-Means". As indicated in [55], [66] the basic Bisecting approach significantly outperforms agglomerative hierarchical clustering algorithm in terms of clustering quality and efficiency. Thus, we focus our research on Bisecting Incremental K-Means.

As mentioned in Section 1.1, despite its effectiveness, the main disadvantage of Bisecting Incremental K-Means is that it terminates when each leaf cluster contains a single document which is not only slow but also produces lots of meaningless clusters at the bottom of the hierarchy (e.g., singleton clusters). The produced clustering result is not appropriate for navigation, data summarization and browsing of information. For this, a terminating condition must be defined.

We propose a novel hierarchical clustering approach which incorporates the use of the "Bayesian Information Criterion (BIC)" or Schwarz Criterion [52] for terminating the splitting of the Bisecting Incremental K-Means algorithm. We suggest using BIC as the splitting criterion of a cluster. BIC estimates the cohesiveness of clusters in order to denote whether a cluster should split. If the BIC score of the

produced children clusters is less than the BIC score of the parent cluster we do not accept the split and keep the parent cluster as is. We terminate the divisive procedure when there is no separable leaf cluster according to the BIC function. Similarly to our approach X-Means [42], which is a variant of K-Means, first used BIC to estimate the best K in a given range of values. Notice that X-Means is a partitional clustering algorithm.

Overall, we propose the so-called here-after "BIC-Means" clustering algorithm which is the main contribution of this thesis. It produces a hierarchical clustering solution and combines all the following ideas:

1. Bisecting clustering approach to build a hierarchy of clusters effectively.
2. Incremental K-Means as the proposed partitional method to bisect the selected leaf cluster at each bisecting step.
3. A termination criterion from preventing clustering from over-splitting based on Bayesian Information Criterion (BIC).

The proposed BIC-Means terminates before each leaf cluster becomes a single document. As a result, the obtained clusters are more meaningful as compared to meaningless singleton clusters of standard hierarchical algorithms. The proposed algorithm combines the strengths of partitional and hierarchical clustering methods.

We focus on evaluating the performance of the proposed clustering algorithms in terms of clustering quality and time required to obtain clustering solutions. We used two standard document collections (OHSUMED and Reuters). F-Measure was used to examine the quality of the produced clustering results. It measures the performance of clustering methods in terms of how well the documents belonging to each of the pre-defined classes match the documents belonging to the corresponding cluster.

Experimental results indicated that Bisecting Incremental K-Means performs significantly better than K-Means and (our proposed variant) Incremental K-Means in terms of F-Measure on both test collections. We also observed that Incremental K-Means always produces better partitional clustering solutions than standard K-Means. We also explored Medical Subject Headings (MeSH) [27], a controlled vocabulary for describing medical literature, as features for representing medical documents in

OHSUMED i.e., each document is represented as a vector of MeSH terms (multi-word composite terms) rather than by vectors of single word terms (the state of the art approach). This leads to a more compact representation (each vector contains less terms) which is directly perceivable by humans. Our evaluation showed that MeSH-based representation of documents improves noticeably the performance of Bisecting Incremental K-Means with respect to clustering time and clustering quality.

The performance of our proposed BIC-Means algorithm is evaluated and compared against the performance of hierarchical clustering methods such as Bisecting Incremental K-Means. Experimental results indicated that the main advantage of BIC-Means is that requires significantly less time to build a cluster hierarchy than Bisecting Incremental K-Means (is executed exhaustively). In terms of clustering quality, BIC-Means performs approximately the same as our initial Bisecting approach. Therefore, the proposed BIC-Means is well-suited for obtaining effective hierarchical clustering solutions of large data sets. This is not only due to its low computational requirements, but also comparable performance. Notice that, BIC-Means terminates at meaningful clusters (clusters which are likely to correspond to real classes).

Having established the quality of the implemented document clustering algorithms, we examine how hierarchical clustering could be used to improve the effectiveness and efficiency of retrievals on large medical document collections. Our main goal is to noticeably reduce the number of required similarity computations between the user's query and documents within a collection. We produce a hierarchy of clusters using the BIC-Means. We propose and evaluate several cluster-based strategies for searching hierarchical clustered document collection based on the idea that only leaf clusters need to be searched (intermediate level clusters combine information from lower level leaf clusters). We retrieve the documents contained in the N top-ranked clusters. Notice that, we use MeSH terms to build the document, cluster and query vectors.

The experimental results indicated that among all cluster-based retrieval strategies proposed in this thesis the best results on OHSUMED are obtained in case we examine only the leaf clusters which contain all the MeSH terms of the query in their centroid vectors. Contrary to exhaustive search (233,445 documents are searched), the proposed cluster-based retrieval strategy search only 30% of OHSUMED documents. Experiments also showed that the proposed search strategy is

almost as effective as the retrieval by exhaustive search on OHSUMED. Summarizing, this cluster-based retrieval not only saves a huge amount of computation, but does so without loss of retrieval effectiveness.

1.3. Thesis Structure

The rest of this thesis is organized as follows.

Chapter 2 provides a review of related work in the fields of document clustering, evaluation methodology of clustering quality and stopping criteria on hierarchical clustering.

Chapter 3 describes in detail our proposed clustering algorithms for efficient clustering of large document collections. We discuss techniques to improve the quality of the obtained clustering results.

Chapter 4 presents the two sets of experiments that we performed for evaluating our proposed methodology. The first one focuses on evaluating the quality of the clustering solutions produced by the several proposed clustering algorithms. The second set of experiments examined how hierarchical clustering could be used to improve the effectiveness and efficiency of retrieval by exhaustive search on large document collections.

Chapter 5 summarizes the achievements of this thesis and points out possible directions for future research.

Appendix A talks in detail about many parts that took place in the implementation process and describes some technical issues about the developed software.

Chapter 2

Background and Related Work

In this chapter, we provide an overview of research related to document clustering. We highlight the key issues in the area of Information Retrieval (IR), presenting the three common Information Retrieval Models. We describe the main text clustering techniques and we explain some technical issues. Then, we present the basic measures of cluster quality. Finally, we focus on stopping criteria in hierarchical document clustering algorithms, emphasizing on the so-called Bayesian Information Criterion (BIC). We conclude by describing MeSH control vocabulary which we use in our experimental evaluation.

2.1 Information Retrieval

The internet is expanding at increasing rate, and search for information is becoming more difficult in this gigantic digital library. This fact calls for improved automatic methods for searching and organizing documents so requested information can be accessed quickly and accurately. The term Information Retrieval (IR) defines all those activities that can be used to retrieve documents of interest from a given collection of documents.

2.1.1 Information Retrieval Models

The three classic models in IR [2] are known as Vector Space, Boolean, and Probabilistic. In the Vector Space model [50], documents and queries are represented by vectors in a multi-dimensional space, where each dimension corresponds to a unique word in corpus. Thus, we say that the model is algebraic. In the Boolean model, documents and queries are represented as a set of index terms, thus this model is set theoretic. Finally, in the Probabilistic model the representation of documents and queries is based on probabilities of occurrence of terms in a corpus. Among the

three models, vector space is the most commonly used. The various clustering algorithms that are described and implemented in this thesis are based on the vector space model.

2.1.2 Vector Space Model

The Vector Space model (VSM) [51] is the most popular IR model. It has been shown to perform at least as good as the other two models. In VSM, both documents d and queries q are considered to be vectors in a multidimensional term space. The VSM assigns to the terms non-binary weights which are used to compute the degree of similarity between a document and a query or between two documents.

The weights assigned to each term can be either the term frequency (tf) or term frequency-inverse document frequency ($tf-idf$) [32]. In first case, the frequency of occurrence for a term in a document is included in the vector $d_{tf}=(tf_1,tf_2,....,tf_m)$, where tf_i is the frequency of the i^{th} term in the document. Usually, very common words are removed and the terms are stemmed. A refinement to this weighting scheme is the so-called $tf-idf$ weighting scheme. In this approach, a term that appears in many documents should not be regarded as more important than the one that appears in few documents, and for this reason it needs to be de-emphasized.

Let N be the total number of documents in the collection; df_i (document frequency) be the number of documents in which the k_i term appears, and $freq_{i,j}$ be the raw frequency of the term k_i in the document d_j. The inverse document frequency (idf_i) for k_i is defined as:

$$idf_i = \log \frac{N}{df_i} \tag{2.1}$$

The $tf-idf$ weight of term i is computed by:

$$w_{ij} = freq_{ij} \times \log \frac{N}{df_i} \tag{2.2}$$

To account for documents of different length, each vector is normalized so that it is of unit length.

There are many variations of this basic formula. The one we use in our implemented algorithms is described in section 3.2.1.

Similarity Computation

The Vector Space Model computes the degree of similarity between a document d_j and a query q (or between two documents). The similarity between two documents is computed as the cosine of the angle between their document vectors in the multidimensional term space [50]:

$$sim\left(d_i, d_j\right) = \cos\left(\overrightarrow{d_i}, \overrightarrow{d_j}\right) = \frac{\vec{d}_i \cdot \vec{d}_j}{\left|\vec{d}_i\right| \times \left|\vec{d}_j\right|} = \frac{\sum_t w_{it} \times w_{jt}}{\sqrt{\sum_t w_{it}^2 \sum_t w_{jt}^2}} \tag{2.3}$$

This measure simplifies to $\cos\left(d_i, d_j\right) = d_i^T d_j$ due to the unit length of document vectors. All vectors are normalized by document length. The measure takes values between 1 (the two documents are identical) and 0 (the two documents have no common terms).

The main advantages of Vector Space Model (VSM) are [57]:

- The documents are sorted by decreasing similarity with the query q.
- The terms are weighted by importance.
- It allows for partial matching: the documents need not have exactly the same terms with the query.

One disadvantage of VSM is that the terms are assumed to be independent. Moreover, weighting is intuitive and not very formal.

2.1.3 Boolean Model

The Boolean Model [2] is the most simple among the three models and relies on the use of Boolean operators and set theory. The terms in a query are combined together with *AND*, *OR* and *NOT* operators. A document is predicted as relevant to a query expression if it satisfies the query Boolean expression. Each term is either present (1) or absent (0). The basic advantage of Boolean model is that is very simple (based on set theory). It is easy to understand and implement.

The Boolean model also has its drawbacks. It only retrieves exact matching (a retrieved document contains exactly the same terms with the query). The retrieved documents are all equally ranked with respect to relevance. Furthermore, all terms are equally important. Boolean operator usage has much more influence than a critical word. Also query language is expressive but complicated due to complex Boolean expressions. The Boolean retrieval model has been extended and refined to solve these problems [51]. Expanded term weighting operations make ranking of documents possible, where the terms in the document could be weighted according to their frequency in the document.

2.1.4 Probabilistic Model

The Probabilistic Retrieval Model [2], [15] computes the probability that a document is similar to the query. It assumes that for each document an ideal answer set of similar documents exists for each query. Given a query q, a subset of documents, R is relevant to q. The probability that a specific document will be judged relevant to a specific query is based on the assumption that the terms are distributed differently in relevant and non-relevant documents. The weights take binary values (a term exists in a document or not). In general, the Probabilistic model attempts to answer a basic question: "*What is the probability that this document is relevant to this query?*".

If retrieved documents are ordered by decreasing probability of relevance on the data available, then the system's effectiveness is the best that is obtainable on the basis of those data (Probability Ranking Principle) [48]. Moreover, relevance feedback can improve the ranking by giving better term probability estimates.

In conclusion, Probabilistic Model uses probability theory to model the uncertainty in the retrieval process. It evaluates probability of relevance based on the occurrence of terms in queries and in documents.

2.2 A Variety of Document Clustering Algorithms

In this section, we review the most common clustering algorithms. Over the past few years, clustering techniques have been developed [19], [25]. The goal of clustering is to group the points in a feature space optimally based on proximity. Document or text

clustering relates to the automatic grouping of documents into clusters, so that documents within a cluster have high similarity in comparison to one another, but are very dissimilar to documents in other clusters.

Text Clustering differs from text classification. Text classification is a supervised learning process that involves pre-defined category labels; while text clustering is an unsupervised task (no pre-defined category labels are available).

Document Clustering is widely applicable in areas such as web mining, text mining and information retrieval. Recently, it has been used in browsing large collection of documents [6] and in organizing the results returned by a search engine to help users find relevant documents (within the query results) faster [62].

This section focuses on the techniques used in document clustering and offers a brief review of hierarchical and partitional clustering methods, which are used in this study. These techniques are the most common and differ in the way clusters are organized. Hierarchical algorithms produce a hierarchy of clusters, while partitional algorithms generate a flat partition of the data objects.

2.2.1 Hierarchical Clustering Algorithms

Hierarchical Text Clustering creates a hierarchical decomposition of the documents [55]. It produces a nested sequence of partitions with a single cluster at the top and individual documents at the bottom of the hierarchy. Each cluster at the intermediate level can be viewed as combining two or more clusters from the next lower level, or splitting a cluster from the next higher level. A hierarchical clustering defines a tree called a dendrogram [25]. A dendrogram is a tree structure that displays the clusters that are merged during clustering. Figure 2.1 shows how five documents can be merged into a single cluster. The parent-child relationship among the nodes in the dendrogram provides taxonomy and facilitates browsing.

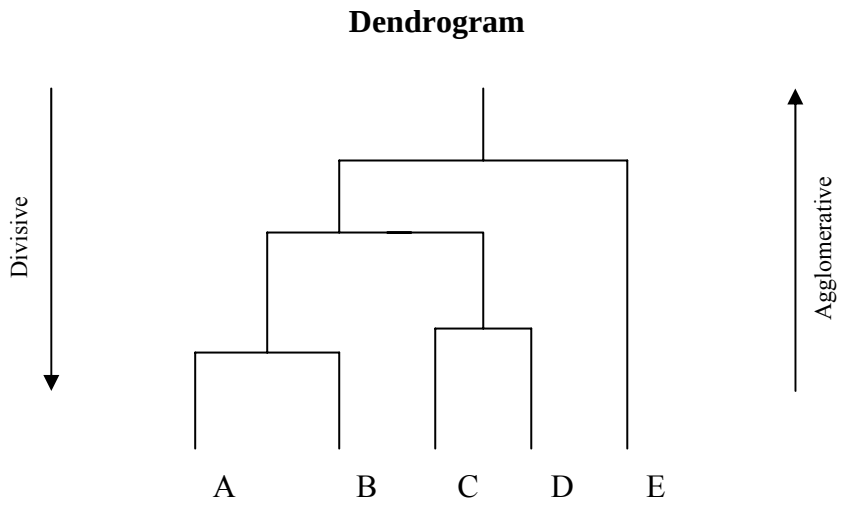

Figure 2.1: Sample Dendrogram

There are two basic approaches to generate a hierarchical clustering. Hierarchical clustering algorithms are either agglomerative (bottom-up) or divisive (top-down).

Hierarchical Agglomerative Clustering (HAC)

Hierarchical Agglomerative Clustering proceeds bottom-up. It starts with the documents as individual clusters and, at each step, computes the similarity between all pairs of clusters and merges the most similar pair. The algorithm continues until a single cluster is formed at the top of the hierarchy. A definition of cluster similarity or distance is required. In the following page we present the most commonly used techniques for calculating the similarity between two clusters.

The following summarizes the basic hierarchical agglomerative clustering algorithm [55]:

1. Treat each document as a cluster on its own.
2. Compute the similarity between all pairs of clusters, calculate the similarity matrix whose ij^{th} entry gives the similarity between the i^{th} and j^{th} clusters.
3. Merge the most similar two clusters.
4. Update the similarity matrix entries for the newly formed cluster and the other clusters.
5. Repeat steps 3 and 4 until only one cluster remains.

The time complexity of hierarchical agglomerative clustering algorithm is $O(n^2)$ where n is the number of documents. All hierarchical methods need to compute similarity for all pairs of n individual instances.

Divisive Methods

The Divisive approach follows the opposite strategy. It starts with all documents in the same root cluster. It works by iteratively splitting each cluster into a number of smaller ones until clusters with one document (singleton clusters) are produced at the leafs of the tree hierarchy or until the desired number of clusters is reached.

Methods to Compute Similarity between Clusters

In Hierarchical Agglomerative Clustering a number of different methods have been proposed for determining the next pair of clusters to be merged, i.e. how we define cluster similarity. There are four commonly used techniques: single-link [54], complete-link [31], group-average [24], [55] and centroid similarity method.

- **Single-Link Method**

In the single-link method, the similarity of a pair of clusters (C_i, C_j) is the maximum similarity between any two individuals, one in each cluster:

$$sim(C_i, C_j) = \max_{x \in C_i, y \in C_j} sim(x, y) \tag{2.4}$$

where x and y are documents in cluster C_i and C_j correspondingly.

However, this method is highly susceptible to noise, outliers, artifacts and suffers from a chaining effect [39]. It has a tendency to form loosely bound clusters [25]. Single-link algorithm remains popular due to its simplicity and the availability of an optimal space and time complexity [53]

- **Complete Link Method**

In complete-link algorithm, the similarity between two clusters (C_i, C_j) is the minimum of all pairwise similarities between documents in the two clusters:

$$sim(C_i, C_j) = \min_{x \in C_i, y \in C_j} sim(x, y) \tag{2.5}$$

where x is a document in cluster C_i and y in cluster C_j. This method produces "tighter" clusters that are typically preferred.

- **Group Average Method (UPGMA)**

Computes the average similarity across all pairs of documents within the two clusters (C_i, C_j) that will be merged (including the pairs of documents within each one of two clusters):

$$sim(C_i, C_j) = \frac{\sum_{x \in C_i, y \in C_j} sim(x, y)}{|C_i| * |C_j|} \quad (2.6)$$

where x is a document in cluster C_i and y in cluster C_j.

However, due to the complexity of computing the similarity between every pair of clusters, UPGMA does not scale up well for large data sets.

- **Centroid Similarity Technique**

The similarity between two clusters (C_i, C_j) is defined as the cosine between their centroid vectors. The centroid vector of a cluster is defined as the mean vector of data objects. The similarity between two centroids is:

$$sim(C_i, C_j) = \cos(\vec{c_i}, \vec{c_j}) = \vec{c}_i \bullet \vec{c}_j / \|\vec{c}_i\| * \|\vec{c}_j\| \quad (2.7)$$

where c_i, c_j are the centroid vectors of the two clusters. Note that the centroid vectors will not necessarily be of unit length.

Among the four methods discussed above, group average is the preferred one performing for document clustering [18], [55]. More elaborate schemes have also been developed. See for example Cure [16], Rock [17] and Chameleon [28].

2.2.2 Partitional Clustering Algorithms

Contrary to hierarchical clustering techniques, a partitional clustering algorithm creates a flat (non-hierarchical) clustering of data objects. There are many partitional clustering techniques available. The K-Means algorithm is widely used in document clustering because it is easy to implement and has low time complexity.

K-Means

K stands for the desired number of output clusters. For K-Means we use the notion of the centroid, which is the mean or the median point of a group of data objects. Given a set S of documents and their corresponding vector representations, the centroid vector C_S is the vector obtained by averaging the weights of the various terms presented in the documents of S. Note that a centroid almost never corresponds to an actual data object. The similarity between a document d and a centroid c is computed according to Equation 2.4. Note that even though the document vectors are of unit length the centroid vector is not necessarily of length one.

The basic K-Means algorithm works as follows [25]:

1. Randomly select K points as the initial cluster centroids (seeds).
2. For each point, put the point in the cluster whose centroid is the closest (most similar). The most common measure to calculate the similarity between a document and a centroid is the vector cosine measure which we use in this study.
3. Re-compute the centroid of each cluster using the current cluster members.
4. Repeat steps 2 and 3 until an objective criterion is met.

At step 4 of the algorithm, there are two most commonly used objective functions.

- The procedure terminates when there is no re-assignment of instances to new cluster.
- The second popular objective function is the mean squared distance function, which tend to work well with isolated and compact clusters. The square error criterion function for a clustering of N documents (containing K clusters), is:

$$e^2(K,N) = \sum_{j=1}^{k} \sum_{i=1}^{n_j} \left\| x_i^{(j)} - c_j \right\|^2 \tag{2.8}$$

  where $x_i^{(j)}$ is the i^{th} document belonging to the j^{th} cluster and c^j is the centroid of the j^{th} cluster.

At step 3 of basic K-Means algorithm, there are two ways to update the centroid:

- Continuously: The centroid is updated after each data object is assigned.
- Non-continuously: The centroid is adjusted only at the end of iteration, when all the data objects have been assigned.

Experimental results [55][13] indicated that the continuous centroid adjustment is more effective.

K-Means and other partitional clustering techniques are well-suited for clustering large data sets due to their low computational requirements. Their time complexity is $O(n)$ where n is the number of data objects (documents). They are more efficient as compared to the HAC algorithm which has quadratic computation time.

However, a drawback of K-Means is that K must be known in advance. An incorrect estimation of the input parameter may lead to poor accuracy. To avoid this, we try out several K and the best configuration is obtained (the one that optimizes the objective function). Apart from that, X-Means [42] is an algorithm implemented to avoid this inaccuracy.

K-Means is also sensitive to the selection of the initial centroids. Several methods have been reported in the literature, which attend to select a good initial partition. The most efficient are:

- Run *K*-Means several times with different initial centroids and pick the best result.
- Use heuristics to pick initial centroids.

Hybrid Approaches to Pick Good Initial Centroids

Buckshot and Fractionation are two methods designed to find the initial centroids in order to avoid random selection. Both techniques are based on other clustering algorithms. They cluster well, but their run time is slower than plain K-Means.

• Buckshot Algorithm

The Buckshot algorithm avoids problems of bad seed selection. It combines Hierarchical Agglomerative Clustering and *K*-Means clustering techniques [6]. Buckshot randomly picks $\sqrt{Kn}$ documents from an input set of n documents. Then, it runs group-average HAC on this sample. This algorithm has $O(\sqrt{(Kn)^2}) = O(Kn)$

time complexity. The K centroids resulting from HAC become the initial centers for *K*-Means.

- **Fractionation Algorithm**

Fractionation [6] uses the HAC, like Buckshot, to build a bottom-up hierarchy. If we want to get K clusters, fractionation algorithm splits N documents into $M > K$ groups of size N/M each (M is a parameter). Then uses HAC for each of the M groups to generate pN/M clusters (p is a parameter). The iteration terminates when only K groups remain. The algorithm takes $O(Kn)$ time.

Summarizing, Buckshot applies HAC to sample the documents randomly in order to find initial centroids. Fractionation uses successive applications of HAC over particular groups of documents to find centroids. Fractionation is more accurate than Buckshot [6]. However, Buckshot is significantly faster, so it is more appropriate for many applications.

Bisecting K-Means

Partitional algorithms can also be used to obtain hierarchical clustering solutions via a sequence of repeated applications of K-Means algorithm. The Bisecting K-Means algorithm uses this approach to build a hierarchy of clusters. It is very effective in many applications (browsing, indexing, navigation, and information retrieval systems). Bisecting K-Means algorithm starts with a single cluster of all the documents and works as follows [55]:

1. Choose a cluster to split (starting with the initial cluster).
2. Apply the basic *K*-Means algorithm to split this cluster into 2 sub-clusters (Bisecting step).
3. Repeat step 2 for ITER times and take the split that produces the clustering with the highest overall similarity (the average pairwise similarity between all documents in the cluster). We want to maximize that sum over all clusters.
4. Repeat steps 1, 2 and 3 until a pre-defined stopping criterion is met.

At step 1, there is a number of different ways to select which cluster to split from the list of leaf clusters. We can select either the cluster with the least cohesion (the least overall similarity), or the one with the largest size. Alternatively, a criterion based on

both overall similarity and cluster size can be used. Experiments in [55] indicated small differences between these possible methods.

To summarize, the Bisecting *K*-Means algorithm is a divisive hierarchical clustering technique. Its time complexity is about linear to the number of documents, $O(n*M)$, where M is the number of the produced clusters. In chapter 3, we present in detail our implementation of Bisecting K-Means algorithm.

2.2.3 Representation of clusters

The meaning of cluster representation was introduced in many studies [7], [12], [38]. In many applications the resulting clusters have to be represented or described in a compact form to achieve data abstraction. Jain [25] summarized three representation schemes and indicated that among them, the use of the centroid to represent a cluster is the most popular way. In many cases, the cluster can be effectively represented by a number of the highest weighted terms in the centroid vector [32].

2.2.4 Comparison of Document Clustering Techniques

Experimental results [55] indicated that group-average hierarchical clustering algorithm (UPGMA) is the best performing hierarchical technique. However, it has limited applicability because of its quadratic time complexity. K-Means and its variants are commonly preferred due to their time complexity which is linear to the number of documents. Moreover, partitional algorithms can also be used to obtain hierarchical clustering solutions via a sequence of repeated bisections (Bisecting K-Means). Notice that, Bisecting K-Means has a linear time complexity.

As reported in [55]:

- Bisecting *K*-Means is better than regular *K*-Means and UPGMA.
- Although, results of basic K-Means can vary from one run to another, K-Means is generally better than UPGMA (i.e., achieves better clustering quality).

Bisecting *K*-Means has linear time complexity as opposed to the quadratic time complexity of HAC. Furthermore, Bisecting *K*-Means is not susceptible to initialization issues. Finally, it is ideal for clustering large document collections not only due to its linear time complexity, but also due to its higher clustering quality.

2.3 Clustering Quality

The quality of various clustering algorithms can be evaluated with regards to both internal and external measures [65]. Internal measures compare different sets of clusters without reference to external knowledge. The cohesiveness of a cluster, which is called "overall similarity" and is based on the pairwise similarity of the documents in a cluster, is an internal measure. Contrary to internal measures, external measures evaluate the clustering quality by comparing the clusters produced from clustering algorithms against already defined classes. The most common external measures are "entropy" and "*F*-Measure". Internal and external metrics are subsequently discussed.

2.3.1 Overall Similarity

In the absence of class labels, as external information, overall similarity is an internal cluster quality measure [55]. In a clustering solution, objects within a cluster are most similar to each other than objects that come from different clusters. Particularly, the cluster cohesiveness is defined as the average pairwise similarity between objects in a cluster S:

$$\frac{1}{|S|^2}\sum_{\substack{d\in S\\ d'\in S}}\cos(d',d) \tag{2.9}$$

The above Equation is just the squared length of the cluster centroid vector, $\|c\|^2$. This equivalence is shown in section 3.3.

2.3.2 Entropy

Entropy is an external measure of cluster "goodness" [65]. It provides a measure of quality for un-nested clusters or for the clusters at a certain hierarchy of clusters. Initially, we calculate the entropy of each cluster, i.e., for cluster j we compute p_{ij} the probability that a member of cluster j belongs to class i. Then, the entropy of each cluster j is defined to be:

$$E_j = -\sum_i p_{ij} * \log(p_{ij}) \tag{2.10}$$

where the sum is taken over all classes. The entropy of the entire clustering solution is defined to be the sum of the individual cluster entropies weighted by the size of each cluster:

$$Entropy = \sum_{j=1}^{m} \frac{n_j * E_j}{n} \tag{2.11}$$

where n_j is the size of cluster j, m is the number of clusters and n is the total number of data objects. A clustering solution is perfect when the entropy is zero.

2.3.3 F-Measure

F-Measure [32] is more suitable for measuring the effectiveness of not only partitional but also of hierarchical clustering. We use this metric in this work to evaluate our clustering implementations. *F*-Measure combines the precision and recall ideas from information retrieval area.

For each manually labelled category (topic) T, we assume that a cluster C corresponding to the topic T will be formed somewhere in the hierarchy. To find the cluster C corresponding to category T, traverse the hierarchy computing precision, recall and *F*-Measure. For any category T and cluster C, we define:

$$\mathrm{P}(C,T) = N / |C| \tag{2.12}$$

$$\mathrm{R}(C,T) = N / |T| \tag{2.13}$$

$$F - Measure = 2 * P * R / (P + R) \tag{2.14}$$

where N is the number of members of category T in cluster C, $|C|$ is the number of documents in cluster C, $|T|$ is the number of documents in category T.

For hierarchical clustering, we consider the cluster with the highest F-Measure to be the cluster corresponding to the category T. The overall F-Measure for the hierarchy is computed by taking the weighted average of the *F*-Measure for each topic T and is defined as:

$$Overall\_F - Measure = \frac{\sum_{TeS} |T| * F(T)}{\sum_{TeS} |T|} \tag{2.15}$$

where S is the set of categories, $|T|$ is the number of documents in topic T and $F(T)$ is the *F*-Measure for topic T.

F-Measure score ranges from 0 to 1. A higher *F*-Measure score indicates a better clustering solution.

2.4 Stopping Criteria in Bisecting K-Means Algorithm

As mentioned Bisecting K-Means is a divisive hierarchical clustering technique. Step 4 of the basic algorithm indicates that the procedure terminates when a stopping criterion is met. Thus, a strategy to stop bisections is needed. In recently published studies there are various criteria which have been proposed as stopping rules. In this section, we make a brief review of all known methods.

- The most commonly used method suggests stopping the algorithm when no more clusters can be split. In case of document clustering this implies that the algorithm continues until each leaf cluster of the hierarchy contains a single document. The reasons of this are that: 1) no prior knowledge of the desired number of clusters is available in a specific application; 2) the purpose of clustering is to find the complete hierarchy.
- Bisecting K-Means algorithm continues partitioning until the desired number K of leaf clusters is reached [55]. This rule is the most simple and can be applied if there is prior knowledge of the desired number of clusters.
- Karypis and Zhao proposed in [64] an alternative stopping criterion for terminating the divisive procedure. Specifically, they stop splitting a cluster if it contains less that 5% of the total number of documents. The algorithm terminates when all the resulted clusters meet the stopping condition.
- The stopping criterion proposed by Ding [8] is derived from the Min-Max Cut algorithm [9]. It was developed using similarity concepts and was based on a min-max clustering principle: "Data should be grouped into clusters such that similarity between different clusters is minimized while the similarity within each cluster is maximized".

We briefly describe the Min-Max algorithm. If n is the number of data objects and $W = (w_{ij})$ is the pairwise similarity matrix, where w_{ij} is the similarity between i, j, we desire to partition the data into two clusters A_1, A_2 using the min-max clustering principle. The similarity between A_1, A_2 is defined to be

$s(A_1, A_2) = \sum_{i \in A_1} \sum_{i \in A_2} w_{ij}$. The similarity within a cluster A_1 is the sum of pairwise similarities within A_1. The clustering principle requires minimizing $s(A_1, A_2)$, while maximizing $s(A_1, A_1)$ and $s(A_2, A_2)$ simultaneously. These requirements lead to the minimization of the Min-Max Cut objective function,

$$J_{MMC} = \frac{s(A_1, A_2)}{s(A_1, A_1)} + \frac{s(A_1, A_2)}{s(A_2, A_2)} \tag{2.16}$$

Generally, for $K \geq 3$, we define $J_{MCC}(A_1, ..., A_k) \equiv J_{MMC}(K)$,

$$J_{MMC}(K) = \sum_{p,q} J_{MMC}(A_p, A_q) = \sum_k \frac{s(A_k, \overline{A_k})}{s(A_k, A_k)} \tag{2.17}$$

where $\overline{A_k} = \sum_{p \neq k} A_p$ is the complement of A_k.

After that brief explanation of Min-Max Cut algorithm, we return to its use as a stopping criterion for terminating Bisecting K-Means technique. The authors proposed that the algorithm terminates when J_{MMC} (computed on recent leaf clusters) exceeds a pre-defined threshold value J_{stop}. They showed that as the algorithm continues (the number of leaf clusters increases) J_{MMC} increases. This strategy to stop bisections can be used in applications where the correct K, as the desired number of clusters, is not known.

2.5 Bayesian Information Criterion (BIC)

In section 3.5 we will propose a strategy for terminating our implementation of Bisecting algorithm. The proposed stopping criterion is based on the Bayesian Information Criterion (BIC) or Schwarz Criterion [52]. BIC is discussed below.

Building upon the BIC criterion, Pelleg and Moore [42] proposed X-Means, a new K-Means variant algorithm. X-Means first adapted BIC for estimating the best K clusters from a given range of values automatically. The algorithm searches over the values of K and scores each clustering result using the BIC criterion. An equivalent technique called Minimum Description Length (MDL) is applied in [41]. The problem of model selection is how to choose the best one among a set of candidate models (we assume as model in this case each clustering result in X-Means).

Let D be a set of documents $\{x_1, x_2, ..., x_n\}$. D can be partitioned into disjoint subsets $D_1, D_2, ..., D_K$. In the case of Bisecting K-Means $K = 2$. Let μ_j be the centroid of the j^{th} cluster ($1 \le j \le K$). Let (i) be the index of the centroid which is closest to the $i-th$ data point. For example, $\mu_{(i)}$ is the centroid nearest to the $i-th$ data point during an iteration ($1 \le i \le n$). Let $D_j \subseteq D$ be the set of data points that have μ_j as their closest centroid. Let $R = |D|$ and $R_j = |D_j|$. The number of dimensions is M. Notice that, in our case the initial data set is partitioned into two clusters.

The BIC of the model M_j (i.e., in our case the parent cluster or the two children clusters) is given by [29]:

$$BIC(M_j) = \hat{l}_j(D) - \frac{p_j}{2}\log R \tag{2.18}$$

where $\hat{l}_j(D)$ is the log-likelihood of the data according to the model M_j, while $p_j = K(M+1)$ is the number of independent parameters in M_j.

The BIC, according to Equation 2.18 contains two components. The first term (log-likelihood of the data points) can be used as a measure of the cohesiveness of a cluster in order to denote whether a cluster should split or not. We estimate how close to the centroid are the documents of the cluster. More specific, given a cluster of points, drawn from a Gaussian distribution $N(\mu, \sigma^2)$, log-likelihood is the probability that a neighborhood of points follows this distribution. The second term penalizes the complexity of the model [4]. We assume that some data points belong to the cluster. However, due to the complexity of the model (many parameters or many data points), the data points, in addition to Gaussian, may follow other distributions. For this reason, we give a penalty by the second term of Equation 2.18.

The maximum likelihood estimate (MLE) for the variance is given by:

$$\hat{\sigma}^2 = \frac{1}{R_n - K}\sum_i \left\| x_i - \mu_{(i)} \right\|^2 \tag{2.19}$$

where R_n denotes the number of documents in cluster D_n.Given a cluster of data points, $P(x_i)$ is the probability that a point x_i follows the distribution $N(\mu, \sigma^2)$ produced by the cluster.

$$P\left(x_i\right) = \frac{R_n}{R}\frac{1}{\sqrt{2\pi}\hat{\sigma}^M}\exp\left(-\frac{1}{2\hat{\sigma}^2}\left\|x_i - \mu_{(i)}\right\|^2\right) \tag{2.20}$$

Thus, the log-likelihood of the data in cluster C_i can be calculated as the logarithm of the product of probabilities:

$$\begin{aligned}\hat{l}\left(C_i\right) &= \log\prod_i \hat{P}\left(x_i\right) \\ &= \sum_i\left(\log\frac{1}{\sqrt{2\pi}\hat{\sigma}^d} - \frac{1}{2\hat{\sigma}^2}\left\|x_i - \mu_{(i)}\right\|^2 + \log\frac{R_n}{R}\right) \\ &= -\frac{R_n}{2}\log\left(2\pi\right) - \frac{R_n M}{2}\log\left(\hat{\sigma}^2\right) - \frac{R_n - K}{2} + R_n\log R_n - R_n\log R\end{aligned} \tag{2.21}$$

To extend the formula in Equation 2.18 for all centroids instead of one, we use the fact that the log-likelihood of the data points that belong to all centroids is the sum of the log-likelihood of the individual centroids. Thus, Equation 2.18 can be re-written as:

$$BIC\left(M_j\right) = \sum_{j=1}^{K}\hat{l}\left(C_j\right) - \frac{p_j}{2}\log R \tag{2.22}$$

The number of free parameters p_j is the sum of:

- $K-1$ class probability
- $M * K$ centroid coordinates
- One variance estimate

The variance (Equation 2.19) estimates the average of the square of the distance of each document from the centroid (mean) of the cluster. This is a measure of the cohesiveness of the cluster. By computing BIC we estimate how close to the centroid are the documents of the cluster.

Given a set of clustering results, the one with the highest BIC score, $\arg\max_i BIC\left(M_j\right)$, is selected. X-Means uses the BIC in order to determine the number of clusters K in K-Means method.

Chapter 3

Clustering Algorithms Implemented

In this chapter, we present our methodology for efficient document clustering. We describe in detail our implemented clustering algorithms and suggest methods that improve their performance. We deal with K-Means which is the basic partitional clustering technique. We focus on a variant of the standard K-Means algorithm, the so-called Incremental K-Means which is combined with a Bisecting K-Means algorithm for obtaining hierarchical clustering solutions. Finally, we suggest incorporating the Bayesian Information Criterion (BIC) as a splitting criterion within the above Bisecting Incremental hierarchical clustering approach. All these suggested techniques are integrated in a new proposed algorithm**,** called here-after "BIC-Means" and is described in detail in this chapter.

3.1 Proposed Methods

In this study, our main objective is to develop a highly efficient algorithm for clustering very large document collections (such as OHSUMED), We focus on partitional clustering techniques due to their low time complexity (which is linear on the number of documents). Partitional methods have advantages in applications with large data sets for which the construction of a dendrogram using hierarchical clustering with agglomerative method is computationally prohibitive. We implemented and evaluated various partitional clustering methods. Our method can organize large collections of documents into a hierarchical binary tree. To prevent over-splitting of clusters (and termination at singleton clusters) we propose a strategy based on Bayesian Information Criterion (BIC) to stop the divisive procedure. The cluster splitting stops when meaningful clusters are reached. The combination of Bisecting Incremental K-Means with Bayesian Information Criterion is the main

contribution of this work. All methods are implemented and evaluated using standard document collections (such as Reuters and OHSUMED)

We implemented several variants of the original K-Means method and we proposed the so-called "Incremental K-Means" a variant of the original K-Means method that differs from K-Means in the way that the centroids are updated during each clustering iteration. In Incremental K-Means each cluster centroid is adjusted after each document is assigned to a cluster rather than recomputing each cluster centroid after each iteration when all documents have been assigned to clusters. The experimental results indicated that the Incremental K-Means algorithm performs better than K-Means and needs less iterations to produce a good clustering result.

Despite their linear time complexity, the main disadvantage of K-Means and Incremental K-Means is that K must be known in advance. An incorrect estimation of the number of clusters K may lead to poor clustering accuracy. Both K-Means and its variants produce a flat partition of the data collection. As mentioned in the introductory chapter, methods which are able to provide effective navigation and organization of information (like hierarchical clustering) are preferred. Thus, we are led to organize information in a hierarchical structure.

In the following we present our version of Bisecting K-Means algorithm, which combines the strengths of partitional and hierarchical clustering methods. Furthermore, it is not as sensitive to initialization issues. A hierarchy is built by recursively applying our version of Incremental K-Means algorithm. For this reason, we call our implemented algorithm "Bisecting Incremental K-Means". The so-obtained clusters are structured as a hierarchical binary tree (or a binary taxonomy). This is the reason why the bisecting approach is very suitable and effective in many applications (e.g. document retrieval, indexing, browsing, navigation systems). The algorithm proceeds until each leaf node of the cluster hierarchy contains one document.

The experimental results indicated that Bisecting K-Means outperforms basic K-Means and our variant Incremental K-Means in terms of accuracy and efficiency. This confirms the results of [55] and [63].

An important issue in divisive clustering approaches is to determine a strategy to terminate the divisive procedure of the Bisecting algorithm. In most cases there is no prior knowledge about the desired number of clusters. For this, a stopping condition must be defined.

We propose the use of Bayesian Information Criterion (BIC) or Schwarz Criterion [52] to build a strategy to stop the divisive procedure. We suggest using BIC as the splitting criterion of a cluster. We compute the BIC score to measure the improvement of a cluster when it is split. If the BIC score of the new cluster structure is less than the BIC score of the parent cluster we do not split the initial cluster. In such cases we keep the parent cluster as is and we do not select it as a candidate cluster to split in the next iteration of the algorithm. Consequently, we terminate the bisecting algorithm when there is no separable cluster according to the BIC function.

Overall, we propose the so-called BIC-Means clustering algorithm. BIC-Means produces a hierarchical clustering solution and combines all these ideas:

1. Bisecting algorithm to build a hierarchy of clusters effectively.
2. Incremental K-Means as the proposed partitional algorithm to bisect the selected leaf cluster at each bisecting step.
3. A stopping criterion for terminating the divisive procedure using the Bayesian Information Criterion (BIC).

Our proposed method is described in detail below.

3.2 Preliminaries on Document Modeling

Representing documents for clustering and other text mining tasks is fundamental in the process of knowledge discovery. We define the similarity measure which is used to compute the similarity between two documents.

3.2.1 Document Representation

The various clustering algorithms are described and implemented based upon the Vector Space Model (VSM) [50] for measuring document similarity. In this model, each document d is considered to be a vector in a multi-dimensional term space. Each dimension of the space corresponds to a unique word from the corpus.

Let D a collection of documents and $T = \{t_1, t_2, ..., t_n\}$ the set of unique terms appearing in at least one document in D. Firstly, individual words are further processed by stop-word removal. Using this preprocessing technique we remove

words without inherent meaning, such as articles or pronouns ("a", "the", "and", etc). For sample lists of stop-words, see [2].

Each document $d \in D$ is represented as a vector $d = \{w_1, w_2, ..., w_m\}$, where w_i is the weight of the term t_i within document d. In this work, a variant of $tf - idf$ weighting scheme is used. The weight w_t for each term t in document d_i is defined as:

$$w_t = \frac{(1 + \log(tf_{i,t})) \times \log \frac{N}{df_t})}{\sqrt{\sum_{s \neq t} (1 + \log tf_{i,s})^2 \times (\log(\frac{N}{df_s}))^2}} \quad (3.1)$$

where $tf_{i,t}$ is the number of times word t occurs in document d_i and df_t is the number of documents in the data set in which the word t occurs. To account for documents of different lengths, we scaled the length of each document vector so that it is of unit length.

Accordingly, a cluster, which is a set of documents, is represented in the similar way such as a document. A cluster is represented by its centroid vector (i.e., the mean vector of all its contained documents).

3.2.2 Similarity computation

Document clustering is based on the definition of document similarity. We measure the similarity between two documents d_i and d_j (or between a document and a centroid vector) using the cosine formula [50]:

$$\cos(\vec{d_i}, \vec{d_j}) = \frac{\vec{d_i} \bullet \vec{d_j}}{\|d_i\| * \|d_j\|} \quad (3.2)$$

If the document vectors are of unit length, the above formula can be simplified to $\cos(\vec{d_i}, \vec{d_j}) = \vec{d_i} \bullet \vec{d_j}$ (by normalizing by document length).

3.3 Methods Implemented

In Section 2.2.2, we determined that a cluster is represented by the centroid vector which is the mean or the median point of a cluster. Given a set S of documents and their corresponding vector representation, we define the centroid vector $\vec{C}_S$ as:

$$\overrightarrow{C_S} = \frac{1}{|S|}\sum_{d \in S} \vec{d} \tag{3.3}$$

The centroid vector is the vector obtained by averaging the weights of the various terms in the document set. The centroid vector has the following properties:

- Computing the cosine measure between a document and a centroid vector is equivalent to computing the similarity between the document and every other document within the same cluster:

$$\cos(\overrightarrow{d_1}, \vec{c}) = \overrightarrow{d_1} \bullet \vec{c} = \frac{1}{|S|}\sum_{d \in S} \overrightarrow{d_1} \bullet \vec{d} = \frac{1}{|S|}\sum_{d \in S} \cos(\overrightarrow{d_1}, \vec{d}) \tag{3.4}$$

- The square of the length of the centroid vector is the average pairwise similarity between all documents in a cluster:

$$\frac{1}{|S|^2}\sum_{\substack{d' \in S \\ d \in S}} \cos(\overrightarrow{d'}, \vec{d}) = \frac{1}{|S|}\sum_{d' \in S} d' * \frac{1}{|S|}\sum_{d \in S} d = \vec{c} \bullet \vec{c} = \| \vec{c} \|^2 \tag{3.5}$$

  Notice that Equation 3.5 includes the pairwise similarities involving the same pairs of vectors. In section 2.3 (where methods for evaluating the clustering quality were described) we used the average pairwise similarity within a cluster ("overall similarity") as a measure for cluster compactness or cluster quality.

3.3.1 K-Means Implementation

Experimental results have shown that partitional clustering methods always lead to better clustering solutions than agglomerative algorithms. Moreover, those are well suited for clustering large document data sets [55].

K-Means creates a flat, non-hierarchical clustering solution that is consisted of K clusters. It takes as input a data set and a parameter K which is the number of clusters desired. Then K-Means typically finds all K-Clusters.

We will use the symbol S to denote the set of n documents that we want to cluster. Let $S_1, S_2, ..., S_k$ be the K desired clusters and $n_1, n_2, ..., n_k$ be the sizes of the corresponding clusters.

Initially, the algorithm picks K documents (at random) as initial centroids. Then the algorithm assigns each document to each one of these random centroids. The clusters (and their centroids) are adjusted iteratively by the algorithm until

convergence (i.e., the centroids do not change significantly). Clustering results can vary based on the selection of initial centroids. For this, there are more sophisticated methods for selecting starting centroids. These methods use a heuristic or the results of another method. Buckshot and Fractionation are the most popular seed selection approaches. We explained these methods in section 2.2.2. However, experimental results published by Larsen and Aone [32] indicated that random seed selection is significantly faster than the other two methods. In the following, we chose the random seed selection in the implementation of K-Means.

Once K seeds are selected as centroids, we compute the similarity between each document and all cluster centroids. The similarity is computed according to Equation 3.2. Each document is assigned to the closest cluster centroid. Notice that even though the document vectors are of length one, the centroids vectors will not necessarily be of unit length. The similarity between a document $\vec{d}$ and a centroid $\vec{c}$ is computed as:

$$\cos(d,c) = \vec{d} \bullet \vec{c} / \| \vec{d} \| * \| \vec{c} \| = \vec{d} \bullet \vec{c} / \| \vec{c} \| \tag{3.6}$$

The next step is the "centroid re-computation". All docs assigned to the same centroid are averaged to compute a new centroid using Equation 3.3. This results to K new centroids.

We repeat the above procedure for ITER times (ITER is user defined) and take the k-way clustering result that produces the clustering with the highest overall similarity. The overall similarity of a resulting clustering is defined as the sum of the average pairwise similarities between all documents assigned to each cluster and is given by:

$$Clustering - Overall - Similarity = \sum_{r=1}^{K} \frac{1}{n_r^2} \sum_{d_i, d_j \in S_r} \cos(d_i, d_j) \tag{3.7}$$

The above formula can be re-written as:

$$Clustering - Overall - Similarity = \sum_{r=1}^{K} \|c\|^2 \tag{3.8}$$

Therefore, the clustering overall similarity is simplified as the sum of the square of the length of each centroid vector. After the K-Means algorithm has been executed ITER times we take the clustering result which has the maximum overall similarity. This is the final k-way partition. The experimental results indicated that satisfactory results are obtained when the parameter ITER is set to 5 or 6.

In most applications, K-Means algorithm continues until the centroids do not change significantly between iterations. However, due to the fact that the centroids rarely stop moving entirely and extra time is required to check for minimal movement; more advantages are obtained from determining when to stop. For this reason we chose to use the parameter ITER in our implemented version of K-Means.

Figure 3.1 summarizes the K-Means clustering algorithm.

Input: K: Number of Clusters, ITER: Number of iterations,
S: $(d_1, d_2, ..., d_n)$ document collection

Output: K clusters $S_1, S_2, ..., S_K$

Step 1. Initialize clustering. Randomly select K documents as the initial centroids of K clusters.

Step 2. Assign each document d_i to the cluster S_i with the most similar centroid. The similarity is computed according to Equation 3.4 for all clusters $S_1, S_2, ..., S_K$.

Step 3. Re-calculate the cluster centroids from assigned documents.

Step 4. Repeat steps 2 and 3 for ITER times and take the split that produces the clustering with the highest overall similarity.

Figure 3.1: Our implementation of K-Means algorithm

Figure 3.2 demonstrates an example of K-Means clustering algorithm. We show the initialization phase and how the iterations proceed.

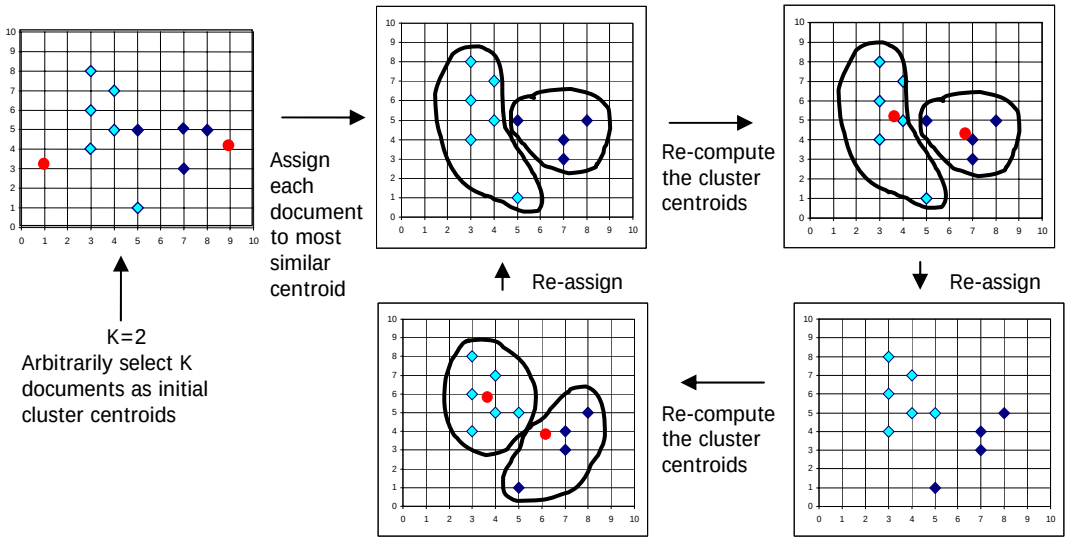

Figure 3.2: Example of K-Means Clustering algorithm

K-Means has linear time complexity on the number of documents (much more effective when compared to the quadratic computation time of hierarchical clustering techniques).

3.3.2 Incremental K-Means

In this section, we present a partitional clustering method, which we call "Incremental K-Means". Incremental K-Means is based on our implementation of K-Means clustering algorithm. The main point of this method is that the centroid is updated incrementally, as each document is assigned to a cluster. This has been shown to improve the effectiveness of basic K-Means algorithm in both execution time and clustering quality.

In K-Means during an iteration the centroid remains fixed. New centroids are computed after each iteration (after all documents have been examined). Incremental K-Means updates centroids after a document is assigned to a cluster. This way the cluster adjusts to information collected during an iteration and the centroid better reflects properties of the documents collected so far within a cluster. The following formula is used to update the centroid of a cluster with centroid $\overrightarrow{C_{old}}$ after a new document $\overrightarrow{d_{assigned}}$ is assigned to the same cluster.

$$\vec{C}_{updated} = \frac{(\vec{C}_{old} * (|S| - 1)) + \vec{d}_{assigned}}{|S|} \tag{3.9}$$

where $C_{updated}$ is the new centroid, C_{old} is the centroid before the assignment of the new document, $d_{assigned}$ is the document which is added to the cluster and $|S|$ is the new size of the cluster. The time requirement to update the centroid is constant.

After all iterations of the algorithm, new updated centroids have been computed. Then, all documents are removed from the clusters and we iterate over all documents in sequence assigning each document to the closest centroid.

Figure 3.3 summarizes Incremental K-Means:

Input: K: Number of Clusters, ITER: Number of iterations,
S: $(d_1, d_2, ..., d_n)$ document collection

Output: K clusters $S_1, S_2, ..., S_K$

Step 1. Initialize clustering. We randomly select K documents as the initial centroids of the K clusters.

Step 2. The documents are visited in a random order. When a document is assigned to a cluster, we update the corresponding centroid.

Step 3. We "clean" the clusters and iterate over the documents assigning each document to the closest centroid.

Step 4. Repeat steps 2 and 3 for ITER times and take the split with the highest overall similarity.

Figure 3.3: Incremental K-Means algorithm

Notice that step 2 examines documents in a random order. Otherwise, in a given data set the clustering will always generate the same cluster solution.

An important advantage of Incremental K-Means over K-Means is that it requires less iterations to produce an acceptable clustering result. As we shall see in the experiments, one or two iterations are sufficient. Furthermore, Incremental K-Means is not as susceptible to the seed selection technique. Experiments by Larsen and Aone [32] have indicated that Incremental K-Means creates equally good clustering results with random, buckshot or fractionation seed selection algorithm.

Similarly to K-Means, the time complexity of Incremental K-Means is $O(n)$, where n is the number of documents.

3.3.3 Bisecting Incremental K-Means

K-Means and Incremental K-Means create a flat, non-hierarchical clustering of a data set. Due to the tremendous growth in classic document collections and the internet there is an increased need for effective clustering allowing also for faster browsing through the contents of a data set. For this hierarchical clustering is more appropriate than partitional clustering. Hierarchical clustering also provides effective navigation, data summarization and organization of information by organizing large data collections into any given number of clusters which are structured as a hierarchical

binary tree. Agglomerative clustering is often thought as the best quality clustering approach for this purpose. However, it is not as effective due to its quadratic time complexity. Experimental results in [55] indicated that Bisecting K-Means always lead to better hierarchical solutions than agglomerative algorithms.

In this section, we present our implementation of the Bisecting K-Means clustering algorithm. It is derived from the standard approach. In our approach, we suggest some modifications.

We produce a hierarchical clustering solution via a sequence of repeated bisections. We chose to use our version of Incremental K-Means, as described in section 3.3.2, to bisect a cluster at each bisecting step. For this reason, we call our method "Bisecting Incremental K-Means". The choice of this algorithm instead of basic K-Means is based on our experimental results which are presented in chapter 4. These show that Incremental K-Means is better than the standard K-Means clustering technique.

The algorithm starts with a single cluster with all documents. Initially, we use our Incremental K-Means algorithm to bisect the entire collection into two clusters. Then, one of two clusters is selected and is further bisected, leading to a total of three clusters. This process of selecting and bisecting a leaf cluster continues $n-1$ times, until n leaf clusters are obtained. In this case, each leaf cluster will contain a single document. Note that n is the number of documents of the entire collection.

There are a number of different ways to choose which cluster to split from the list of leaf clusters. In our approach, we choose to split the cluster with the least overall similarity. Overall similarity is often called "intra cluster similarity" and is given by:

$$\frac{1}{|S|^2}\sum_{\substack{d'\in S\\ d\in S}}\cos(d',d) \tag{3.10}$$

where S is the set of documents in the cluster. However, as is derived from Equation 3.5, we can calculate the overall similarity of a cluster by just computing the squared length of the cluster centroid, $\|c\|^2$. This simplification decreases the time requirements to compute overall similarity before each bisecting step. Therefore, we can quickly choose the cluster to split.

Figure 3.4 summarizes Bisecting Incremental K-Means algorithm:

Input K=2 in Incremental K-Means, S: ($d_1, d_2, ..., d_n$) document collection

Output: A hierarchy of clusters (leaf clusters contain a single documents)

Step 1. Treat all the documents as one initial cluster.

Step 2. Pick a leaf cluster C (or initial) to split. Choose the cluster with the least overall similarity.

Step 3. Bisecting Step: Use Incremental K-Means, as described in section 3.3.3 to split cluster C into two sub-clusters, C_1 and C_2.

Step 4. Add the two clusters that are produced from the partition to the list of leaf clusters (candidate clusters to split).

Step 5. Repeat steps 2, 3 and 4 until each cluster at the bottom of the hierarchy contains a single document.

Figure 3.4: Bisecting Incremental K-Means algorithm

The basic Bisecting K-Means stops when the desired number of clusters is reached. Bisecting Incremental K-Means terminates when each leaf cluster contains a single document. The reason of this modification is that usually there is no prior knowledge on the desired number of clusters.

Figure 3.5 demonstrates an example of Bisecting Incremental K-Means. We use a small data set consisting of five documents and show how our method can be applied to this collection. The algorithm generates a hierarchical binary tree step-by-step. At each step the hierarchical tree is expanded by adding two new leafs. The process starts with a single cluster C, which consists of all the documents and continues until five leaf clusters are obtained, each containing one document. The final five leaf clusters are the C_4, C_5, C_6, C_7 and C_8. At each step, the leaf clusters with the least overall similarity is split in two new clusters (leaf nodes in Figure 3.5). For example, among leaf clusters C_1, C_3 and C_4 we assume that C_1 is the one with the least overall similarity, so we bisect it into clusters C_5 and C_6. Then, we continue the procedure likewise. In Figure 3.5, the clusters which are selected for bisection are highlighted orange.

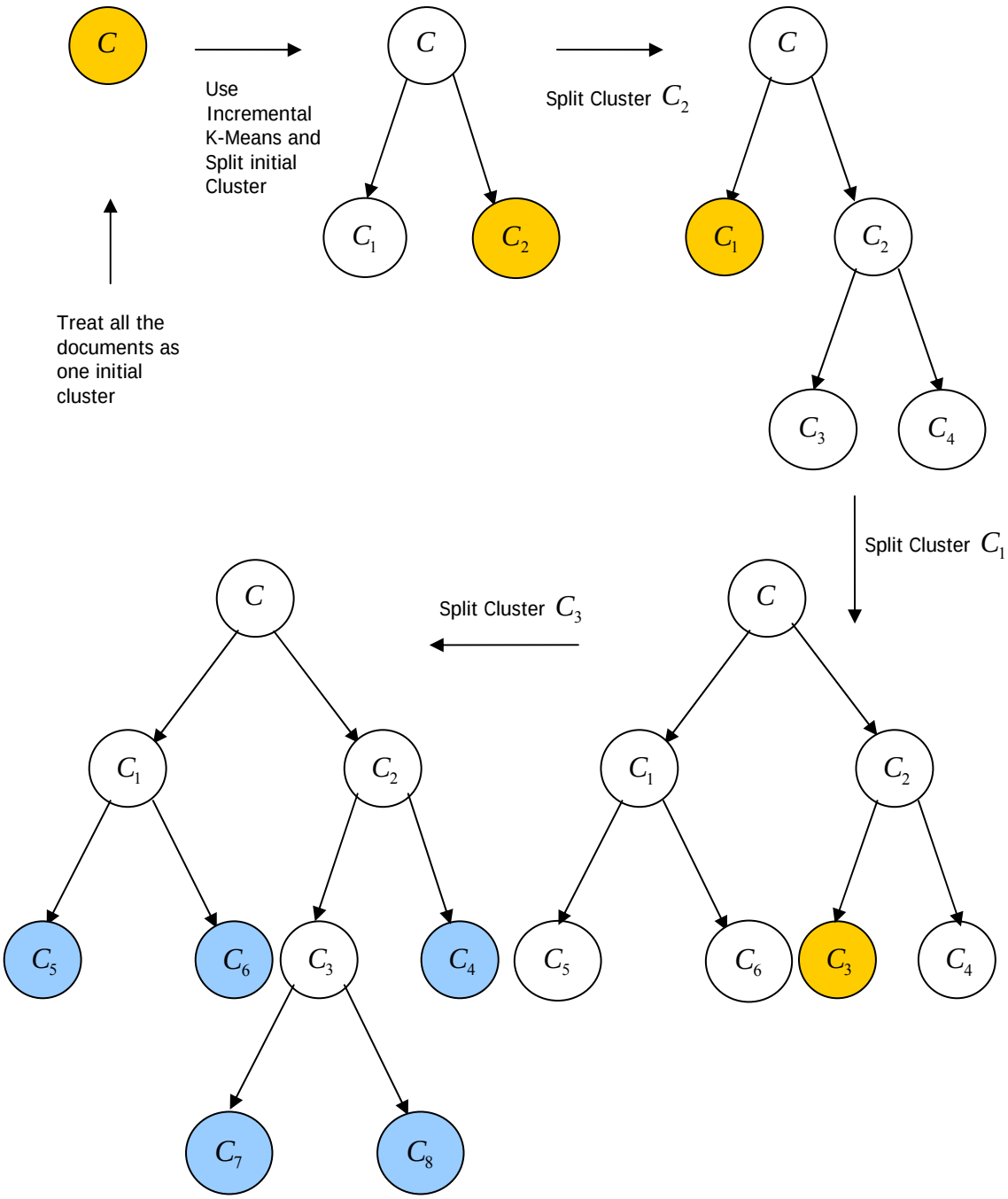

Figure 3.5: Bisecting Incremental K-Means produce a hierarchical tree

It is obvious that Bisecting Incremental K-Means is a divisive hierarchical clustering procedure. It builds a hierarchical binary tree from top (i.e. a cluster of all the documents) to bottom (each cluster contains a single document), as opposed to agglomerative approaches which build the hierarchy bottom-up.

The run time of Bisecting Incremental K-Means method is $O(n\log_2(n))$, where n is the number of documents in the entire collection. Thus, it is appropriate technique for clustering large datasets and producing hierarchy of clusters.

3.4 The Bayesian Information Criterion (BIC)

In the previous section, we described that the Bisecting Incremental K-Means algorithm continues until n leaf clusters are obtained, each containing a single document. In this case, n is the number of documents in the collection. We exhaustively execute the algorithm, because usually there is no prior knowledge on the desired number of clusters. However, terminating the procedure when each leaf cluster has one document is time-consuming. Moreover singleton clusters are meaningless. To prevent over-splitting of clusters we must define a strategy to stop the Bisecting algorithm when meaningful clusters are reached.

Toward this goal, we propose the use of Bayesian Information Criterion (BIC) or Schwarz Criterion [52] as the splitting criterion of a cluster. As discussed in section 2.5, X-Means [42] (a variant of the K-Means algorithm) first adapted the BIC to clustering algorithms for estimating the best K in a given range of values. The algorithm searches over many values of K and scores each clustering result using the so-called Bayesian Information Criterion. X-Means choose the clustering result with the best BIC score in the data (i.e., the K clusters with the highest BIC score).

In this study, we use the BIC to perform a splitting test at each cluster in order to decide whether a cluster should split or not. The BIC score is used to measure the improvement of the cluster structure between the parent cluster and its two children clusters. We compute the BIC score to initial cluster and to the resulting (child) clusters. If the BIC score of the produced children clusters is less than the BIC score of their parent cluster we do not accept the split. We keep the parent cluster as it is (we do not select it again). Otherwise, we accept the split and the algorithm proceeds similarly at lower levels.

3.4.1 Computing the BIC Score

In our case we have a collection of documents and we partition it into two clusters. The parent cluster can be considered as a model and the resulting cluster structure (with the two children clusters) as a second model. For each model, we compute the BIC score. We compare the BIC score of the two models and we accept the split if the BIC score of the second model is higher than the BIC score of the first model. Below, we describe how BIC is computed.

Let D be a set of documents $\{x_1, x_2, ..., x_n\}$. D can be partitioned into disjoint subsets $D_1, D_2, ..., D_K$. In the case of Bisecting K-Means $K = 2$. Let μ_j be the centroid of the j^{th} cluster ($1 \le j \le K$). Let (i) be the index of the centroid which is closest to the $i-th$ data point. For example, $\mu_{(i)}$ is the centroid nearest to the $i-th$ data point during an iteration ($1 \le i \le n$). Let $D_j \subseteq D$ be the set of data points that have μ_j as their closest centroid. Let $R = |D|$ and $R_j = |D_j|$. The number of dimensions is M. Notice that, in our case the initial data set is partitioned into two clusters.

The BIC of the model (i.e., in our case the parent cluster or the two children clusters) is given by [29]:

$$BIC(M_j) = \hat{l}_j(D) - \frac{p_j}{2}\log R \tag{3.11}$$

where $\hat{l}_j(D)$ is the log-likelihood of the data according to the model M_j, while $p_j = K(M+1)$ is the number of independent parameters in M_j.

The BIC, according to Equation 3.11, contains two components. The first term (log-likelihood of the documents) can be used as a measure of the cohesiveness of a cluster in order to denote whether a cluster should split or not. We estimate how close to the centroid are the documents of the cluster. More specific, given a cluster of points, that produces a Gaussian distribution $N(\mu, \sigma^2)$, log-likelihood is the probability that a neighborhood of data points follows this distribution. The second term penalizes the complexity of the model [4]. We assume that some documents belong to the cluster. However, due to the complexity of the model (many parameters

or many data points), some data points, in addition to Gaussian may follow and other distributions. For this reason, we give a penalty by the second term of Equation 3.11.

In the case of BIC score of the parent cluster K is set to 1, while in case of the two resulting clusters K is set to 2. M is the number of terms in the representations of documents. The maximum likelihood estimate for the variance is given by:

$$\hat{\sigma}^2 = \frac{1}{R-K}\sum_i \left\| x_i - \mu_{(i)} \right\|^2 \tag{3.12}$$

The variance (Equation 3.12) estimates the average of the square of the distance of each document from the centroid (mean) of the cluster. This is a measure of the cohesiveness of the cluster.

According to Equation 2.21 the maximum log-likelihood of the data in cluster D_n can be computed as:

$$\hat{l}\left(D_n\right) = -\frac{R_n}{2}\log\left(2\pi\right) - \frac{R_n M}{2}\log\left(\hat{\sigma}^2\right) - \frac{R_n - K}{2} + R_n \log R_n - R_n \log R \tag{3.13}$$

R_n denotes the number of documents in cluster D_n. The maximum log-likelihood is computed separately for the parent cluster and for each one of the two children clusters. In Equation 3.13, we always set the variable K to 1, as we pertain to the log-likelihood of a single cluster. The maximum likelihood estimate (MLE) for the variance $\hat{\sigma}^2$ is computed separately for each cluster according to Equation 3.12.

To extend the formula in Equation 3.11 for two centroids (two children clusters) instead of one, we use the fact that the log-likelihood of the data points that belong to the two centroids is the sum of the log-likelihood of the individual centroids. Thus, Equation 3.11 can be re-written as:

$$BIC\left(M_j\right) = \sum_{j=1}^{2} \hat{l}\left(C_j\right) - \frac{p_j}{2}\log R \tag{3.14}$$

The number of free parameters p_j is the sum of:

- $K-1$ class probability
- $M * K$ centroid coordinates
- One variance estimate

We can see in Equation 3.13 that, as $\hat{\sigma}^2$ increases, the likelihood decreases and therefore the BIC score (see Equation 3.11) decreases. As a result the parent

cluster must be partitioned. We can conclude that $\hat{\sigma}^2$ mostly determines the BIC score of a cluster as it provides a measure of the cohesiveness of the cluster.

As far as Equation 3.12 is concerned, computationally significant simplifications can be applied. This allows for a fast and memory efficient implementation of BIC. Particularly, we rewrite the sum $\sum_i \left\| x_i - \mu_{(i)} \right\|^2$ in Equation 3.12. By some simple algebraic manipulations this sum can be re-written as:

$$\sum_{x_i \in D_n} \left\| x_i - \mu_{(i)} \right\|^2 = \frac{1}{R_n} \sum_{x_i, x_j \in D_n} \left\| x_i - x_j \right\| = \frac{1}{R_n} 2 \sum_{x_i, x_j \in D_n} \left(1 - \cos\left(x_i, x_j \right) \right)$$

$$= \frac{2}{R_n} \sum_{x_i, x_j \in D_n} 1 - \frac{2}{R_n} \sum_{x_i, x_j \in D_n} \cos\left(x_i, x_j \right)$$

$$= \frac{2R_n R_n}{R_n} - \frac{2}{R_n} \sum_{x_i, x_j \in D_n} \cos\left(x_i, x_j \right)$$

$$= 2R_n - \frac{2}{R_n} \sum_{x_i, x_j \in D_n} \cos\left(x_i, x_j \right) \tag{3.15}$$

By using Equation 3.5 the above formula can be re-written as:

$$\sum_{x_i \in D_n} \left\| x_i - \mu_{(i)} \right\|^2 = 2R_n - 2R_n \frac{1}{R_n^2} \sum_{x_i, x_j \in D_n} \cos\left(x_i, x_j \right)$$

$$= 2R_n - 2R_n \left\| c \right\|^2$$

$$= 2R_n \left(1 - \left\| c \right\|^2 \right) \tag{3.16}$$

where $\left\| c \right\|^2$ is the square of the length of the centroid vector. Thus, the Equation 3.12 after the modifications can be re-written as:

$$\hat{\sigma}^2 = \frac{1}{R_n - K} 2 * R_n * (1 - \left\| c \right\|^2) \tag{3.17}$$

Summarizing, the value of $\hat{\sigma}^2$ for a cluster, as defined in Equation 3.17, is used in Equation 3.13 for computing the maximum log-likelihood of a cluster. In case of the parent cluster, the value of log-likelihood is applied in Equation 3.11 to compute the BIC score of the initial cluster. In case of the two resulting clusters, we compute the log-likelihood value separately for each cluster. Then, the two computed values are added, as we can see in Equation 3.14 and the BIC score of the resulting model (two children clusters) is computed.

Figure 3.6 demonstrates an example of the use of BIC as splitting criterion of a cluster. We assume cluster C is a leaf cluster in the hierarchy and according to its overall similarity (see Equations 3.5 and 3.10) we select to bisect it. Cluster C is bisected into clusters C_1 and C_2.

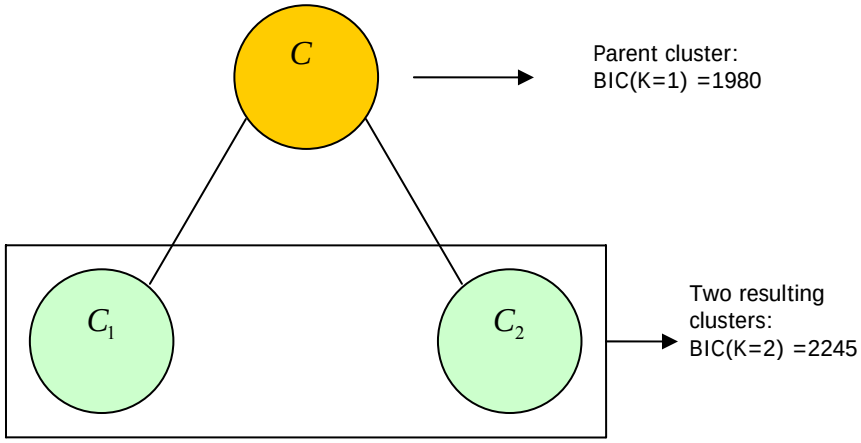

Figure 3.6: BIC as splitting Criterion of a cluster

We use the BIC to determine if the bisection is acceptable. It can be seen in Figure 3.6 that we have computed two distinct BIC scores. One for the parent cluster and another for the two children clusters. We compare these scores to decide if we split the initial cluster. It is shown in Figure 3.6 that the BIC score of the parent cluster is less than BIC score of the generated cluster structure. Thus, we accept the bisection.

3.5 BIC-Means

In this section, all techniques presented in the previous sections are integrated in a new proposed algorithm, which we call *"BIC-Means"*. It is a partitional clustering method which structures the resulting clusters as a hierarchical binary tree by recursively applying the Incremental K-Means algorithm presented in section 3.3.2. Moreover, a significant modification in our proposed final algorithm as compared with the basic Bisecting approach is the use of a stopping criterion in order to stop bisecting the clusters. Instead of continuing the algorithm until each leaf cluster contains one document, BIC-Means uses a strategy for terminating the divisive procedure. The BIC plays the most important role towards this goal.

In the following, we propose a strategy for terminating the divisive procedure in BIC-Means, when meaningful cluster are reached. Let Incremental K-Means method, as is described in section 3.3.2, be repeatedly applied in a data set which contains n documents. When this process has been executed $m-1$ times, a hierarchy of m leaf clusters is obtained, where $m < n$.

As mentioned in section 3.4, the BIC score is applied locally as the splitting criterion of a leaf cluster. It measures the improvement of a cluster when it is split. If the BIC score of the two new clusters is less than the BIC score of their parent node we do not accept the split. In such cases, the proposed strategy defines that we keep the parent cluster as is and we do not select it as a candidate cluster to split in the next iteration of the algorithm. Consequently, the BIC-Means terminates when there is no separable cluster according to the BIC function, instead of terminating at meaningless singleton clusters.

Overall, BIC-Means produces a hierarchical clustering solution and combines all the following ideas:

1. Bisecting clustering approach to build a hierarchy of clusters effectively.
2. Incremental K-Means as the proposed partitional method to bisect the selected leaf cluster at each bisecting step.
3. A termination criterion for preventing clustering from over-splitting using the Bayesian Information Criterion (BIC).

Step-by-step, the proposed BIC-Means algorithm is presented in Figure 3.7:

Input: K=2 in Incremental K-Means method, S: ($d_1, d_2, ..., d_n$) document collection, BIC formula.

Output: A hierarchy of clusters (consist of meaningful leaf clusters)

Step 1. Treat all the documents as one initial cluster.

Step 2. Pick a leaf cluster C (or initial) to split from the list of leaf clusters. Choose the cluster with the least overall similarity which is the average pairwise similarity between all documents in the cluster.

Step 3. Bisecting Step: Use Incremental K-Means, as described in section 3.3.2 to split cluster C into two sub-clusters, C_1 and C_2.

Step 4. Calculate two BIC scores for the two distinct models. One for the initial cluster C and another for the two resulting clusters C_1 and C_2. We define two possible cases:

- If the BIC score of the parent cluster is less than the BIC score of the new cluster structure: We accept the split and add the two generated clusters to the list of leaf clusters (candidate clusters to split).
- Otherwise: we keep the cluster C as it is and do not select it as a candidate cluster to split in a next iteration of the BIC-Means method. In other words, we remove cluster C from the list of leaf clusters.

Step 5. Repeat steps 2, 3 and 4, until there is no leaf cluster in the hierarchy which is separable according to the BIC score. Then, the BIC-Means algorithm terminates.

Figure 3.7: The proposed BIC-Means algorithm

Figure 3.8 demonstrates an example of BIC-Means. We assume that via a sequence of repeated bisections on a document collection, a hierarchy of clusters has been obtained. Figure 3.8 illustrates the last level of the hierarchy, where are the leaf clusters. We apply the pre-defined strategy on the four leaf clusters which are appeared in Figure 3.8 to indicate when the BIC-Means algorithm terminates according to our proposed methodology. C_1, C_2, C_3, C_4 denote the four leaf clusters in the initial hierarchy and highlight them orange.

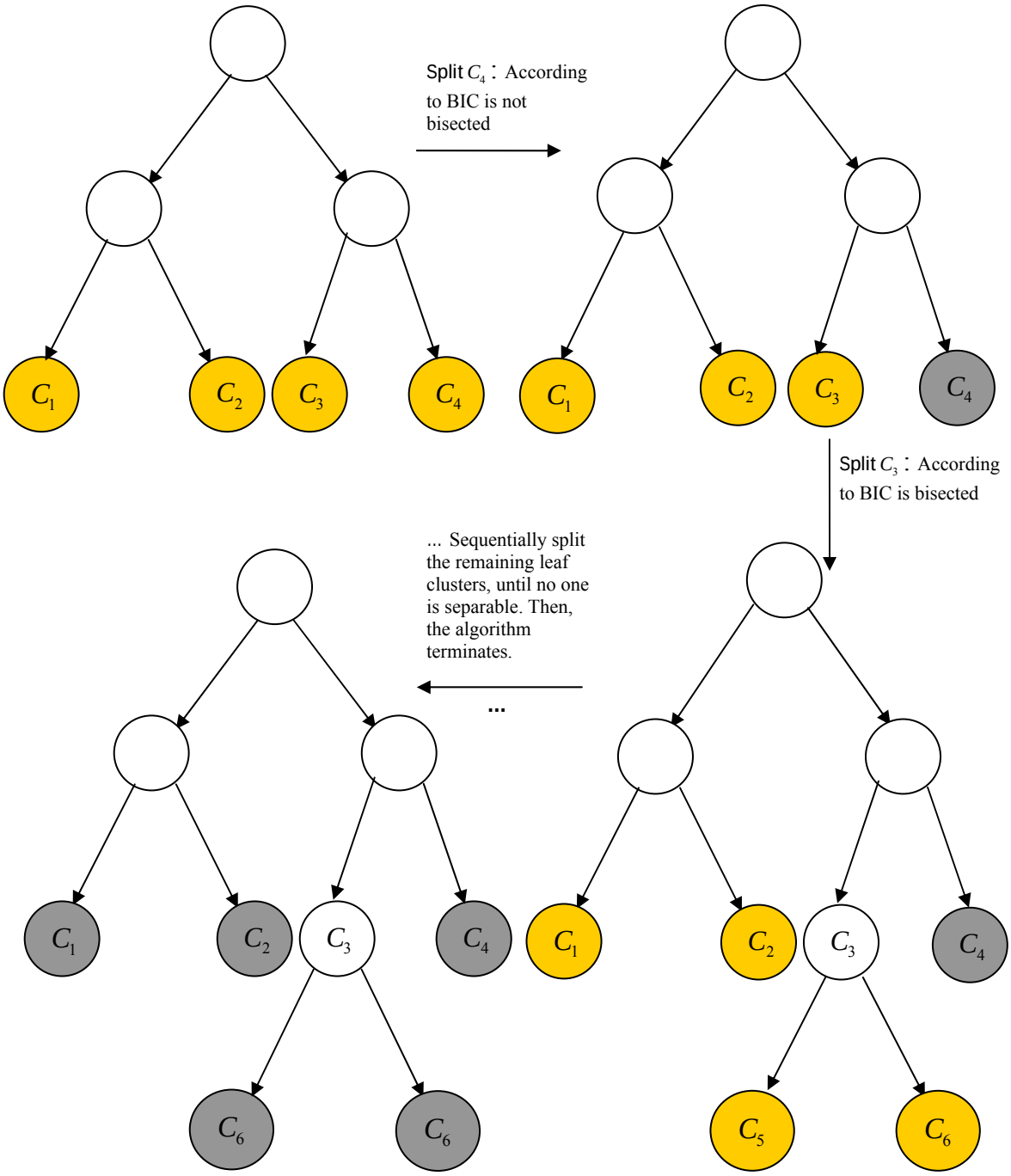

Figure 3.8: Algorithm for terminating the BIC-Means method

According to our Bisecting approach, we pick a leaf cluster to split from the list of leaf clusters. Let C_4 be the cluster with the least overall similarity. We bisect it and assume that its BIC score is greater than the BIC score of the two resulting clusters. Thus, as we described in section 3.4, we do not split the cluster C_4 and additionally do not select it for further bisections. Also, C_4 is removed from the list of leaf clusters and is highlighted gray.

We continue by selecting the next cluster for splitting. Let C_3 has the least overall similarity. We partition it into two sub-clusters C_5 and C_6 and consider that the BIC score is determining that C_3 can be split. Consequently, the list of leaf clusters consists of C_1, C_2, C_5 and C_6. For short, as the remaining leaf clusters are concerned, we bisect them sequentially. For each one, we assume that it can not be partitioned if we compare its BIC score to the BIC score of the corresponding children clusters. Therefore, there is no separable leaf cluster in the hierarchy and as step 4 of our proposed method indicates, the BIC-Means algorithm terminates.

Chapter 4

Experimental Results

We carried out two different sets of experiments. The first set of experiments focuses on the evaluation of the clustering quality of all algorithms presented in chapter 3. F-Measure was used to measure the overall "goodness" of the generated clusters. Clustering techniques have been tested on OHSUMED [21] and Reuters-21578[1] [35], two standard text corpora widely available on the Web. Both corpora offer a pre-defined categorization of its content into clusters which can be used to measuring the clustering quality of the implemented clustering algorithms. The results demonstrate that our proposed BIC-Means algorithm performs at least as good as other state of the art clustering techniques.

The second set of experiments focuses on measuring the effectiveness and efficiency of a cluster-based information retrieval system. Having established the quality of document clustering algorithms, we applied the suggested BIC-Means on OHSUMED (a very large document collection with 233445 medical articles from Medline) in order to create a hierarchy of clusters. For the evaluations, we applied a subset of 61 queries of the original OHSUMED query set developed by Hersh et al. [21]. The correct answers to these queries were compiled by human experts. We matched each query against the leaf clusters of the hierarchy and the clusters were ranked based on their similarity to the query. We evaluated several cluster-based retrieval strategies and compare them against retrieval results by exhaustive search on OHSUMED.

4.1 MeSH

MeSH[2] (Medical Subject Headings) is a taxonomic hierarchy (ontology) of medical and biomedical terms (or concepts) suggested by the U.S National Library of

[1] http://www.davidlewis.com/resources/testcollections/reuters21578
[2] http://www.nlm.nih.gov/mesh

Medicine (NLM)[3]. It is used for subject indexing and searching of journal articles in MEDLINE[4] database and other databases that are produced by the NLM. MeSH is widely used in indexing and cataloging by libraries and other institutions around the world. NLM has adopted the Extensible Markup Language (XML)[5] as the description language for MeSH. The MeSH vocabulary file is available in XML (Bray) format. There exist 23880 main headings, termed descriptors in 2006 MeSH edition. Moreover, MeSH descriptors are organized in a logical "tree" structure. There are 16 subtrees (taxonomies) or branches in the MeSH ontology (see Figure 4.1), of ISA kind of relationship between nodes (concepts) in each subtree. Within each sub-category, descriptors are arrayed hierarchically from most general (e.g. "chemicals and drugs") to most specific (e.g. "aspirin") in up to eleven hierarchical levels. Each MeSH descriptor appears in at least one place in the subtree and may appear in several places in the hierarchy. MeSH concepts correspond to MeSH objects which are described with terms of several properties [see section A.1 in Appendix A]. The most important of them being:

MeSH Headings (MH): MeSH Headings or descriptors are a collection of terms for primary themes or topics contained in literature. They are used in MEDLINE as the indexing terms for documents. Every journal article is indexed with 10-12 headings. Its use indicates the topic discussed by the work cited.

Qualifiers or Subheadings: In addition to the descriptor's hierarchy, MeSH contains a small number of standard qualifiers, which can be added to descriptors to narrow down the topic. There are 83 qualifiers in 2006 MeSH ontology. Qualifiers afford a convenient means of grouping together those citations which are concerned with a particular aspect of a subject [59].

Entry Terms: These terms are used as pointers to the MeSH Headings. Entry vocabulary has been thought of as synonyms or very similar terms of the main Heading. Entry terms, sometimes called "See cross references", indicate that information related to one term will be found under a different term. Moreover, the set of entry terms that points to a MeSH Heading are the terms that indicate the concept introduced by the MeSH Heading [22].

[3] http://www.nlm.nih.gov
[4] http://www.nlm.nih.gov/pubs/factsheets/medline.html and http://medline.cos.com/
[5] http://www.w3.org/XML

MeSH Tree Number: In the MeSH taxonomy, each MeSH Heading is characterized by its MeSH tree number (or code name) indicating the exact position of the term in the MeSH tree taxonomy. For example C is the code name of the "Diseases" subtree and the term "Slow Virus Diseases" has a tree number C02.839 meaning that this MeSH Heading belongs to C subtree (see Figure 2.1).

MeSH Scope Note: This short piece of free text provides a type of definition in which the meaning of the MeSH Heading is circumscribed.

Names of descriptors reflect the broad meaning of the concepts involved. The hierarchical relationships must be intellectually accessible to users of MeSH (e.g., clinician, librarian, and indexer). An indexer must be able to assign a given MeSH Heading to an article and a clinician must be able to find a specific MeSH Heading in the tree hierarchy.

1. **+ Anatomy [A]**
2. **+ Organisms [B]**
3. **+ Diseases [C]**
    - Virus Diseases [C02]
        - Slow Virus Diseases [C02.839] +
4. **+ Chemicals and Drugs [D]**
5. **+ Analytical, Diagnostic and Therapeutic Techniques and Equipment [E]**
6. **+ Psychiatry and Psychology [F]**
7. **+ Biological Sciences [G]**
8. **+ Physical Sciences [H]**
9. **+ Anthropology, Education, Sociology and Social Phenomena [I]**
10. **+ Technology and Food and Beverages [J]**
11. **+ Humanities [K]**
12. **+ Information Science [L]**
13. **+ Persons [M]**
14. **+ Health Care [N]**
15. **+ Publication Characteristics [V]**
16. **+ Geographic Locations [Z]**

Figure 4.1: MeSH Tree Structures 2006

4.2 Document Collections

For evaluating the quality of clustering algorithms document clustering results are compared against manually and pre-defined categorization of the corpus. To reduce

the risk that our results might be valid only on a particular corpus, we experimentally evaluated the performance of the implemented clustering algorithms on two different data sets: the Reuters-21578[6] text categorization test collection [35] and the OHSUMED collection [21]. A pre-defined categorization exists for both corpora. We created two subsets of each data set to perform the document clustering experiments. The entire OHSUMED collection was also used in cluster-based retrieval experiments. Issues related to Reuters-21578 and OHSUMED along with the description of their subsets are discussed below.

4.2.1 Reuters-21578

Reuters-21578 is a commonly used document collection for text categorization tasks. It consists of 21578 newswire articles from the Reuters news service obtained in 1987 [35]. Its domain is broad enough to be realistic and the content of the news is understandable for non-experts. Reuters-21578 is freely available and is distributed in 22 files. The files are in SGML format. Each of the first 21 files contains 1000 documents, while the last contain 578 documents. All Reuters-21578 documents have more information than the simple article reference. The structure of a Reuter's document can be found in Appendix A at section A.2.1. The most commonly used attributes in a Reuter's article are the title, the abstract and the topic.

Reuters-21578 collection comprises an "a priori" categorization of documents. They were annotated and indexed with categories by personnel from Reuters Ltd. and Carnegie Group, Inc. in 1987. The topic field is used to classify each document in a pre-define category. Documents have been categorized into 135 distinct topics (categories). Each article may be labeled with none, one or with many pre-defined topics. The lack of a label indicates that the human annotator could find an adequate topic. In our experiment we used the most commonly used split of Reuter's documents, the so-called "Mod-Apte" where the 21578 documents are separated into 9603 training documents, 3299 test documents and 8676 unused documents.

To experimentally evaluate the implemented clustering algorithms we formed two subsets of Reuters-21578. For both subsets we have selected articles that belong

[6] http://www.davidlewis.com/resources/testcollections/reuters21578

to exactly one of 135 topics (categories). Additionally, documents with an empty body field were also discarded.

The first subset, which we call *reuters1*, contains documents in which the value of LEWISSPLIT attribute is “TEST” and attribute TOPIC = ”YES” according to “Mod-Apte” split. Each article classified with a single topic. R*euters1* contains 2583 documents into 59 categories. In this categorization, categories with extremely few documents (less than 10) have been discarded. Thus, “outlier categories” are ignored in the evaluation. The resulting subset consists of 2442 documents which have been classified in 24 classes (categories). The distribution of documents per topic is shown in Table 4.1. At each cell we note the name of the topic and the number of documents that are contained in the corresponding category.

Reuters1 – 2442 documents (Category: No. of Documents)			
earn: 1081	ship: 36	cpi: 17	reserves: 12
acq: 696	money-supply: 28	cocoa: 15	jobs: 12
crude: 121	sugar: 25	gnp: 15	ipi: 11
money-fx: 87	coffee: 22	copper: 13	veg-oil: 11
interest: 81	gold: 20	iron-steel: 12	grain: 10
trade: 76	alum: 19	nat-gas: 12	tin: 10

Table 4.1: *reuters1* - Category Distribution

We call *reuters2* the second subset of Reuters-21578. It is larger than *reuters1* containing 9120 documents into 66 distinct classes (categories). The only difference between the two subsets is the value of LEWISSPLIT attribute. In *reuters2* this value can be set either “TEST” or “TRAIN”. The other settings are the same as in *reuters1*. The categories which contain less than 31 documents were discarded from this subset as well. Thus, *reuters2* contains 8712 documents into 24 classes. Category distribution is shown in Table 4.2. In both subsets the majority of the documents have been labeled with “earn” topic.

Reuters2 – 8712 documents (Category: No. of Documents)			
earn: 3920	money-supply: 151	cpi: 71	ipi: 44
acq: 2290	ship: 144	Cocoa: 61	cooper: 44
crude: 374	sugar: 122	grain: 51	iron-steel: 38
trade: 327	coffee: 111	alum: 50	nat-gas: 35
money-fx: 291	gold: 90	reserves: 49	veg-oil: 30
interest: 270	gnp: 73	jobs: 49	tin: 27

Table 4.2: *reuters2* - Category Distribution

4.2.2 OHSUMED

OHSUMED document collection was compiled by William Hersh et al. [21] at the Oregon Health Sciences University. It is a clinically oriented subset of Medline. Medline is the bibliographic database of the U.S. National Library of Medicine (NLM). It contains more that 15 million references (version 2006) to journal articles in life sciences, medicine and bio-medicine. OHSUMED consists of 348566 Medline documents from 270 medical journals taken between the years 1987-1991. 233445 of the references contain abstracts and can be downloaded from ftp://medir.ohsu.edu/pub/OHSUMED. OHSUMED has become an evaluation benchmark in text categorization and IR research since 1994 [60], [61].

OHSUMED stores a rich set of metadata associated with each article. The structure of an OHSUMED document can be found in Appendix A at section A.2.2. Publications in OHSUMED are manually indexed by NLM using MeSH Headings (MH), with typically 10-12 descriptors assigned to each reference. Title (TI), abstract (AB) and MeSH Headings (MH) are the most commonly used fields of OHSUMED references. We used these fields in our document clustering evaluation.

To evaluate implemented clustering algorithms pre-classified sets of documents are needed. For this reason, two OHSUMED subsets were formed.

We assume that OHSUMED documents belong to categories related to the MeSH Headings that are manually assigned to them. The produced subsets which we call *ohsumed1* and *ohsumed2* contain documents from the risk factors, tomography, prognosis, pregnancy, receptors, molecular sequence data, in-vitro, DNA, carcinoma,

and antibodies categories. This OHSUMED categorization has also been used for document clustering evaluations in [63]. The two subsets differ in the number of documents they contained. Ohsumed1 consists of 32230 documents classified in 10 categories (classes). Ohsumed2 contains 10902 documents into 10 classes and was produced from Medline documents of the year 1990. Category distribution of both subsets is shown in Table 4.3:

Ohsumed1 – 32230 Documents		**Ohsumed1 – 10902 Documents**	
Category	**No. docs in this cat.**	**Category**	**No. docs in this cat.**
In-Vitro	5172	In-Vitro	1194
Carcinoma	323	Carcinoma	723
Antibodies	375	Antibodies	1327
Molecular Sequence Data	6049	Molecular Sequence Data	1051
DNA	245	DNA	797
Receptor	419	Receptor	306
Prognosis	6145	Prognosis	1045
Tomography	345	Tomography	1397
Risk Factors	5896	Risk Factors	1251
Pregnancy	7261	Pregnancy	1811

Table 4.3: Ohsumed1 & Ohsumed2 – Category Distribution

The entire OHSUMED collection was used in our information retrieval experiments. The basic reason for this choice is that OHSUMED is a domain specific collection. A set of 106 queries have also been defined on OHSUMED along with the set of documents which are relevant to each query. Apart from the original OHSUMED query set developed by Hersh et al, a subset of 63 queries were used in TREC-9[7] (Trec Retrieval Conference) IR experiments.

[7] http://trec.nist.gov/data/t9_filtering.html

4.3 MeSH-Based Document Representation in OHSUMED

An important consideration for document clustering is the representation of documents. Traditionally, documents are represented by extracting individual words from text (abstract, title). In OHSUMED, each document is represented by abstract, title and MeSH terms (MeSH Headings) fields. MeSH is a control vocabulary offering a hierarchical categorization of medical concepts. In OHSUMED (the same every document in Medline) each document has been indexed manually by a set of MeSH terms.

One of the goals in our experiments was to explore the MeSH terms as features for document representation. A summary of MeSH is given in section 2.6. In a part of our experiments, instead of obtaining the term collection of a document from single word terms in title and abstract, MeSH terms were extracted and used to represent the document. They were extracted from title and abstract fields. In this MeSH term collection we added the MeSH terms accompanying each document. The use of MeSH terms is important for two reasons. First, they are assigned to OHSUMED references by trained indexers, thus many issues involved with natural language processing may be avoided. Second, they are multi-word representations corresponding to medical concepts and as such they are directly comprehensive by humans.

A MeSH term is often consisted of two or more words. For example, “abdominal pain” is a MeSH term. It is consisted of the words “abdominal” and “pain”. An issue that needs special attention here is how MeSH terms can be extracted from OHSUMED documents.

For this, we check if a word combined with its next one that come across in the document consists a MeSH term. If they do, then we check both of them with the next one if they consist a MeSH term, and so on. If they do not, then a) if a MeSH term was found until then, we keep the term and continue checking words after this term, b) if a MeSH term was not found until then, we keep the word as is and continue checking with the others. For example,

“Abdominal pain in children”

- Stopwords to remove: in

check: abdominal? NO

check: abdominal pain? YES

check: abdominal pain children? NO (END of text)

- Found MeSH term? YES (keep term)
- Continue checking after MeSH term

check: children? NO (END of text)

- Found MeSH term? NO (keep word)

Checked text: “abdominal pain children”

4.4 Evaluation Method

One of the most important issues in document clustering experiments is to find an algorithm-independent measure to evaluate the quality of the clustering result. As presented in section 2.3, several measures have been proposed in the literature. They include “entropy”, “purity”, “overall similarity”, “F-Measure” and more.

In this study, F-Measure was used to evaluate the quality of the generated clusters. We examine how closely the clusters produced by each clustering algorithm match the set of categories previously assigned to the documents. This requires the preparation of the data sets so that at each document is assigned a single topic label. The category distribution for the two subsets of Reuters-21578 was shown in Table 4.1 and Table 4.2. The categories assigned to the documents of two OHSUMED subsets were presented in Table 4.3. Each table shows the topic labels and the number of documents that belong to the specific category.

In section 2.3.3, we presented in detail how F-Measure is computed given a set of generated clusters and a pre-defined categorization of the documents. The overall F-Measure for a clustering solution is computed according to Equation 2.15. A perfect clustering solution will be one that leads to clusters which contain documents solely from a single category (class). In such case the F-Measure score will be one. In general, the higher the F-Measure values, the better the clustering result is.

We continue giving a simple example of evaluating F-Measure on a cluster hierarchy. Figure 4.2 illustrates a hierarchical clustering solution. We assume that A, B, C, D and E are five documents which constitute a small data set. Suppose, we apply on this collection the Bisecting Incremental K-Means algorithm as described in section 3.3.3. Each leaf cluster at the bottom of the hierarchy contains one document.

We assume that documents B, C and D are in fact members of a real class T. Thus, the number of documents in the category T is $|T| = 3$.

We want to find which cluster in the tree hierarchy corresponds to T. To find this cluster we traverse the hierarchy of the clusters, calculating precision, recall and F-Measure for each cluster with respect to topic T. Initially, we meet the root cluster at the top, which contains all documents ($|C| = 5$). There are three common documents in root cluster and category T. Thus, we calculate precision, recall and F-Measure according to Equations 2.12, 2.13 and 2.14.

$$\text{Pr}ecision(Root\_Cluster, T) = 3/5$$

$$\text{Re}call(Root\_Cluster, T) = 3/3$$

$$F - Measure = 0.75$$

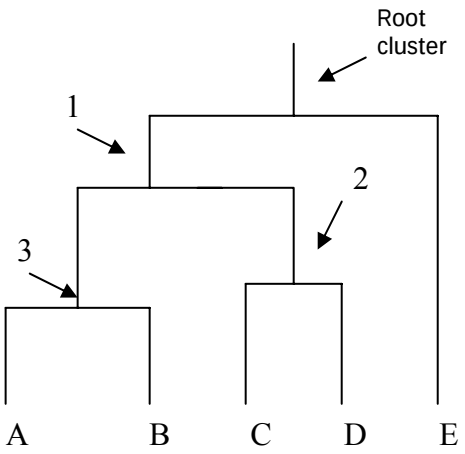

Figure 4.2: A representative evaluation example

We apply the same computation to each cluster in the tree hierarchy. The highest F-Measure is 0.85 and is obtained in cluster 1. Cluster 1 contains the documents A, B, C and D. Therefore, we consider the cluster 1 to be the cluster C corresponding to category T and 0.85 is the final F-Measure for category T. The overall F-Measure, as given by Equation 2.15 is used to indicate the quality of the whole hierarchy. It is the weighted average of the F-Measures for each category T.

4.5 Document Clustering Experiments

In this section we present our first set of experiments evaluating the quality of the clustering solutions produced by the clustering algorithms implemented in this thesis. Specifically, we evaluate and compare results obtained by the K-Means, Incremental K-Means and Bisecting Incremental K-Means methods. These are results obtained from documents represented by simple (single words) terms. For OHSUMED documents we also experimented with documents represented by MeSH terms (multi-word terms).

4.5.1 Experimental Setup

The main features of the four document collections were used in our experiments are summarized in Table 4.4.

Data	Source	No of Doc.	No of Classes
reuters1	Reuters-21578	2442	24
reuters2	Reuters-21578	8712	24
ohsumed1	OHSUMED-233445	32230	10
ohsumed2	OHSUMED-233445	10902	10

Table 4.4: Summary of the data sets

All clustering methods were implemented on top of Lucene[8] (see section A.3 in Appendix A) which is a Java-based open source toolkit for text indexing. In both data sets the documents sets were indexed by the Lucene utility. Reuters-21578 documents were indexed by title, body and topic fields. Additionally, we created a field with all distinct terms in title, body and topic. Reuters-21578 documents were indexed by this field as well. OHSUMED documents were indexed by title, abstract and MeSH terms (MeSH Headings) fields. Similarly to Reuters, one more field was indexed consisted of the distinct terms in title, abstract and MeSH field. In case of experiments in section 4.4.3, OHSUMED documents were represented only by MeSH terms extracted from title, abstract and MeSH terms fields.

[8] http://lucene.apache.org

The fields of each document were syntactically analyzed and reduced into separate term vectors (or MeSH vectors in case of MeSH-based representation).Each term in this vector was represented by its weight. The $tf - idf$ weighting scheme was used for computing the weight of each term. Each term vector was normalized by document length so that it is of unit length. On two data sets, we used a stop-list to remove common words (stop-words) (i.e. insignificant words like 'a', 'the', 'and', 'or') [2]. F-Measure was used as a measure of cluster "goodness". Additionally, we discuss the results in terms of clustering time required by each algorithm.

As mentioned in chapter 2, the main disadvantage of partitional clustering methods is that their performance is sensitive to the selection of the initial cluster centroids (i.e., clustering the same set of documents more than once with the same parameter values will generate a different clustering result). This is the reason why multiple trials are needed. Consequently, we carried out ten separate runs for each document clustering evaluation. The experimental results on partitional algorithms reported in this section correspond to the average F-Measure over ten runs.

All algorithms are implemented in Java programming language, and experiments were run on a PC with a Pentium 4 3.2GHz processor, with 2GB RAM, running Linux.

4.5.2 Evaluation and Comparison of our K-Means, Incremental K-Means and Bisecting Incremental K-Means algorithms

Experimental results on K-Means, its variant Incremental K-Means and the Bisecting Incremental K-Means method are presented below.

Evaluation and comparison of K-Means and Incremental K-means

First in our experiments we evaluated the effectiveness and efficiency of K-Means and Incremental K-Means clustering algorithms in terms of F-Measure and clustering time. In parallel, we examined for each method how the vector representation of a document can affect the clustering quality. We used reuters1 (see Table 4.1) test corpus and assumed three different ways that a document can be represented:

- By the terms from BODY field of Reuters-21578 texts,

- By the terms from TITLE field in a first vector, the terms from BODY field in a second vector and the TOPIC in a third vector and
- By all distinct terms from body, title and topic in a single vector.

For example, in the second case, when we want to compute the similarity between two documents we compute it separately for each field and then we sum the three computed values.

We set K=24, as 24 are the categories of reuters1 subset. As far as K-Means is concerned, we set the parameter ITER (number of iterations) of the algorithm equal to 6. For Incremental K-Means ITER was set to 4. As described in section 3.3.2, this value determines the number of iterations in K-Means and Incremental K-Means techniques. The resulting F-Measure values for the various document vector representations are shown in Figure 4.3. We call this experiment 1a.

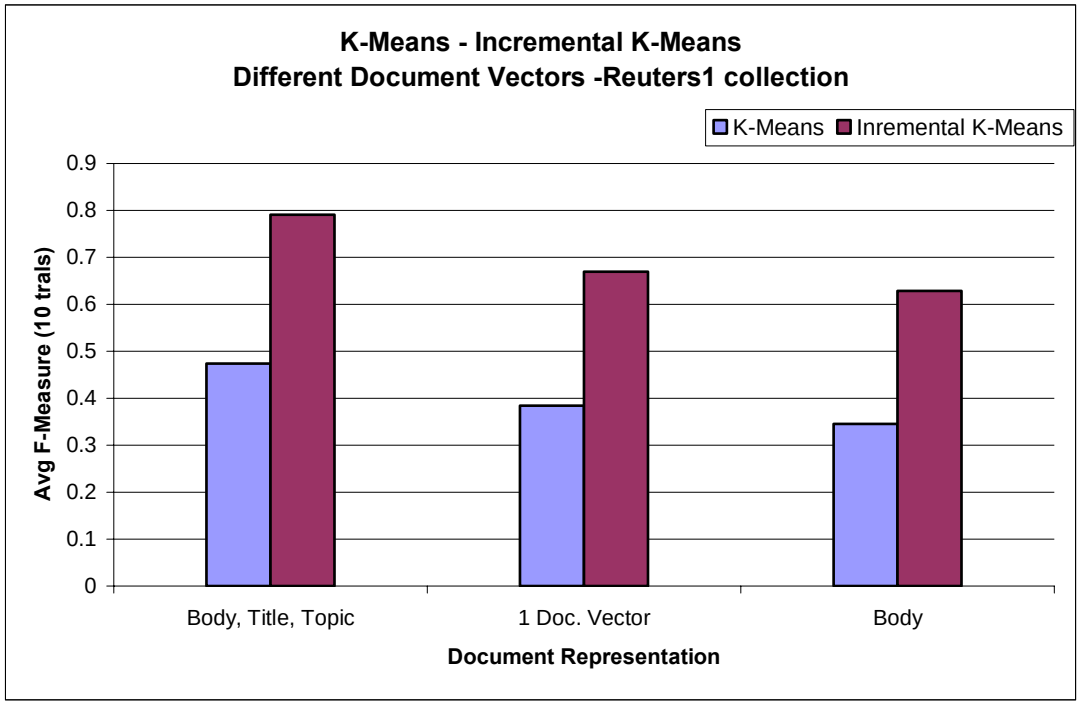

Figure 4.3: Experiment 1a – Experiments varying document vector representations

Incremental K-Means performs significantly better than basic K-Means algorithm. We can see that independently of the type of the document representation Incremental K-Means outperforms the standard implementation of K-Means method by 20-32%. The performance of Incremental K-Means method fluctuates between 62% and 80%, whereas F-Measure values for K-Means are between 34% and 47%. The results indicate that the continuously center adjustment and the last re-assignment

of the documents to clusters, (as suggested in section 3.3.2), produce better clustering results than the naive K-Means procedure.

Regarding different document vector representations, Figure 4.2 illustrates that for both algorithms the best F-Measure values are obtained in case of documents are represented by three distinct vectors (body, title and topic), instead of a single unified vector. This observation was expected due to the structure of the reuters1 test set. The likelihood that a document will be assigned to the correct cluster increases when the topic field is included in the vector. Notice that, in Reuters-21578 the topic of each document is used to classify it in a pre-defined category. Also because the topic determines the class that a Reuters document belongs to, having the topic in a separate document vector would be unfair to the other two cases presented in Figure 4.3 (this would increase the performance of algorithm drastically). For this reason we decided to use topic only as part of a single vector, together with all other fields and not as a separate vector. In the following each document in Reuters-21578 is represented by a single vector formed by all distinct terms from title, body ant topic.

We indicated that Incremental K-Means is much more effective than regular K-Means in terms of F-Measure. In the second experiment (experiment 1b), we evaluate the performance of Incremental K-Means under different values of the number ITER of iterations. The number of iterations of continuous center adjustment examined is 1, 2, 3, and 4. Similarly to experiment 1a, we used reuters1 subset and set K equal to 24, as 24 is the number of the hand-labeled classes in this set. Figure 4.4.indicates the quality of the clustering solutions produced by Incremental K-Means algorithm setting different number of iterations

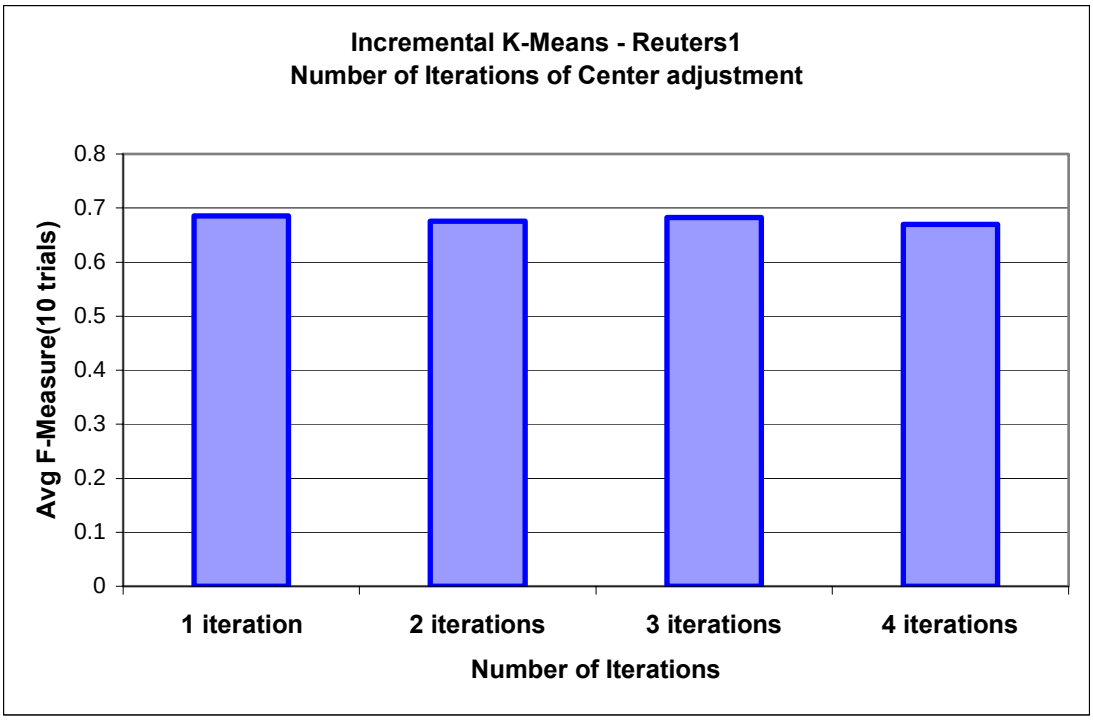

Figure 4.4: Experiment 1b – Examine the number of iterations of continuous center adjustment

As we can see in Figure 4.4, F-Measure is relatively independent on the number of iterations. Incremental K-Means method showed a stabilized accuracy at about 68% regardless of the number of iterations. The most significant observation is that it is sufficient only a single iteration of center adjustment to produce equally good partitions. This is an important conclusion, because we can efficiently reduce the clustering time of our Incremental K-Means technique. It is obvious that quadruple time could be required in case of 4 iterations as compared to a single iteration. While this time comparison may not be noticeable for small data sets like reuters1, it becomes much more significant for clustering on large document collections. Notice that [32] also examined the effects of the number of iterations of centroid adjustment in clustering quality. They also suggested that multiple iterations are not necessary.

Evaluation of Bisecting Incremental K-Means algorithm

Given the good performance of Incremental K-Means algorithm, we then examined its performance against our proposed Bisecting Incremental K-Means clustering technique. We compared it to K-Means and Incremental K-Means methods.

Prior to the evaluation of our Bisecting algorithm we run the following experiment. We investigated the behaviour of our hierarchical algorithm under different values of K in Incremental K-Means method. As described in section 3.3.3, repeated applications of Incremental K-Means algorithm produces a hierarchy of clusters. Different cluster hierarchies are produced with different values of K. Notice that different K affects the number of clusters that a given cluster is split and therefore the higher the value of K the lower the depth of the hierarchy. The larger the value of K, the broader and shallower is the resulting hierarchy.

To conduct this evaluation (experiment 1c), we set the K equal to 2, 10 and 25 and used the reuters1 test set. Incremental K-Means method terminated as each leaf cluster contained a single document. According to the results in experiment 1b (see Figure 4.4), at each bisecting step the parameter ITER in Incremental K-Means technique was set to 1. F-Measure scores for the various values of K are shown in Figure 4.5, whereas the Figure 4.6 reports a comparison of the Incremental K-Means with different K (various-secting clustering at each step) in terms of clustering time (experiment 1d).

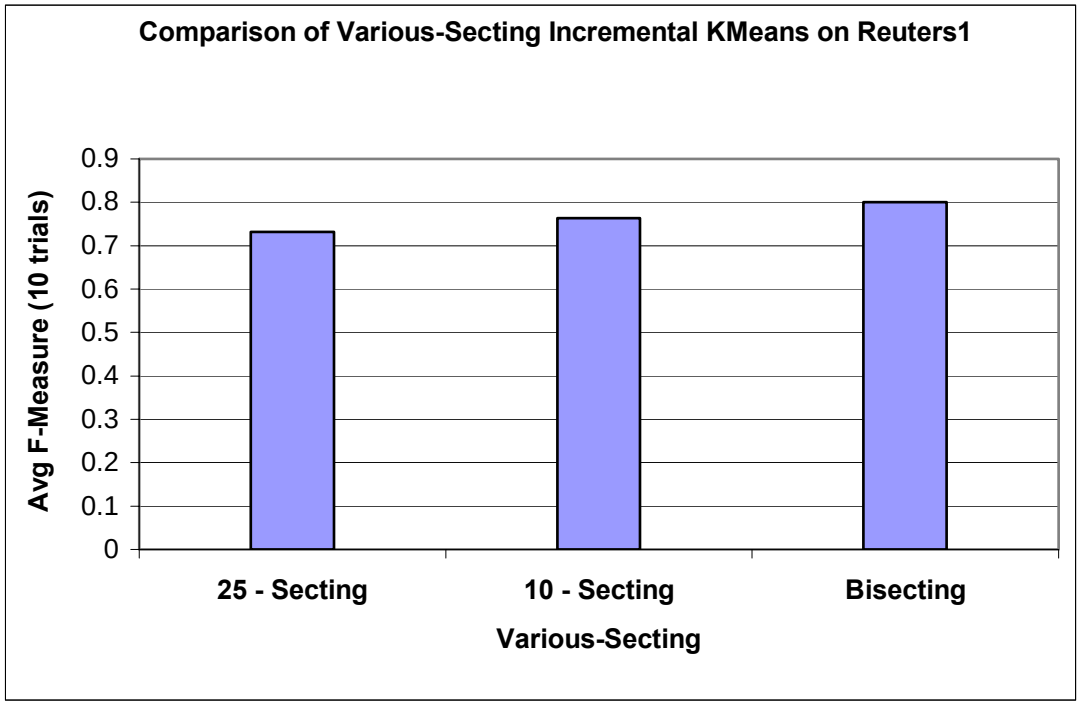

Figure 4.5: Experiment 1c - Various Secting Incremental K-Means – F-Measure

As we can see in Figure 4.5, the F-Measure of the generated cluster hierarchy increases as the value of K decreases. Notice that F-Measure is rather independent on

K (increases slightly for K=2). The best F-Measure score was obtained in Bisecting Incremental K-Means method. However, there are no significant differences in the effectiveness of clustering results using one of the three values of K.

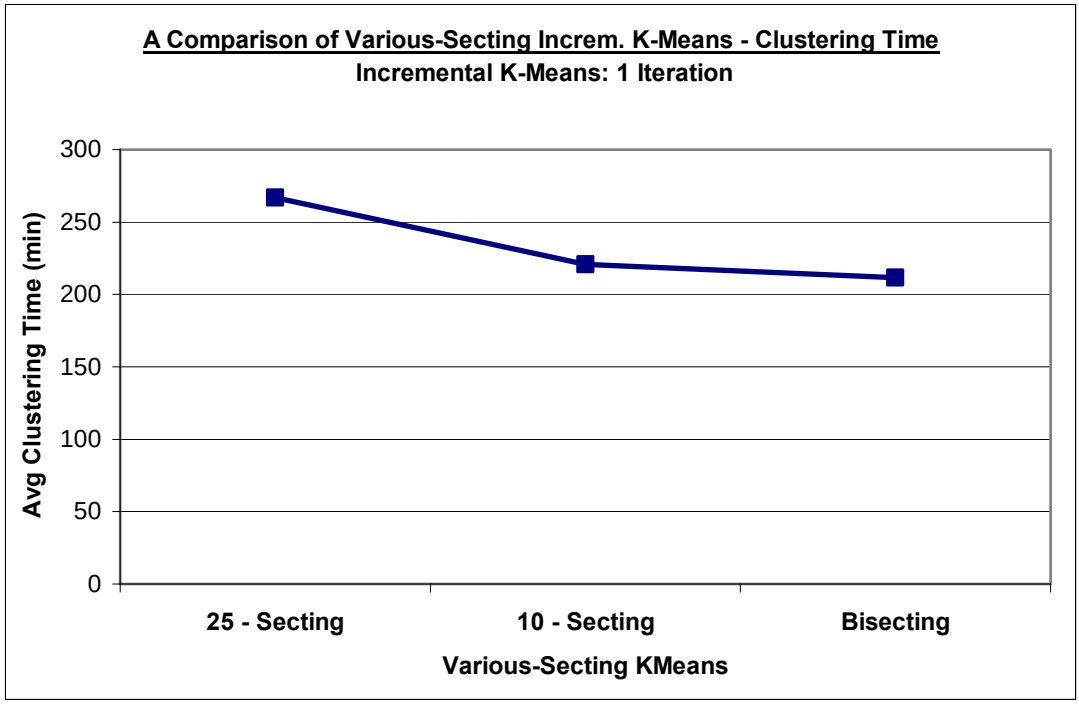

Figure 4.6: Experiment 1d - Various Secting Increm. K-Means – Clustering time

As far as the clustering time is concerned, Figure 4.6 shows that the time (minutes) required building a cluster hierarchy increases with K. Thus, bisecting is the approach which requires the less clustering time.

We conclude that results in experiment 1c and 1d confirmed our initial decision to apply our proposed methodology on Bisecting Incremental K-Means algorithm, instead of other K-Secting techniques**.**

The main goal of the following document clustering experiment (experiment 1e) is to evaluate Bisecting Incremental K-Means algorithm and compare its performance against K-Means and Incremental K-Means. In this experiment (experiment 1e) we used reuters1 and ohsumed1 (see Table 4.4). As far as Bisecting Incremental K-Means is concerned, the experiments were done by using the same parameter values discussed in experiment 1c regarding the number ITER of iterations at each bisecting step and the terminating procedure. In K-Means and Incremental K-Means algorithms, the number of iterations was set equal to 6 and 1 respectively.

Figure 4.7 shows the results from the comparison of the three clustering methods in terms of clustering quality.

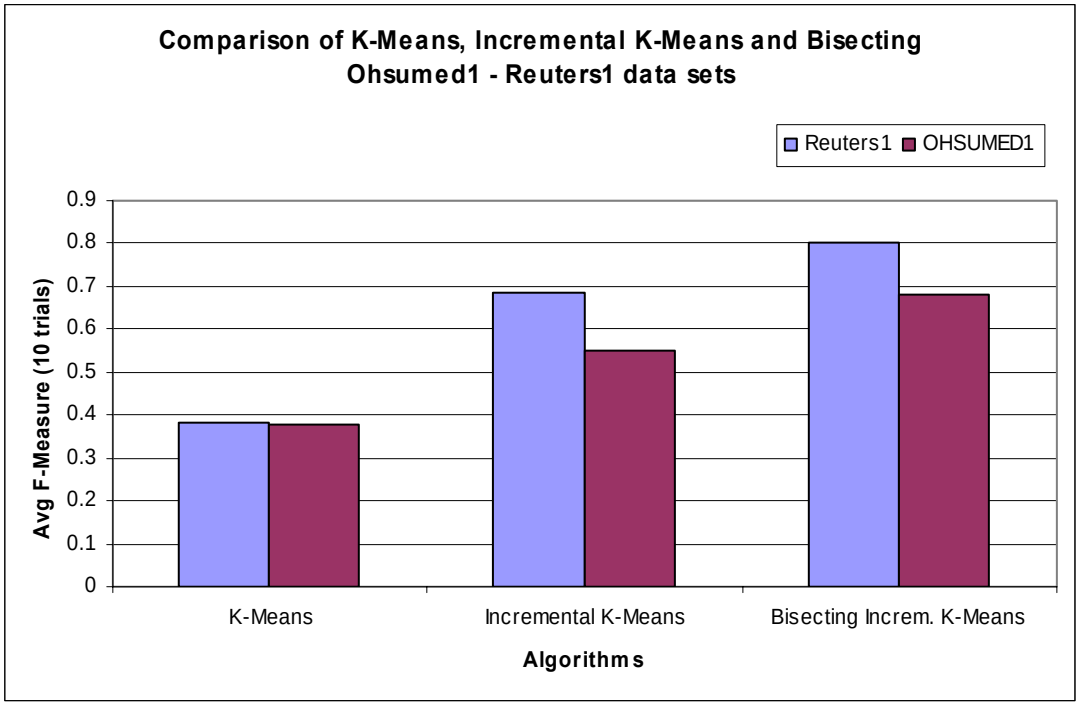

Figure 4.7: Experiment 1e – Comparison of K-Means, Incremental K-Means and Bisecting Incremental K-Means

Figure 4.7 illustrates that our Bisecting Incremental K-Means algorithm achieves significantly better F-Measure than the other two clustering methods on both data sets (reuters1 and ohsumed1). Specifically, our Bisecting algorithm outperformed the basic K-Means and our Incremental version by 42% and 12% respectively on reuters1 and by 28% and 13% respectively on ohsumed1.

As we can see in Figure 4.7, the resulting F-Measure values for K-Means and Incremental K-Means on ohsumed1 are consistent with those obtained on reuters1 and presented in Figure 4.3. We observe that on both data sets the Incremental method performs noticeably better than the standard K-Means algorithm.

Finally, experimental results in Figure 4.7 indicated that for each one of the three evaluated algorithms the F-Measure score was less on ohsumed1 as compared to the corresponding value obtained on reuters1. Thus, we could make the supplementary conclusion that in document clustering evaluation the OHSUMED collection gives lower F-Measure values as compared to Reuters-21578 data set. Results in [63] confirm this observation.

Summarizing, Bisecting Incremental K-Means algorithm performs significantly better than K-Means and its variant Incremental K-Means on both reuters1 and ohsumed1. Incremental K-Means outperforms basic K-Means by 17-30% in terms of F-Measure. Incremental K-Means also requires only a single iteration for center adjustment to produce a good clustering solution. Finally, in experiment 1c we observed that our version of Bisecting Incremental K-Means performs better than the other Various-Secting algorithms.

4.5.3 Evaluation of MeSH based Representation on Clustering Quality

We showed that Bisecting Incremental K-Means method outperforms the other two partitional clustering techniques on both data sets. In this experiment, we evaluated how the clustering quality is affected by the way the documents are represented.

We examined the performance of Bisecting Incremental K-Means method using vector representation of documents consisting of MeSH terms. Then, we compared these results with those obtained by representation with single word terms. The latter approach was evaluated in subsection 4.5.2 on reuters1 and ohsumed1. In this experiment, we used the ohsumed2 data set which contains 10902 documents. We selected an OHSUMED subset, as it is a medical corpus which contains articles from Medline published. As described in section 4.2.2, 10-12 MeSH terms are assigned to each OHSUMED document by human indexers and constitute a specific field.

MeSH terms are extracted from the title and the abstract field of each Medline reference using the technique described in section 4.3. Then, we added the existing (within each document) MeSH terms to obtain a vector of MeSH terms for each OHSUMED document.

To conduct this experiment (experiment 2a), we used ohsumed2 subset (see table 4.3). The number of parameter ITER in Incremental K-Means method was set equal to 1 and the divisive procedure terminated when each leaf cluster contained a single document. Figure 4.8 shows the F-Measures scores obtained by using MeSH and single word terms to represent the OHSUMED documents. In terms of clustering time the evaluation is shown in Figure 4.9. We called this experiment 2b.

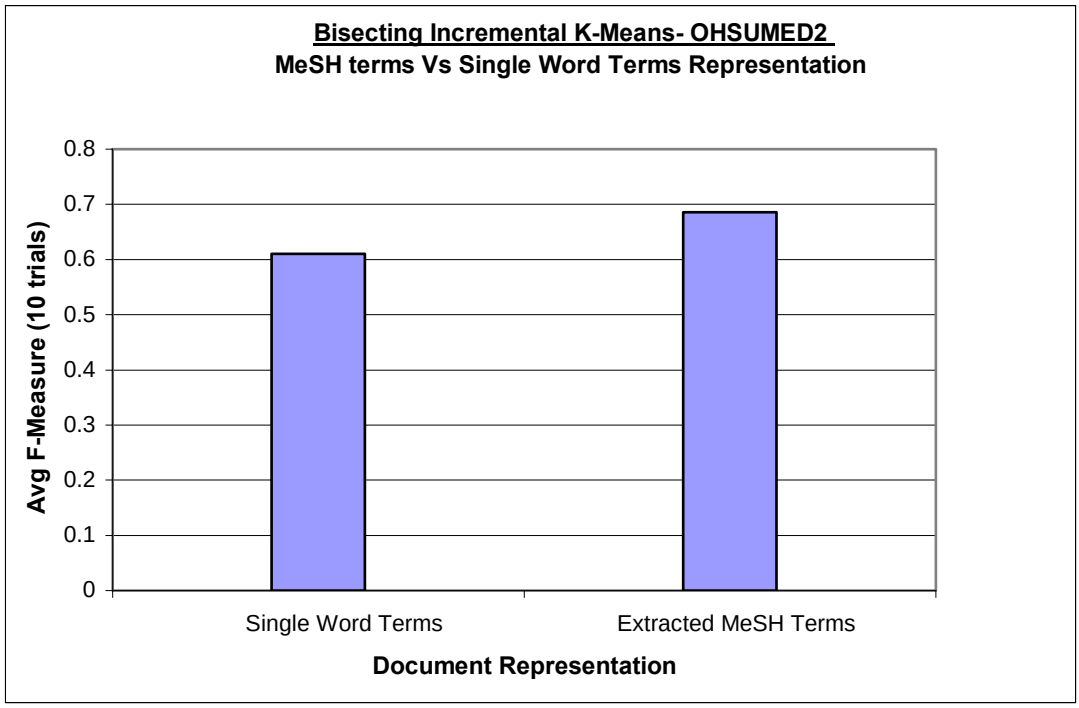

Figure 4.8: Experiment 2a – F-Measure corresponding to Bisecting Incremental K-Means. Document representation with single word terms and MeSH terms

We observe that our Bisecting Incremental K-Means method yields better F-Measure when the OHSUMED documents represented by MeSH terms rather than by single word terms. Figure 4.8 indicates an 8% increase in F-Measure.

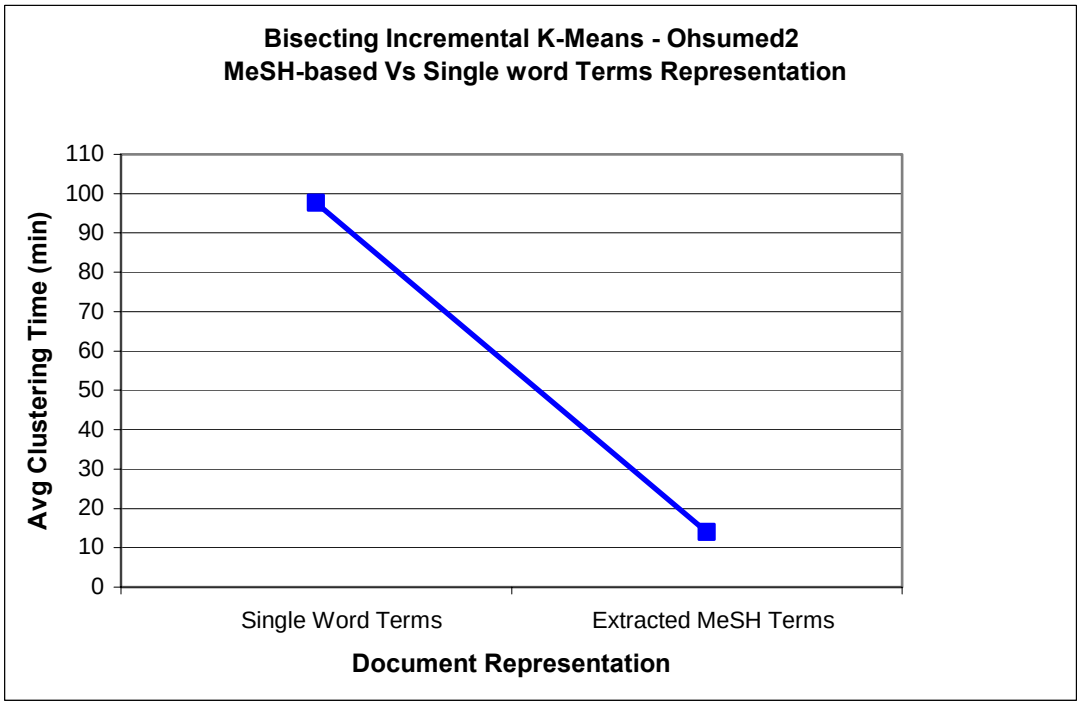

Figure 4.9: Experiment 2b – The effects of document representation on clustering quality in terms of Clustering Time

In terms of clustering time, experimental results in Figure 4.9 show that our version of Bisecting algorithm needs much less execution time when the document vectors contain only MeSH terms. On ohsumed2 the clustering time for the MeSH-based representation of documents was 14 minutes whereas for single word terms representation the clustering hierarchy was obtained in 97.6 minutes. Therefore, significant decrease in execution time was observed. This happened due to the size of the document vectors. In case of the document representation by single word terms the size of each document vector is about 80-100 terms, while in case of MeSH based representation each vector contains about 20 distinct MeSH terms.

Summarizing, the MeSH-based representation of documents as compared to single word terms representation improves the performance of our Bisecting Incremental K-Means algorithm. The results showed that the F-Measure increases while the clustering time decreases notably. Moreover, MeSH terms form a more meaningful representation for documents and clusters. The set of MeSH terms contained in each document specifies well the subject of the document. In case of clusters the centroid is consisted of MeSH terms and can satisfactorily gives the semantic content of the cluster.

4.5.4 Evaluation of BIC-Means - Experiments on BIC

In section 3.5, we proposed the BIC-Means, a hierarchical clustering algorithm based on Bisecting Incremental K-Means method. This set of experiments focused on evaluating the quality of the hierarchical clustering solution produced by BIC-Means. We examined the use of BIC and evaluated how the clustering quality is affected by the proposed technique for terminating the divisive procedure.

The performance of BIC-Means was evaluated in terms of clustering quality and clustering time. The values obtained in this experiment were compared to the corresponding results of Bisecting Incremental K-Means method where the procedure terminates as each leaf cluster contains a single document. To conduct this evaluation the document collections were selected are ohsumed2, reuters1 and reuters2. With regard to ohsumed2, we used vector representation of documents based on MeSH terms. Experiments in subsection 4.5.3 showed that this representation outperforms the representation by single word terms. At each bisecting step the parameter ITER was set to 1.

The resulting F-Measures for BIC-Means and Bisecting Incremental K-Means are presented in Figure 4.10 (experiment 3a). For each of the three test corpora the corresponding scores are compared. The comparison in terms of clustering time (experiment 3b) is shown in Figure 4.11.

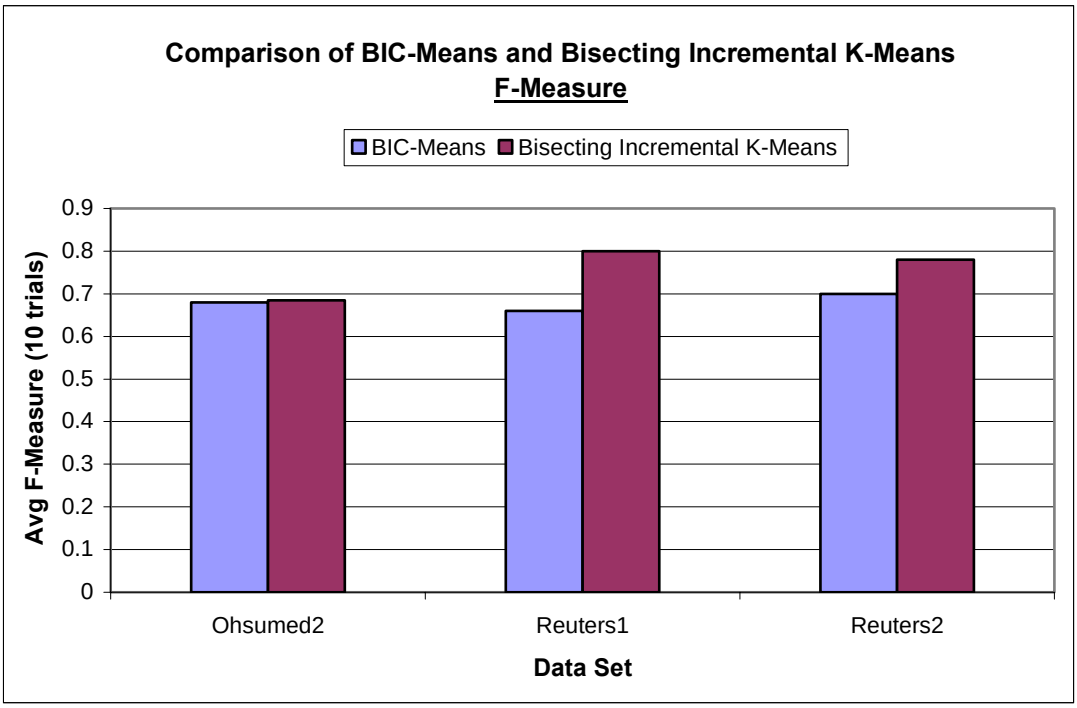

Figure 4.10: Experiment 3a – Comparison of BIC-Means and Bisecting Incremental K-Means on clustering quality

As we can see in Figure 4.10, in ohsumed2 the proposed BIC-Means algorithm achieved similar F-Measure value as Bisecting Incremental K-Means method. Results on reuters1 and reuters2 indicated that BIC-Means performed slightly worse as compared to initial Bisecting approach. In terms of F-Measure, for reuters1 the decrease in clustering quality was 14%, whereas in reuters2 collection was 8%.

Thus, we observe that BIC-Means does not yield better F-Measures values than Bisecting Incremental K-Means. However, these results were prospective. From the beginning BIC-Means was not expected to improve the clustering quality of our basic Bisecting technique as the last is exhaustive producing the entire clustering hierarchy (terminating in singleton clusters) while BIC-Means was introduced here as a means for non-exhaustive clustering aiming at terminating at rather meaningful clusters. However, the performance sacrifices compared to Bisecting Incremental K-Means is negligible. As described in section 3.5, BIC-Means expands the

functionality of Bisecting Incremental K-Means method. It uses BIC as the splitting criterion of a leaf cluster and then, a strategy is applied to terminate the divisive procedure.

We can conclude that the basic advantage of BIC-Means is the automatic way for terminating the Bisecting technique rather than executing the algorithm until each leaf cluster contains a single document. Figure 4.10 indicates that on the three test corpus F-Measure values of BIC-Means decreased slightly or were the same as compared to the corresponding values of Bisecting Incremental K-Means. Only on reuters1 is observed a high decrease in F-Measure score. This can be explained as follows. First, the BIC which is used as the splitting criterion of a cluster needs a large collection in order its application to be more effective. In our case, reuters1 contains 2442 documents. As a result, the use of BIC in the specific collection produced a small number of clusters and thereby F-Measure score was decreased as compared to initial Bisecting algorithm. Contrary to Reuters, on ohsumed2 BIC-Means achieves F-Measure value similar to this obtained by our initial Bisecting approach. This is because OHSUMED is fairly big data set and the hierarchy obtained by BIC-Means was quite deep due to the many bisections done.

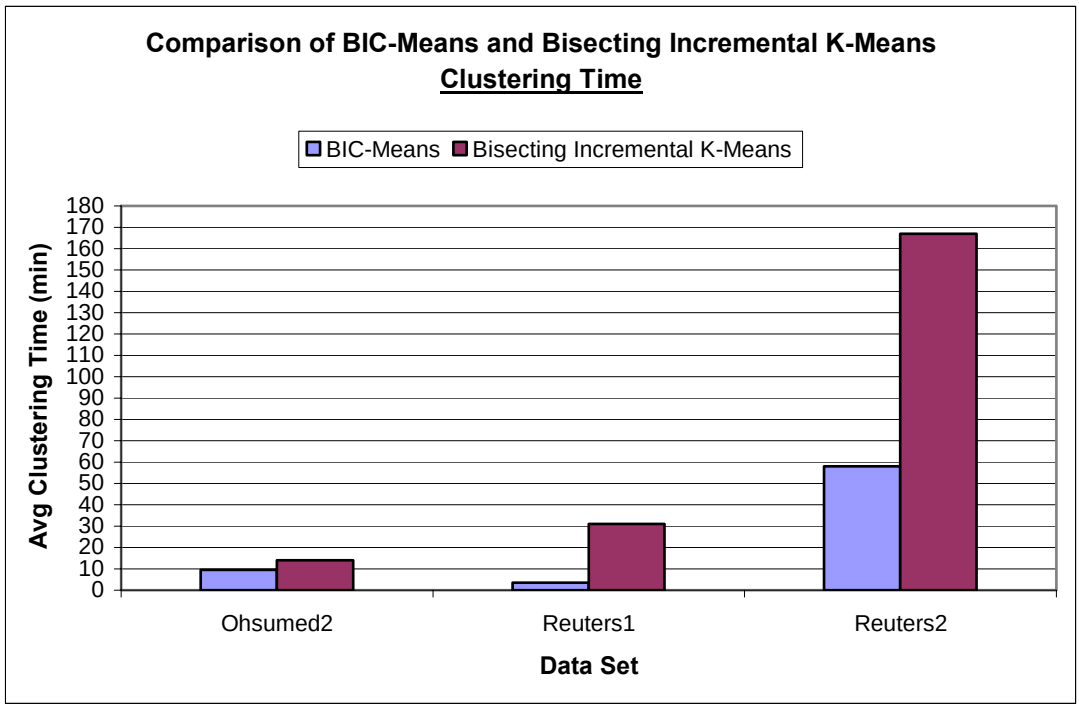

Figure 4.11: Experiment 3b – Comparison of BIC-Means and Bisecting Incremental K-Means on clustering time

In terms of clustering time, the experimental results in Figure 4.11 indicate that BIC-Means runs much faster than Bisecting Incremental K-Means method on both document collections. For the three test collections were used in this set of experiments it took much more time for basic Bisecting technique than proposed BIC-Means algorithm to build a hierarchy of clusters. On reuters1 the clustering time of BIC-Means was about 9 times less than that of initial Bisecting method (3 - 31.5 minutes). On reuters2 the clustering time of BIC-Means was about 3 times less as compared to the basic Bisecting approach (58 - 167 minutes). Finally, on ohsumed2 it took 9.5 minutes for BIC-Means to produce the cluster hierarchy, whereas Bisecting Incremental K-Means required 14 minutes.

Regarding reuters1 the too much difference in clustering time can be explained due to the small number of bisections applied on this subset. We discussed this fact earlier in this subsection. Thus, the algorithm terminated much more quickly and had a small decrease in clustering quality as compared to basic Bisecting method. For ohsumed2 the clustering time was short because only 20-25 MeSH terms were contained at document vectors.

Summarizing, BIC-Means is a hierarchical clustering approach which incorporates a strategy for terminating the divisive procedure. Its main advantage is that requires significantly less time to run compared to Bisecting Incremental K-Means method. Thus, it is an appropriate algorithm for clustering very large document collections since it does not execute the procedure exhaustively. Finally, the automatic way that BIC-Means uses to stop the algorithm keeps F-Measure scores at the same levels or causes a slight decrease in clustering quality as shown in Figure 4.10.

4.5.5 Summary of Document Clustering Experimental Results

In this section we presented our experiments and results on document clustering. This evaluation revealed strengths and weakness of the different clustering algorithms implemented in this study. First in our experiments we evaluated and compared the clustering quality of K-Means and Incremental K-Means. F-Measures scores were computed for different vector representations of documents on reuters1. The results showed that Incremental K-Means yielded noticeably better F-Measure values than K-Means. Additionally, we showed that Incremental K-Means needs only a single iteration of center adjustment to produce a good clustering partition.

Then, on reuters1 and ohsumed1 we compared the Bisecting Incremental K-Means with basic K-Means and Incremental K-Means. The results indicated that Bisecting Incremental K-Means performs much better than the two other techniques on both data sets.

We continued our evaluation by examining the performance of Bisecting Incremental K-Means method using MeSH-based representation of documents on ohsumed2. We compared these results with those obtained by using single word terms to represent a document. The comparison showed that MeSH-based representation improves significantly the performance of Bisecting Incremental K-Means algorithm in terms of F-Measure and clustering time.

Finally, we evaluated (on three data sets) the quality of the hierarchical clustering solution produced by BIC-Means algorithm. BIC-Means incorporates a strategy to stop the divisive procedure. We computed F-Measure scores and clustering time and then compared them to the corresponding values obtained from Bisecting Incremental K-Means method which is executed exhaustively. Experimental results indicated that BIC-Means requires much less time to build a cluster hierarchy as compared to initial Bisecting approach (see Figure 4.11). This is important in case of large document collections. In terms of F-Measure, BIC-Means achieves the same or slightly decreased values as compared to Bisecting Incremental K-Means algorithm.

4.6 Retrieval using Document Clusters

In the following we demonstrate that it is possible to apply clustering to reduce the size of the search (and therefore retrieval response times) on large data sets. We propose several cluster-based retrieval strategies and evaluated their performance. In parallel, we examined the use of MeSH terms in document, cluster and query vector representation.

The majority of the document retrieval systems which have been described in the literature match the query against documents in the entire collection. They do an exhaustive search (document-based retrieval). Similarity scores between the query and each document are computed and the documents are then ranked in order of decreasing similarity with the query. However, the computation of similarities between user's request and all the documents is time consuming due to the exhaustive

search is done. For this reason an alternative approach is required, mostly for retrieval on large document collections.

Below, we examined how this goal could be achieved by incorporating of document clustering into the information retrieval process (cluster-based retrieval). Cluster-based retrieval incorporates the application of a clustering technique on a document collection in order to group documents into clusters, matches the query with a representative representation of each cluster and then ranks clusters based on their similarity to the query. We search the documents which are contained in the N top-ranked clusters and not all the documents exhaustively. This is the general approach of the examined retrieval strategies based on clusters. We evaluated the efficiency and effectiveness of cluster-based retrieval as compared to exhaustive retrieval method.

A number of studies [26], [46], [47] have been proposed in the literature on applying clustering to improve retrieval results. Some experimental results [5], [26] have shown that cluster-based retrieval using static clustering outperforms retrieval by exhaustive search. Other results [58] have indicated that exhaustive retrieval is generally more effective.

In most experiments the size of document collections used was small. This is due to the time and space performance of hierarchical clustering approaches. There are no conclusive results on large data sets. In this study, we examined how cluster-based retrieval can perform across collection of realistic size. Experimental results in subsection 4.5.4 showed that BIC-Means can be applied on large document collections. As described in the following subsection, in our evaluation in addition to documents, the queries contained only MeSH terms. We examined how MeSH-based document and query representation affect cluster-based retrieval.

Jardine and van Rijsbergen [26] first suggested that the associations between documents contain information about the relevance of documents to user's requests. They formulated and examined the cluster hypothesis. "Closely associated documents tend to belong to the same clusters and are expected to be relevant to the same queries". Correspondingly, dissimilar documents are unlikely to be relevant to the same requests.

4.6.1 Cluster-based Retrieval on OHSUMED using MeSH

We examine the cluster-based retrieval on OHSUMED which contains 233445 Medline articles. All documents include abstract. In order to build the document vectors we extracted MeSH terms from title, abstract and MeSH terms fields. The MeSH term extraction technique was presented in section 4.3. We chose the MeSH based representation due to the clustering results presented in subsection 4.5.3. They indicated that MeSH terms improve the performance of our Bisecting algorithm in terms of F-Measure and clustering time. Additionally, using MeSH terms much less time is required to compute similarities between the documents or clusters and the query due to the small size of document or cluster term vectors. OHSUMED documents were indexed by the Lucene utility. The weights of all MeSH terms in OHSUMED documents are computed by $tf - idf$.

The experiments required that documents be first organized into clusters. We applied our proposed BIC-Means algorithm on entire OHSUMED and a static hierarchy of clusters was produced. Each cluster was represented by the centroid vector which is the vector obtained by averaging the weights of the various terms in cluster.

To examine the retrieval performance a test collection of 106 queries was used. A group of novice physicians generated these queries using Medline. Each document has been judged by physicians as relevant, possibly relevant or not relevant to a query. In our experiment we consider the possibly relevant documents as relevant. Each OHSUMED query contains patient and topic information, in the format:

.I Sequential identifier

.B Patient Description

.W Information Request

We present an example of a query:

. .I 6

.B 55 yo female, postmenopausal

.W does estrogen replacement therapy cause breast cancer

In our evaluation the MeSH terms are extracted from each query in order to represent it. We used the extraction technique presented in section 4.3. The reason for this extraction was the MeSH based-representation of OHSUMED documents. Each

MeSH term had a unique participation at each query. After this process the above query was converted as follows:

"Breast_neoplasms female estrogen_replacement_therapy"

The words which are connected with "_" constitute a MeSH term.

In our experiments we ignored some queries from the OHSUMED query set due to three reasons. First, several queries do not contain MESH terms so that to extract them and represent the corresponding query. Second, a query does not have relevant documents in the judged pool. Finally, some queries were removed from the set because in an initial exhaustive search done no relevant documents were retrieved for these queries. As a result we removed 45 queries. The final query set consisted of 61 queries. Each query contained between 1 and 6 MeSH terms. Section A.4.1 of the Appendix A shows the 61 OHSUMED queries while section A.4.2 illustrates their corresponding MeSH-based representation. Apart from the original OHSUMED query set developed by Hersh et al, a sub-set of 63 queries were used in TREC-9[9] (Trec Retrieval Conference) IR experiments. Similarly to original queries, relevance judgements provided by NIST (National Institute of Standards and Technology) determine OHSUMED relevant documents to each query. We observed that 40 of the 61 queries used in our retrieval experiments are contained in TREC-9 query set.

Vector Space Model (VSM) was used for retrieval of documents in OHSUMED. This state-of-the-art method uses the classic dot product between centroids of clusters and queries as the matching function. The retrieval system was built upon Lucene. Notice that in addition to text indexing, Lucene is a full-featured text search engine library in Java. All retrieval strategies were implemented on top of Lucene. The weight of each query term was initialized to 1, because a MeSH term can be contained only once in a query. Each query retrieved the 100 highest ranked answers due to the tendency of users to examine only the top-ranked documents retrieved by the system.

As presented in the following subsection we examined several search strategies. In all cases in order to find the clusters that best match a query we searched the bottom-level clusters (leaf clusters). Experimental results in [5], [18], [46] indicated that this method instead of searching all the clusters of the hierarchy gives the best retrieval results. The 633 leaf clusters of our produced hierarchy are non-

[9] http://trec.nist.gov/data/t9_filtering.html

singleton clusters due to the stopping strategy we use in BIC-Means algorithm. They were scanned and the most relevant to the query were retrieved according to the corresponding search strategy.

Efficiency and effectiveness are usually the measures used for the evaluation of an IR system. The former measure looks at the time and space requirements of the algorithms used by the system. It checks operations such indexing and searching in terms of functionality. On the other hand, the effectiveness of an IR system addresses the quality of the retrieval results. The measures used to examine the effectiveness of our retrieval system are recall (R) (the ratio of relevant documents that are retrieved) and precision (P) (the ratio of retrieved documents that are relevant). We computed averaged precision which is the value of precision averaged over the 61 queries and averaged recall which is the value of recall averaged over the 61 queries.

4.6.2 Experimental Results: Precision/Recall and Evaluation

As cluster hierarchy has been built, a search for the clusters that best match the query was done. We introduced several retrieval strategies which are based on the bottom-level clusters of the hierarchy (leaf clusters). Each search strategy incorporates different criteria in order to match the query against leaf clusters and retrieve them. We evaluated the effectiveness and efficiency of each of these search strategies and compared the results with the retrieval results obtained by exhaustive search on OHSUMED.

The results of each method are represented by a precision/recall curve. Each point on a curve is the average precision and recall over all queries. As mentioned we selected the 100 highest ranked answers for each query, so the precision/recall plot of each method contains exactly 100 points representing the average precision and recall over the 61 queries. Precision and recall values are computed from each answer set after each answer. The top-left points of a precision/recall curve corresponds to the precision/recall values for the best answer or best match while, the bottom right point corresponds to the precision/recall values for the entire answer set. A method is better than another if it achieves better precision and recall.

First in our experiments we performed an exhaustive search (document-based retrieval) on all OHSUMED documents. Similarities between each query and each document were computed. Then, the documents were ranked and a list of the top 100

documents was produced. The generated precision/recall curve compared with all the cluster-based retrieval strategies are suggested below.

Experiment 1: Retrieve and Search the N highest Ranked Clusters

We start describing the most simple of the implemented cluster-based retrieval strategies. Each leaf cluster was represented as a vector of MeSH terms. We computed the similarity between the query and each leaf cluster. The clusters were ranked based on their similarity to the query. The issue examined in this strategy was how many best match clusters to select in order to search the documents of that clusters. We evaluated seven cases. To select 1, 3, 10, 30, 50, 100 and 150 of the top-ranked leaf clusters. We chose these values as considered to be representative in indicating the behaviour of the retrieval system. We called these retrieval strategies top_1Cluster, top_2Clusters, top_10Clusters, top_30Clusters, top_50Clusters, top_100Clusters and top_150Clusters.

In each case, the documents of the selected clusters were collected. Thus, a new document collection was produced. It was a very small subset of the initial data set. Any document in this subset was considered more likely to be relevant to the query than documents from clusters ranked lower and were not contained in the selected list of clusters (1, 3, 10, 30, 50, 100 or 150). Then, we computed the similarity between the queries and the documents of the produced collection. The documents were order by decreasing similarity. We selected the 100 highest ranked documents for each query to evaluate precision and recall.

Figure 4.12 shows the averaged precision and recall values obtained by these retrieval experiments. For each evaluation (1, 3, 10, 30, 50, 100 and 150 top-ranked clusters) we present a precision/recall curve. We compare these curves with the retrieval result by exhaustive search (document-based retrieval).

As we can see in Figure 4.12, document-based retrieval performed better than all the examined cases of cluster-based retrieval. The best results of cluster-based retrievals obtained as the number of the selected clusters was 150 and 100. We observe in Figure 4.12 that top_100Clusters and top_150Clusters strategies achieve almost the same precision and recall. Notice that performance improves with N (in this experiment N=150 and N=100 achieve better precision and recall). The reason for this behavior is that more relevant documents are revealed even within less similar

clusters and their number of similar documents increases with N. Notice though, that as N approaches K (total number of clusters), the cluster-based retrieval approaches exhaustive search. The overall performance of cluster-based retrieval depends on whether top-ranked clusters contain relevant documents.

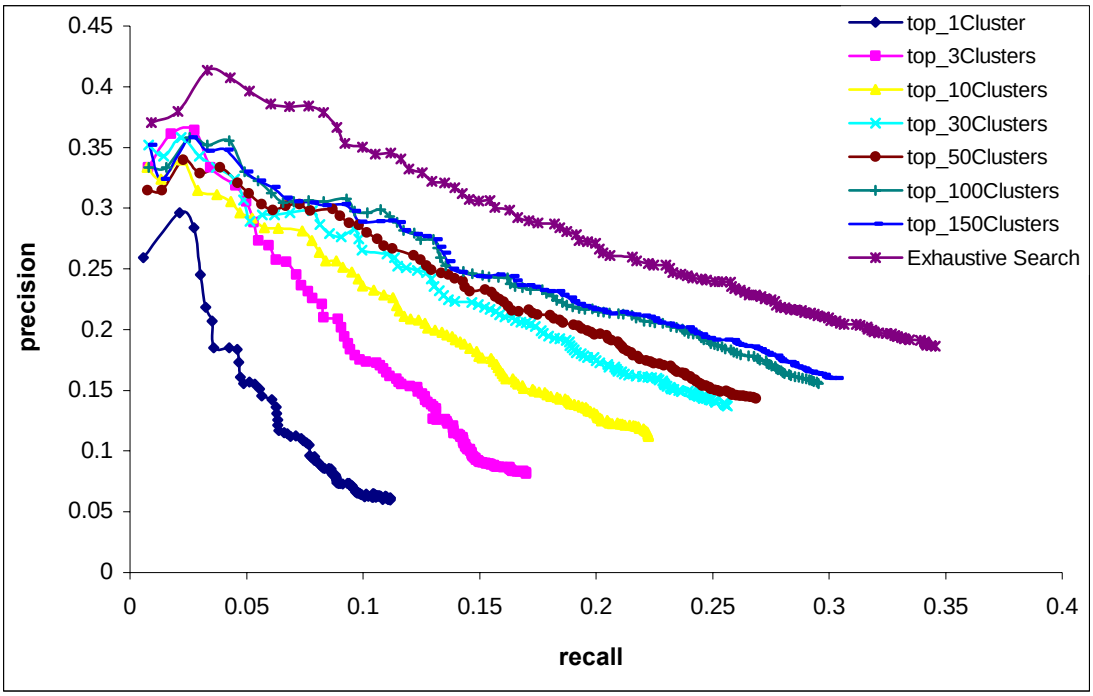

Figure 4.12: Precision-recall diagrams of exhaustive search on OHSUMED and cluster-based retrieval strategy using the n top-ranked clusters for retrievals

The better performance of document-based retrieval is reasonable due to the smaller number of documents contained in the top 150 ranked clusters and the other cases of top clusters. We counted that in top_150Clusters experiment the averaged number of documents searched over the 61 queries was only 88806, while the corresponding number in top_100Clusters experiment was 67648. On the contrary, in document-based retrieval were exhaustively searched 233445 articles. As a result, in case of cluster-based retrieval experiments the number of similarity computations between the query and the documents was significantly decreased. On the other hand, we observe that retrieval by exhaustive search as compared to top_150Clusters and top_100Clusters retrieval strategies achieved about up to 5% better precision and up to 5% better recall. We conclude that top_100Clusters and top_150Clusters retrieval methods improve noticeably the computation efficiency of the retrieval while the

effectiveness was slightly decreased as compared to retrieval results by exhaustive search on OHSUMED documents.

Experiment 2: Use the 20 Highest Weighted MeSH terms of the Centroid and Search the N Top-Ranked Clusters

The first retrieval experiment used all the centroid terms to compute the similarity scores between the clusters and the query. The second set of experiments examined the effectiveness of cluster-based retrieval using the 20 highest weighted MeSH terms of the centroid. The centroid terms were sorted by decreasing frequency and the top 20 MeSH terms were selected to represent the cluster. Then, the clusters were ranked by decreasing similarity with the query. In this experiment we evaluated four cases. To select the 10, 50, 100 or 150 of highest ranked clusters. We called these retrieval strategies 20Terms-top_10clusters, 20Terms-top_50clusters, 20Terms-top_100clusters and 20Terms-top_150clusters. Similarly to first set of retrieval experiments, we computed the similarity between the queries and the documents contained in the top 10, 50, 100 or 150 ranked clusters. We used the cosine similarity function to match each query against documents of top retrieved clusters. A ranked document list was produced for each experiment. We selected the 100 highest ranked documents to evaluate the retrieval process.

Figure 4.13 illustrates the precision-recall curves for the methods tested in these retrieval experiments. We compare them with the document-based retrieval (exhaustive search) on OHSUMED and the top_150Clusters retrieval strategy presented in Experiment 1 (first set of retrieval experiments).

Figure 4.13 indicates that document retrieval by exhaustive search on OHSUMED is more effective than the search based on leaf clusters and use the 20 highest weighted MeSH terms of the centroid. Comparing the precision-recall diagrams obtained in the first set of experiments (Experiment 1) with them illustrated in Figure 4.13 we observe that the use of the 20 highest weighted MeSH terms of cluster's centroid did not improve the cluster-based retrieval results on OHSUMED. More specifically, Figure 4.13 shows that top_150clusters retrieval method (uses all the MeSH terms of the centroid) performs better than 20Terms-top_150Clusters (uses the 20 highest weighted MeSH terms of the centroid).

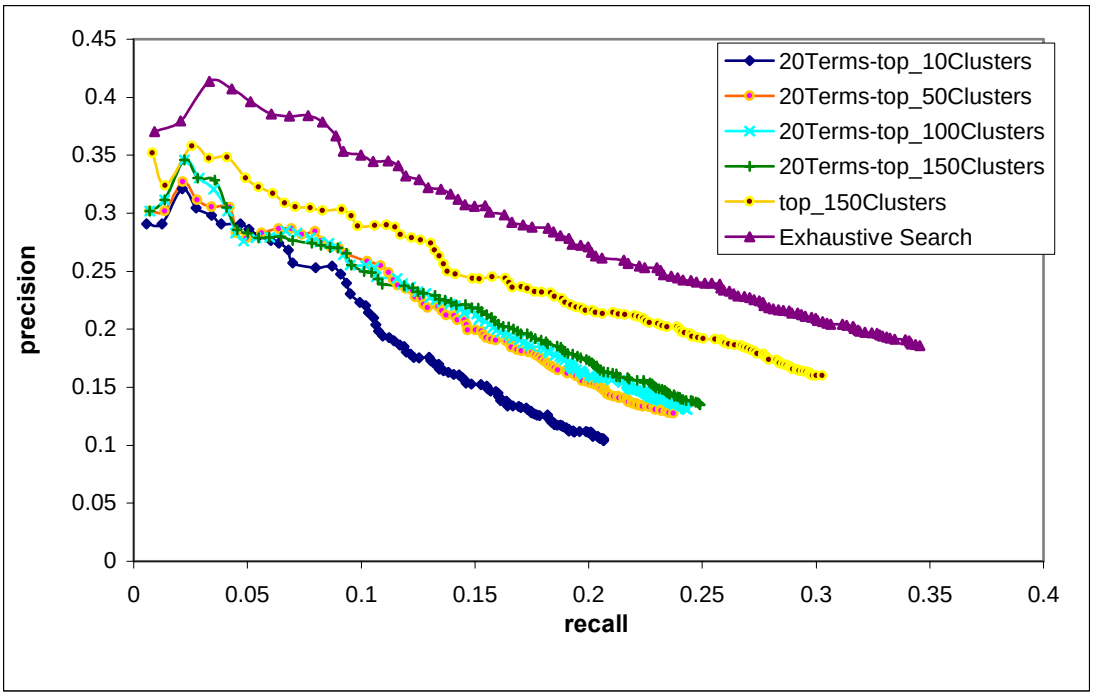

Figure 4.13: Precision-recall diagram of exhaustive search on documents and search based on leaf clusters using the 20 highest weighted centroid terms and the N top ranked clusters for retrievals on OHSUMED

Also, we observe that the efficiency of the retrieval does not noticeably increase with the number of clusters searched. 20Terms-top_150Clusters retrieval strategy performs slightly better than 20Terms-top_100Cluster and 20Terms-top_50Cluster methods. This may occur because for some queries there were not 100 or 150 clusters that contained one or more of the query terms in their centroid vectors. This conclusion can be confirmed by examining the number of documents searched over the 61 queries. In case of 20Terms-top_50Clusters retrieval strategy 33607 documents were searched whereas for 20Terms-top_100Clusters and 20Terms-top_150Clusters the searched documents were 46786 and 54991 correspondingly. We observe that while the number of retrieved clusters were doubled or trebled, the searched documents were not increased significantly.

Experiment 3: Retrieve the Clusters with all Query Terms in Centroids and Search them

The last set of experiments evaluated the performance of cluster-based retrieval using an alternative strategy for selecting the leaf clusters that best match the query. For a specific query we examined only the leaf clusters which contained all the MeSH

query terms in their centroid vectors. Then, the retrieved clusters were ranked according to their similarity to the query. Regarding the number of finally selected ranked clusters we examined three cases. To select all the retrieved clusters, the top 50, the top 30 or the top 15 from the ranked list. We called these experiments AllQinCen_AllClusters, AllQinCen_Top_50Clusters AllQinCen_Top_30Clusters and AllQinCen_Top_15Clusters. For each case a ranked list of documents was produced by computing the cosine similarity between the documents of the selected clusters and the query. To conduct each experiment we selected the 100 top-ranked documents from the list.

In Figure 4.14, we present the precision-recall diagram for the third set of experiments. We compare the results with the curve produced by exhaustive search on OHSUMED documents.

Analyzing the results in Figure 4.14 we observe that retrieval based on leaf clusters of the hierarchy that contained all the query terms in their centroids is almost as effective as the retrieval done by exhaustive search on OHSUMED. Document-based retrieval achieved just up to 2% better precision and up to 2% better recall than AllQinCen_AllClusters retrieval strategy.

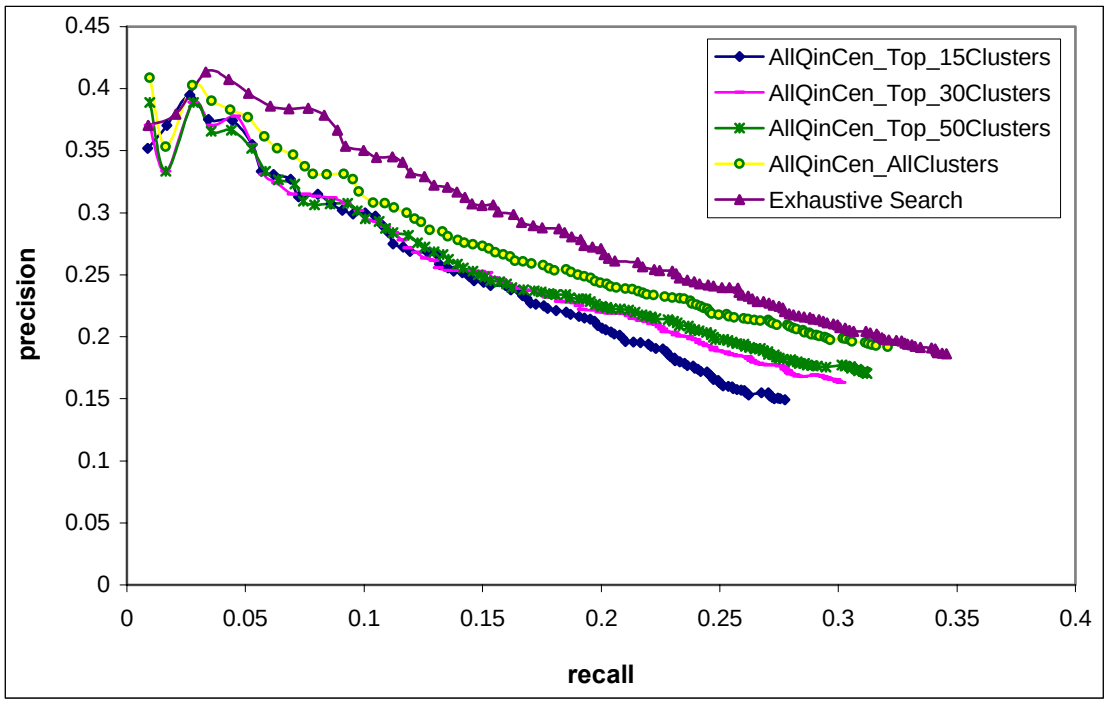

Figure 4.14: Precision-recall curves using the leaf clusters which contains all the query terms in their centroids and a precision/recall curve produced by exhaustive search

The other three examined retrieval methods (AllQinCen_50Clusters, AllQinCen_30Clusters and AllQinCen_15Clusters) performed slightly worse than AllQinCen_AllClusters and document-based retrieval strategy.

Additionally, we evaluated the efficiency of our proposed retrieval strategies in terms of required similarity computations between the documents and the query. In large document collections the computational overhead matching all documents with the query is a major drawback in the retrieval process. Figure 4.15 shows for each retrieval strategy the average number of documents searched over the 61 queries. Regarding the exhaustive search on OHSUMED, 233445 documents were compared to the query to produce the ranked document list. As far as AllQinCen_AllClusters retrieval strategy is concerned, the corresponding average number of documents over all queries was 71649, while for AllQinCen_50Clusters, AllQinCen_30Clusters and AllQinCen_15Clusters retrieval methods were 46262, 34759 and 21606 respectively.

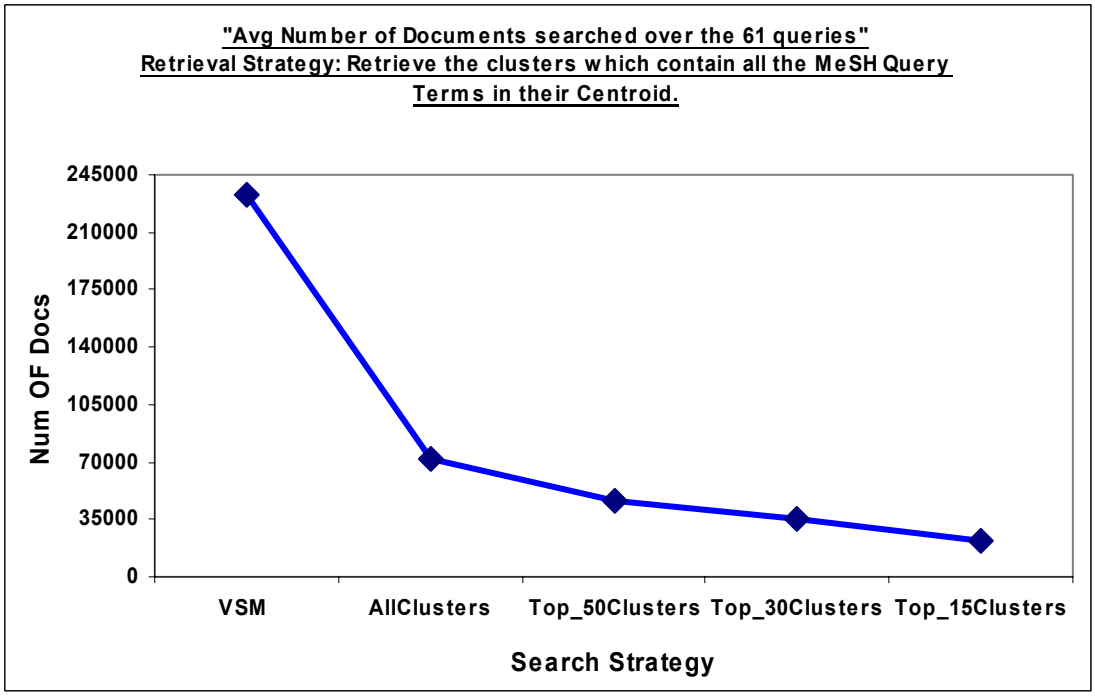

Figure 4.15: The average number of searched documents over the 61 queries for the four retrieval strategies examined in this set of experiments

Summarizing, Figure 4.15 indicate that the three proposed cluster-based retrieval strategies achieved a significant decrease in time and space requirements as compared to retrieval by exhaustive search. Mostly, AllQinCen_AllClusters retrieval method not only saves a huge amount of computation but does so without significant

loss in precision and recall. We observe that among all the cluster-based information retrieval strategies suggested in this section the best results are obtained in case of AllQinCen_AllClusters method. This method is as effective as document-based retrieval on OHSUMED and much more efficient.

Figure 4.16 presents a summary of the best proposed cluster-based retrieval strategy (AllQinCen_AllClusters).

Input: Bottom Level Clusters of the hierarchy (leaf clusters), Query q, Document d (MeSH-based representation), Cosine Similarity Function

Output: Documents ordered by decreasing similarity with the query.

1. MeSH terms extraction from query using the extraction technique presented in section 4.3.
2. Match the query against the leaf clusters which contain all the MeSH query terms in their centroid vector. Use cosine similarity function.
3. Rank clusters by decreasing similarity with the query.
4. Match the query against documents in all retrieved clusters using cosine similarity function.
5. Return a ranked list of documents to the user. (by decreasing similarity to the query).

Figure 4.16: "AllQinCen_AllClusters" cluster-based retrieval method

Chapter 5

Conclusions

We present a short summary of the research conducted in this thesis and provide possible directions for future research.

5.1 Summary

The main objective of this thesis was to develop a highly efficient algorithm for clustering large document collections. We focused on partitional clustering algorithms mainly due to their low time complexity (i.e. linear on the number of documents) as opposed to hierarchical clustering methods which have quadratic time complexity. Therefore partitional techniques are well-suited for clustering large document collections.

Initially, we focused on the standard K-Means clustering approach. We implemented several variants of the original K-Means and we proposed a new variant, the so-called "Incremental K-Means". Incremental K-Means differs from basic K-Means in the way the centroids are updated during each clustering iteration. In K-Means new centroids are computed after each iteration (after all documents have been examined and assigned to clusters). Incremental K-Means updates centroids after a document is assigned to a cluster.

Due to the very large size of document collections and the tremendous explosion of electronic information available on the internet, there is an increased need for effective and efficient clustering algorithms that would aim in reasonable time even on such large document collections and create clusters that correspond to real classes. However, both K-Means and Incremental K-Means produce a flat partition of the data while a construction of a hierarchy of clusters using traditional hierarchical clustering methods is computational prohibitive.

In the following we examined the so-called Bisecting Incremental K-Means which produces a hierarchical clustering solution by recursively applying the Incremental K-Means on a document collection: All documents are initially partitioned into two clusters. Then, the algorithm iteratively selects and bisects each one of the bottom-level clusters until singleton leaf clusters are reached. Bisecting Incremental K-Means can be thought as a divisive hierarchical clustering approach. The so-obtained clusters are structured as a hierarchical binary tree. The run time of the algorithm is $O(n\log(n))$ where n is the number of documents.

The main drawback of the Bisecting Incremental K-Means algorithm was that terminates when each leaf cluster contains a single document. This is because there is no prior knowledge on the desired number of clusters and moreover there is not a criterion for stopping bisections before singleton clusters are reached. In case of large document collections terminating at singleton clusters is time-consuming and the clustering result does not correspond to real classes (mainly at leaf levels and close to the meaningless leaf clusters).

To prevent over-splitting of clusters we proposed a strategy based on the Bayesian Information Criterion (BIC) (introduced earlier in the literature [42]) to stop the divisive procedure. We use BIC to perform a splitting test at each leaf cluster in order to decide whether a cluster should split or not. The BIC score is computed to measure the improvement of a cluster when it is split. If the BIC score of the produced cluster structure is less than BIC score of the parent cluster we do not split the initial cluster. We terminate the divisive procedure when there is no separable leaf cluster according to the BIC function

"BIC-Means", a novel hierarchical clustering algorithm, is the main contribution of this thesis. Building upon Bisecting Incremental K-Means and BIC, BIC-Means combines the advantages of all these ideas. Specifically, BIC-Means has the following characteristics:

1. It is a Bisecting clustering approach which can be used to build a hierarchy of clusters effectively.
2. It incorporates Incremental K-Means as the partitional method for bisecting the selected leaf cluster at each bisecting step. Incremental K-Means efficiently updates cluster's centroids.

3. It uses BIC as the splitting criterion of a cluster and proposes a strategy to stop the divisive procedure based on the Bayesian Information Criterion (BIC).

As a result, BIC-Means produces clusters which are more meaningful as compared to the singleton clusters of Bisecting Incremental K-Means. Overall the proposed algorithm combines the strengths of partitional and hierarchical clustering methods.

We run several sets of experiments. In the first set, we focused on the evaluation of the document clustering algorithms proposed in this thesis. All methods were tested using standard document collections (such as Reuters-21578 and OHSUMED). F-Measure was used to measure the overall "goodness" of the generated clusters. We examined how good the clusters produced by each clustering method match the set of categories (or classes) assigned to the documents (by human experts).

Experimental results on Reuters-21578 [35] indicated that the proposed Incremental K-Means yielded noticeably better F-Measure than the standard K-Means. Additionally, we showed that Incremental K-Means needs only a single iteration of center adjustment to produce a good clustering partition. Then, we examined the performance of Bisecting Incremental K-Means. The results indicated that our Bisecting approach performs significantly better than Incremental K-Means in terms of F-Measure on both data sets. We continued our experiments by examining the performance of Bisecting Incremental K-Means method using vector representation of OHSUMED documents consisting of MeSH terms. We compared these results with those obtained by representation with single word terms. The results indicated that Bisecting Incremental K-Means yields significantly better F-Measure when the OHS0UMED documents are represented by MeSH terms rather than by single word terms.

Then, we evaluated the proposed BIC-Means algorithm in terms of clustering quality and clustering time and compared it with Bisecting Incremental K-Means. Experimental results on both data sets showed that a main advantage of BIC-Means is that requires significantly less time to build a cluster hierarchy than Bisecting Incremental K-Means method (the algorithm does not have to reach at singleton clusters at the leafs). In terms of F-Measure, BIC-Means achieved approximately the same performance with Bisecting Incremental K-Means. Notice though that BIC-

Means was not expected to improve the clustering quality of our initial Bisecting technique (the exhaustive approach). It was introduced as an algorithm that incorporates a criterion for terminating the divisive procedure and for preventing from reaching meaningless leaf clusters. Therefore, BIC-Means is more suited than its competitors for clustering very large document collections effectively. This is due to not only its low computational requirements, but also comparable performance. Notice that, BIC-Means produces meaningful leaf clusters.

The second set of experiments focused on examining the effectiveness and efficiency of a cluster-based information retrieval system. We applied the proposed BIC-Means on OHSUMED (a very large document collection with 233445 medical articles from Medline) in order to create a hierarchy of clusters. We demonstrated that it is possible to apply clustering to reduce the size of the search (and therefore retrieval response times) on large data sets such OHSUMED.

The search strategy relied on searching for the clusters that best match the query. We tested several variants of the above idea. All searched clusters at the leaf level of the hierarchy (intermediate clusters need not be searched as they contain documents which are also combined by the leaf clusters). Each search strategy incorporates different criteria for matching the query against leaf clusters. We evaluated the cluster-based retrieval strategies and compared them against retrieval results by exhaustive search on OHSUMED. In parallel, we examined the retrieval strategies using MeSH terms in document and cluster representation. These are more compact than single word representations and produce better clustering solutions on medical data sets.

The experimental results indicated that among all cluster-based retrieval strategies proposed in this thesis the best results are obtained in case we examined only the leaf clusters which contained all the MeSH terms of the query in their centroid vectors (we searched the documents which were contained in the retrieved clusters). The best proposed cluster-based retrieval strategy searched only 30% of all OHSUMED documents as opposed to the sequential one which matches all documents (one by one) with the query. Experiments also demonstrated that this strategy is almost as effective as the retrieval by exhaustive search on OHSUMED. Summarizing, this cluster-based retrieval method runs faster without significant loss in precision and recall.

5.2 Future Work

We present some open issues for future work in the following sub-sections.

5.2.1 Additional Document Clustering and Retrieval Evaluation

In this work, we experimentally evaluated our proposed document clustering algorithms on two document collections (OHSUMED and Reuters). We plan to extend our evaluation using other general or application specific data sets. Furthermore, we would like to compare our experimental results with results have reported by other hierarchical and partitional clustering algorithms proposed in the literature.

In addition to document clustering experiments we proposed several cluster-based retrieval strategies to improve retrieval by exhaustive search on OHSUMED. It would be interesting to investigate additional cluster-based retrieval strategies. First, "top-down" strategy proposes that the search begins from the root of the tree and moves down the tree following the path of maximum similarity. Second, we would like to examine the "bottom-up" strategy. The search starts from a bottom-level cluster towards the root of the tree.

5.2.2 Medline Clustering and Browsing

In this thesis we applied the proposed BIC-Means algorithm on entire OHSUMED (subset of Medline) to produce a hierarchy of clusters. In the future, we plan to apply BIC-Means on the Medline database. Medline contains more that 15 million references (version 2006) to journal articles in life sciences, medicine and bio-medicine. Due to the huge size of Medline, it would be a challenging task for us to organize this enormous amount of documents into meaningful clusters which contain related documents. Thus, hierarchical clustering of Medline could be used to improve the effectiveness and efficiency of document retrieval. The users will be able to locate quickly and accurately relevant information. Moreover, the produced clustering result will provide effective and intuitive browsing, navigation and summarization of the millions Medline documents.

5.2.3 Clustering Dynamic Document Collections

Modern information systems have vast amount of un-organized data that change dynamically. Consider for example the flow of information that arrives incrementally on news wires message systems (Reuters, Marketwatch, Yahoo, etc) or a document collection that varies over time as new documents arrive continuously, and they need to be inserted in the collection. Clustering centroids are updated continuously and after a while clustering has to be recomputed.

Static clustering algorithms, such BIC-Means, generate a fixed number of clusters. We plan to develop the dynamic version of BIC-Means. It might incrementally compute clusters of similar documents, supporting both insertions and deletions. As a new document is inserted or deleted from the corpus the hierarchy of clusters is re-organized. This process would incorporate either new split on leaf clusters or merges of existing leaf clusters.

Summarizing, the dynamic clustering algorithm will keep dynamic corpora or databases organized. So far, dynamic versions of clustering algorithms have not been examined adequately in the literature. Dynamic clustering can be applied in research areas, such peer to peer systems and sensor networks, as well.

5.2.4 Semantic Similarity Methods in Document Clustering

In this study, the similarity between two documents is computed according to the Vector Space Model (VSM) [50] as the cosine of the inner product between their document vectors. VSM relates documents that use identical terminology. However, plain lexicographic analysis and matching between terms is not generally sufficient to determine if two terms are similar and consequently whether two documents are similar. The lack of common terms in two documents does not necessarily mean that the documents are not related. Two terms can be semantically similar (e.g., can be synonyms or have similar meaning) although they are lexically different terms in the documents. Therefore, computing document similarity by word-based classical information retrieval models (e.g., VSM, Probabilistic, Boolean) is not so effective. For example, VSM will not recognize synonyms or semantically similar terms (e.g., “car”, “automobile”).

In order to take advantage of semantically similar terms, we plan to integrate semantic knowledge into proposed document clustering algorithms. Several methods for determining semantic similarity between terms have been proposed in the

literature, most of them using ontologies such as WordNet[10] and MeSH[11]. The selection of ontology depends on the application domain. In case of natural language terms semantic similarity can be implemented and evaluated using WordNet as the underlying reference ontology. WordNet is a controlled vocabulary and thesaurus offering a taxonomic hierarchy of natural language terms developed at Princeton University. In case of medical terms semantic similarity can be computed using the MeSH ontology which contains medical and biomedical terms. As we used OHSUMED in many document clustering experiments, MeSH ontology will be appropriate for computing semantic similarity between medical terms and consequently between OHSUMED documents.

Regarding our cluster-based information retrieval experiments, documents that contained related information but their context was described by other terms, were not returned to the user. For example, let's say that some documents use the term "ache" instead of "pain". Although the two terms are synonyms, if the user's query contains just the term "ache", documents that use "pain" instead, won't be returned.

For this, it would be interesting to investigate cluster-based retrieval methods capable for discovering semantic similarities between documents and queries. In our retrieval experiments on OHSUMED, retrieval by semantic similarity could be applied by using MeSH as the underlying reference ontology and by associating terms using semantic similarity methods [33], [36], [37], [44], [45], [49], [56].

[10] http://wordnet.princeton.edu
[11] http://www.nlm.nih.gov/mesh

References

[1] Charu C. Aggarwal, Stephen C. Gates, and Philip S. Yu. *On the merits of building categorization systems by supervised clustering.* In Proc. of the Fifth ACM SIGKDD Int'l Conference on Knowledge Discovery and Data Mining, pages

[2] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley Longman, 1999.

[3] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler (2000). *Extensible Markup Language (XML) 1.0 (Second Edition)* W3C Recommendation, October 2000. Available at: http://www.w3.org/TR/2000/REC-xml-20001006/

[4] D. Chickering, D. Heckerman, and C. Meek. *A Bayesian Approach to Learning Bayesian Networks with Local Structure*. In Proceedings of Thirteenth Conference on Uncertainty in Artificial Intelligence, pages 80–89. Morgan Kaufmann, 1997.

[5] W. B. Croft. *A model of cluster searching based on classification. Information Systems*, Vol. 5, pp. 189-195, 1980.

[6] Douglass R. Cutting, David R. Karger, Jan O. Pedersen, and John W. Tukey, *Scatter/Gather: A Cluster-based Approach to Browsing Large Document Collections*. SIGIR '92, pages 318 – 329, 1992

[7] E. Diday and J. C. Simon. *Clustering analysis. In Digital Pattern Recognition,* K. S. Fu, Ed. Springer-Verlag, Secaucus, NJ, pages 47–94, 1976

[8] C. Ding, X. He. *Cluster merging and splitting in hierarchical clustering algorithms.* In IEEE International Conference on Data Mining (ICDM'02), Japan, 2002.

[9] C. Ding, X. He, H. Zha, M. Gu, and H. Simon. *A min-max cut algorithm for graph partitioning and data clustering.* Proc. IEEE Int'l Conf. Data Mining, pages 107-114, 2001.

[10] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification*. John Wiley & Sons, 2001.

[11] W. Douglas Johnston Stuart J. Nelson and Betsy L. Humphreys. Relationships in Medical Subject Headings (MeSH). In *National Library of Medicine, Bethesda, MD, USA*, 2002.

[12] B. S. Duran and P. L. Odell. *Cluster Analysis: A Survey*. Springer-Verlag, New-York, NY, 1974

[13] V. Faber. *Clustering and the Continuous k-Means Algorithm*, Los Alamos Science, November 22, 1994.

[14] Benjamin C. M. Fung, Ke Wang, Martin Ester. *Hierarchical Document Clustering Using Frequent Itemsets.* Proceedings of the 2003 SIAM International Conference on Data Mining.

[15] N. Fuhr. Probabilistic M*odels in Information Retrieval*. The Computer Journal, vol. 35, no. 3, pages 243--255, 1992.

[16] S. Guha, R. Rastogi, and K. Shim. *CURE: An efficient clustering algorithm for large databases.* In Proceedings of the Int'l Conference on Management of Data (SIGMOD'98) (Seattle, WA). ACM Press, June 1998.

[17] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. *ROCK: a robust clustering algorithm for categorical attributes.* In Proc. of the 15th Int'l Conf. on Data Eng., 1999.

[18] A. El-Hamdouchi and P. Willett. *Comparison of Hierarchic Agglomerative Clustering Methods for Document Retrieval*. The Computer Journal 32(3): 220-227, 1989.

[19] Jiawei Han and Michelle Kamber. *Data Mining: Concepts and Techniques.* Morgan Kaufmann, 2001.

[20] J.A. Hartigan and M.A. Wong. *A K-Means clustering algorithm*. In applied statistics, pages 208-220

[21] W. Hersh, C. Buckley, T.J. Leone, and D. Hickam. *OHSUMED: An interactive retrieval evaluation and new large test collection for research.* In SIGIR-94, pages 192–201, 1994.

[22] A. Hliautakis. *Semantic Similarity Measures in MeSH Ontology and their application to Information Retrieval on Medline.* Technical Report TR-TUC-ISL-02-2005, September 2005. Available on the WWW at URL http://www.intelligence.tuc.gr/publications/Hliautakis.pdf

[23] J. E. Jackson. *A User's Guide To Principal Components*. John Wiley & Sons, 1991.

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

[25] A.K. Jain., M.N. Murty, and P.J Flynn, *Data Clustering: A Review*. ACM Computing Surveys, 31(3), pp.264-323, 1999.

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

[27] W. Douglas Johnston, Stuart J. Nelson and Betsy L. Humphreys. *Relationships in Medical Subject Headings (MeSH).* In National Library of Medicine, Bethesda, MD, USA, 2002.

[28] G. Karypis, E.H. Han, and V. Kumar. *Chameleon: A hierarchical clustering algorithm using dynamic modeling. IEEE Computer*, 32(8):68–75, 1999.

[29] R. Kass and L. Wasserman. *A reference Bayesian test for nested hypotheses and its relationship to the Schwarz criterion.* Journal of the American Statistical Association, 90, 773-795, 1995.

[30] L. Kaufman and P.J. Rousseeuw. *Finding Groups in Data: an Introduction to Cluster Analysis*. John Wiley & Sons, 1990.

[31] B. King. *Step-wise clustering procedures*. Journal of the American Statistical Association, 69:86–101, 1967.

[32] Bjornar Larsen and Chinatsu Aone. *Fast and effective text mining using linear-time document clustering.* In *Proc. of the Fifth ACM SIGKDD Int'l Conference on Knowledge Discovery and Data Mining*, pages 16–22, 1999.

[33] C. Leacock and M. Chodorow. *Combining Local Context and WordNet Similarity for Word Sense Identification in WordNet.* In Fellbaum, C., ed.: An Electronic Lexical Database. MIT Press (1998) 265–283

[34] R.C.T. Lee. Clustering *analysis and its applications.* In J.T. Toum, editor, *Advances in Information Systems Science*. Plenum Press, New York, 1981.

[35] D. D. Lewis. *Reuters-21578 text categorization test collection distribution 1.0.* http://www.research.att.com/lewis, 1999.

[36] Y, Li, Z.A. Bandar, and D. McLean. *An Approach for Measuring Semantic Similarity between Words Using Multiple Information Sources*. IEEE Trans. on Knowledge and Data Engineering 15(4) (2003) 871–882

[37] D. Lin. *Principle-Based Parsing Without Overgeneration*. In: Annual Meeting of the Association for Computational Linguistics (ACL'93), Columbus, Ohio (1993) 112–120

[38] R. Michalski, R. E. Stepp, and E. Diday. *A recent advance in data analysis: Clustering objects into classes characterized by conjunctive concepts.* In Progress in Pattern Recognition, Vol. 1, L. Kanal and A. Rosenfeld, Eds. North-Holland Publishing Co., Amsterdam, The Netherlands, 1981

[39] G. Nagy. *State of the art in pattern recognition*. Proc. IEEE 56, 836-862, 1968.

[40] R. Ng and J. Han. *Efficient and effective clustering method for spatial data mining.* In Proc. of the 20th VLDB Conference, pages 144–155, Santiago, Chile, 1994.

[41] T. Nomoto and Y. Matsumoto. *A New Approach to Unsupervised Text Summarization.* Proceedings of the 24th ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'01), pages 26–34, 2001.

[42] D. Pelleg and A. Moore. *X-means: Extending K-means with Efficient Estimation of the Number of Clusters.* In Seventeenth International Conference on Machine Learning, 2000.

[43] Qi Quifen, Gao Qigang and Michael Shepherd. *Accessing Tacit Knowledge in the Pediatric Pain e-Mail Archives.* Proceedings of the 38th Hawaii International Conference on System Sciences – 2005.

[44] R. Rada, H. Mili, E. Bicknell, and M. Blettner. *Development and Application of a Metric on Semantic Nets.* IEEE Trans.on Systems, Man, and Cybernetics 19(1) (1989) 17–30

[45] O. Resnik. *Semantic Similarity in a Taxonomy: An Information-Based Measure and its Application to Problems of Ambiguity and Natural Language*. Journal of Artificial Intelligence Research 11 (1999) 95–130

[46] Van Rijsbergen, C.J. & Croft, W. B. *Document clustering: An evaluation of some experiments with the Cranfield 1400 collection.* Information Processing & Management, 11, pp. 171-182, 1975.

[47] Van Rijsbergen, C.J. *Information Retrieval* (2nd ed.). London: Butterworths, 1979.

[48] S. Robertson. *The Probability Ranking Principle in IR.* Journal of Documentation 33, pages 294-304, 1977.

[49] M. Rodriguez and M. Egenhofer. *Determining Semantic Similarity Among Entity Classes from Different Ontologies.* IEEE Trans. on Knowledge and Data Engineering 15(2) (2003) 442–456

[50] G. Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Addison-Wesley, 1989.

[51] G. Salton. *Introduction to Modern Information Retrieval.* McGraw-Hill, 1983.

[52] G. Schwarz, Estimating *the dimension of a model.* Ann. Statistics 6, 461-464, 1978.

[53] R. Sibson. *SLINK: An optimally efficient algorithm for the single link cluster method.* Computer Journal, 16:30-34, 1973.

[54] P. H. Sneath and R. R. Sokal. *Numerical Taxonomy.* Freeman, London, UK, 1973.

[55] M.Steinbach, G. Karypis, and V. Kumar. *A comparison of document clustering techniques.* In KDD Workshop on Text Mining, 2000.

[56] A. Tversky. *Features of Similarity*. Psychological Review 84(4) (1977) 327–352

[57] G. Varelas. *Semantic Similarity Methods in WordNet and Their Application to Information Retrieval on the Web.* Technical Report TR-TUC-ISL-01-2005. Department of Electronic and Computer Engineering, Chania, Crete, Greece, 2005. Available on the WWW at URL http://www.intelligence.tuc.gr/publications/Vatelas.pdf

[58] E. M. Voorhees, E.M. *The cluster hypothesis revisited.* In SIGIR 1985, pp.188-196, 1985.

[59] Wikipedia: http://en.wikipedia.org/wiki/MeSH

[60] Y.Yang. *An evaluation of statistical approaches to text categorization.* Journal of Information Retrieval, 1(1/2):67-88, 1999.

[61] Y.Yang and J. Pedersen. *A comparative study on feature selection in text categorization.* In J. D. H. Fisher, editor, The Fourteenth International Conference on Machine Learning (ICML'97), pages 412-420. Morgan Kaufmann, 1997.

[62] Oren Zamir, Oren Etzioni, Omid Madani and Richard M. Karp, *Fast and Intuitive Clustering of Web Documents*, KDD '97, Pages 287-290, 1997.

[63] Ying Zhao and George Karypis. *Criterion functions for document clustering: Experiments and analysis.* Technical Report TR#01–40, Department of Computer Science, University of Minnesota, Minneapolis, MN, 2001. Available on the WWW at *http://cs.umn.edu/˜karypis/publications*.

[64] Y. Zhao and G. Karypis. *Soft Clustering Criterion Functions for Partitional Document Clustering*. Technical Report TR-04-022, Department of Computer Science, University of Minnesota, Minneapolis, 2004. Available on the WWW at URL *http://www.cs.umn.edu/˜karypis*.

[65] Ying Zhao, George Karypis, Jack G. Conrad and Khalid Al-Kofahi. *Effective Document Clustering for Large Heterogeneous Law Firm Collections.* In Proceedings of ICAIL, 2005.

[66] Y. Zhao and G. Karypis. *Evaluation of hierarchical clustering algorithms for document datasets.* In Proc. of Int'l. Conf. on Information and Knowledge Management, pages 515–524, 2002.

[67] K. Wang, S. Zhou, Y. He, "*Hierarchical classification of real life documents"*, SIAM International Conference of Data Mining (SDM'01, Chicago, United States, 2001.

[68] P. Willett, "*Recent Trends in Hierarchic Document Clustering: A Critical Review"*, Information Processing & Management, 24(5), pp. 577-597, 1988.

Appendix A

Appendix

A.1 MeSH DTD File

```
<!-- MESH DTD files for descriptors desc2006.dtd -->

<!ENTITY  % DescriptorReference "(DescriptorUI, DescriptorName)">
<!ENTITY  % normal.date "(Year, Month, Day)">
<!ENTITY  % ConceptReference" (ConceptUI, ConceptName,
                                                    ConceptUMLSUI?)">
<!ENTITY  % QualifierReference "(QualifierUI, QualifierName)">
<!ENTITY  % TermReference "(TermUI, String)">

<!ELEMENT DescriptorRecordSet (DescriptorRecord*)>
<!ELEMENT DescriptorRecord (%DescriptorReference;,
                    DateCreated,
                    DateRevised?,
                    DateEstablished?,
                    ActiveMeSHYearList,
                    AllowableQualifiersList?,
                    Annotation?,
                    HistoryNote?,
                    OnlineNote?,
                    PublicMeSHNote?,
                    PreviousIndexingList?,
                    EntryCombinationList?,
                    SeeRelatedList?,
                    ConsiderAlso?,
                    RunningHead?,
```

```
                    TreeNumberList?,
                    RecordOriginatorsList,
                    ConceptList) >
<!ATTLIST DescriptorRecord DescriptorClass (1 | 2 | 3 | 4)  "1">

<!ELEMENT ActiveMeSHYearList (Year+)>
<!ELEMENT AllowableQualifiersList (AllowableQualifier+) >
<!ELEMENT AllowableQualifier (QualifierReferredTo,Abbreviation )>
<!ELEMENT Annotation (#PCDATA)>
<!ELEMENT ConsiderAlso (#PCDATA) >
<!ELEMENT Day (#PCDATA)>
<!ELEMENT DescriptorUI (#PCDATA) >
<!ELEMENT DescriptorName (String) >
<!ELEMENT DateCreated (%normal.date;) >
<!ELEMENT DateRevised (%normal.date;) >
<!ELEMENT DateEstablished (%normal.date;) >
<!ELEMENT DescriptorReferredTo (%DescriptorReference;) >

<!ELEMENT EntryCombinationList (EntryCombination+) >
<!ELEMENT EntryCombination     (ECIN,
                        ECOUT)>
<!ELEMENT ECIN (DescriptorReferredTo,QualifierReferredTo) >
<!ELEMENT ECOUT (DescriptorReferredTo,QualifierReferredTo? ) >
<!ELEMENT HistoryNote (#PCDATA)>
<!ELEMENT Month (#PCDATA)>
<!ELEMENT OnlineNote (#PCDATA)>
<!ELEMENT PublicMeSHNote (#PCDATA)>
<!ELEMENT PreviousIndexingList (PreviousIndexing)+>
<!ELEMENT PreviousIndexing (#PCDATA) >
<!ELEMENT RecordOriginatorsList (RecordOriginator,
                        RecordMaintainer?,
                        RecordAuthorizer? )>
<!ELEMENT RecordOriginator (#PCDATA)>
<!ELEMENT RecordMaintainer (#PCDATA)>
```

```
<!ELEMENT RecordAuthorizer (#PCDATA)>
<!ELEMENT RunningHead (#PCDATA)>
<!ELEMENT QualifierReferredTo (%QualifierReference;) >
<!ELEMENT QualifierUI (#PCDATA) >
<!ELEMENT QualifierName (String) >
<!ELEMENT Year (#PCDATA) >
<!ELEMENT SeeRelatedList (SeeRelatedDescriptor+)>
<!ELEMENT SeeRelatedDescriptor (DescriptorReferredTo)>
<!ELEMENT TreeNumberList (TreeNumber)+>
<!ELEMENT TreeNumber (#PCDATA)>
<!ELEMENT ConceptList (Concept+)  >
<!ELEMENT Concept (%ConceptReference;,
            CASN1Name?,
            RegistryNumber?,
            ScopeNote?,
            SemanticTypeList?,
            PharmacologicalActionList?,
            RelatedRegistryNumberList?,
            ConceptRelationList?,
            TermList)>
<!ATTLIST Concept PreferredConceptYN (Y | N) #REQUIRED >

<!ELEMENT ConceptUI (#PCDATA)>
<!ELEMENT ConceptName (String)>
<!ELEMENT ConceptRelationList (ConceptRelation+) >
<!ELEMENT ConceptRelation (Concept1UI,
                Concept2UI,
                RelationAttribute?)>
<!ATTLIST ConceptRelation RelationName (NRW | BRD | REL) #IMPLIED >
<!ELEMENT Concept1UI (#PCDATA)>
<!ELEMENT Concept2UI (#PCDATA)>
<!ELEMENT ConceptUMLSUI (#PCDATA)>
<!ELEMENT CASN1Name (#PCDATA)>
<!ELEMENT PharmacologicalActionList (PharmacologicalAction+)>
```

```
<!ELEMENT PharmacologicalAction (DescriptorReferredTo) >
<!ELEMENT RegistryNumber (#PCDATA)>
<!ELEMENT RelatedRegistryNumberList (RelatedRegistryNumber+)>
<!ELEMENT RelatedRegistryNumber (#PCDATA)>
<!ELEMENT RelationAttribute (#PCDATA)>
<!ELEMENT ScopeNote (#PCDATA)>
<!ELEMENT SemanticTypeList (SemanticType+)>
<!ELEMENT SemanticType (SemanticTypeUI, SemanticTypeName) >
<!ELEMENT SemanticTypeUI (#PCDATA)>
<!ELEMENT SemanticTypeName (#PCDATA)>
<!ELEMENT TermList (Term+)>
<!ELEMENT Term (%TermReference;,
            DateCreated?,
            Abbreviation?,
            SortVersion?,
            EntryVersion?,
            ThesaurusIDlist?)>
<!ATTLIST Term    ConceptPreferredTermYN (Y | N) #REQUIRED
             IsPermutedTermYN (Y | N) #REQUIRED
             LexicalTag              (ABB|ABX|ACR|ACX|EPO|LAB|NAM|NON|TRD)
#REQUIRED
             PrintFlagYN (Y | N) #REQUIRED
             RecordPreferredTermYN (Y | N)  #REQUIRED>
<!ELEMENT TermUI (#PCDATA)>
<!ELEMENT String (#PCDATA)>
<!ELEMENT Abbreviation (#PCDATA)>
<!ELEMENT SortVersion (#PCDATA)>
<!ELEMENT EntryVersion (#PCDATA)>
<!ELEMENT ThesaurusIDlist (ThesaurusID+)>
<!ELEMENT ThesaurusID (#PCDATA)>
```

A.2 Document Collections

A.2.1 Reuters-21578

Figure A.1 presents a Reuters-21578 document to demonstrate its structure and its attributes. The Reuters's document tags are bolded.

<REUTERS TOPICS="YES" LEWISSPLIT="TEST" CGISPLIT="TRAINING-SET" OLDID="7485" NEWID="18040">
<DATE> 2-JUN-1987 10:54:40.06**</DATE>**
<TOPICS><D>acq**</D></TOPICS>**
<PLACES><D>usa**</D></PLACES>**
<PEOPLE></PEOPLE>
<ORGS></ORGS>
<EXCHANGES></EXCHANGES>
<COMPANIES></COMPANIES>
<UNKNOWN>
F
f1270 reute
d f BC-ORION-BROADCAST-<OBGI 06-02 0079**</UNKNOWN>**
<TEXT>
<TITLE>ORION BROADCAST <OBGI.O> BUYS FORD <F> UNIT**</TITLE>**
<DATELINE> DENVER, June 2 - **</DATELINE><BODY>**Orion Broadcast Group Inc said its majority-owned Orion Financial Services Corp subsidiary has agreed to purchase FN Realty Services Inc from Ford Motor Co for 1,200,000 to 1,500,000 dlrs in cash and notes.

It said closing is expected within 45 days after receipt of regulatory approvals.

FN provides loan collection, accounting, data processing and administrative services to the real estate industry.

Reuter
</BODY></TEXT>
</REUTERS>

Figure A.1: A Reuters-21578 document

A.2.2 OHSUMED

Figure A.2 illustrates an OHSUMED document and Figure A.3 presents the field (attribute) definitions.

.I 6

.U

87049093

.S

Am J Emerg Med 8703; 4(6):514-5

.M

Abdominal Injuries/ET; Accidents, Occupational; Accidents, Traffic/*; Adult; Amputation; Blood Transfusion/*; Case Report; Female; Fractures/ET; Human; Pelvic Bones/IN; Shock, Hemorrhagic/ET/*TH; Wounds, Nonpenetrating/*CO.

.T

Massive transfusion without major complications after trauma.

.P

JOURNAL ARTICLE.

.W

A case of massive degloving injury of the trunk, with open pelvic fracture, and evisceration of abdominal contents from blunt trauma is presented. The most significant aspect of this case was the transfusion of 173 units of packed cells and 176 units of fresh frozen plasma in the first thirty hours. The patient ultimately recovered and returned to work.

.A

Brotman S; Lamonica C; Cowley RA.

Figure A.2: An OHSUMED document

.I	sequential identifier (important note: documents should be processed in this order)
.U	MEDLINE identifier (UI) (<DOCNO> used for relevance judgements)
.M	Human-assigned MeSH terms (MH)
.T	Title (TI)
.P	Publication type (PT)
.W	Abstract (AB)

.A Author (AU)

.S Source (SO)

Figure A.3: Field Definitions

A.3 Lucene

Lucene is a full-featured (Java-based) open source toolkit for text indexing and searching. It is easy to use, flexible, and powerful - a model of good object-oriented software architecture. Powerful abstractions and useful concrete implementations make Lucene very flexible, and allow new users to get up and running quickly and painlessly. We use Lucene in order to perform various operations needed by the document clustering and retrieval experiments we conduct as part of this work (indexing, searching etc). Lucene is freely available at *http://lucene.apache.org.*

A.4 Retrieval on OHSUMED – Evaluation Queries

A.4.1 61 Original OHSUMED Queries

For the retrieval evaluation we used a subset of 61 queries of the original OHSUMED query set. We present them below.

.I 2
.B
60 yo male with disseminated intravascular coagulation
.W
pathophysiology and treatment of disseminated intravascular coagulation

.I 3
.B
prolonged prothrombin time
.W
anticardiolipin and lupus anticoagulants, pathophysiology, epidemiology, complications

.I 5
.B
58 yo with cancer and hypercalcemia
.W
effectiveness of etidronate in treating hypercalcemia of malignancy

.I 6
.B
55 yo female,postmenopausal
.W
does estrogen replacement therapy cause breast cancer

.I 9
.B
30 year old with fever, lymphadenopathy, neurologic changes and rash
.W
t-cell lyphoma associated with autoimmune symptoms

.I 10
.B
57yo male with hypercalcemia secondary to carcinoma
.W
effectiveness of gallium therapy for hypercalcemia

.I 12
.B
30 y old female suvivor of satanic cult
.W
descriptions of injuries associated with cult activities

.I 14
.B
35 y o male with aids and pancytopenia
.W
pancytopenia in aids, workup and etiolog

.I 16
.B
chronic fatigue syndrome
.W
chronic fatigue syndrome, managment and treatment

.I 17
.B
29 yo female 3 months pregnant
.W
Rh isoimmunization, review topics

.I 18
.B
endocarditis
.W
endocarditis, duration of antimicrobial therapy

.I 19

.B
18 yo pregnant woman with hyperthroidism
.W
use of beta-blockers for thyrotoxicosis during pregnancy

.I 20
.B
cerebral palsy with depression
.W
relationship of cerebral palsy and depression

.I 22
.B
35 yo with advanced metastatic breast cancer
.W
chemotherapy advanced for advanced metastatic breast cancer

.I 25
.B
49 yo B male with hypotension, hypokalemia, and low aldosterone.
.W
isolated hypoaldosteronism, syndromes where hypoaldosteronism and hypokalemia occur concurrently

.I 29
.B
24 y o female g1 p0 9 months pregnant with thrombocytopenia
.W
thrombocytopenia in pregnancy, etiology and management

.I 30
.B
63 y o male with acute renal failure probably 2nd to aminoglycosides/contrast dye
.W
acute tubular necrosis due to aminoglycosides, contrast dye, outcome and treatment

.I 31
.B
45 yo wf , chronic lower extremity pain
.W
chronic pain management, review article, use of tricyclic antidepressants

.I 32
.B
40 y o male with cocaine withdrawal
.W
cocaine withdrawal management

.I 33
.B

67 yo wm with hemiballismus
.W
carotid endarterectomy, when to perform

.I 35
.B
42 YO WITH HEPATOCELLULAR CARCINOMA
.W
RISK FACTORS and TREATMENT for HEPATOCELLULAR CARCINOMA

.I 37
.B
FATIGUE
.W
FIBROMYALGIA/FIBROSITIS, DIAGNOSIS AND TREATMENT

.I 38
.B
DIABETIC GASTROPARESIS
.W
DIABETIC GASTROPARESIS, TREATMENT

.I 39
.B
35 Y O WITH GASTROENTERITIS
.W
VIRAL GASTROENTERITIS, CURRENT MANAGEMENT

.I 41
.B
46 Y0 NEW ASCITES
.W
ASCITES, DIFFERENTIAL diagnosis and work-up

.I 42
.B
31yo female with downs syndrome
.W
keratoconus, treatment options

.I 43
.B
55 yo male with back pain
.W
back pain, information on diagnosis and treatment

.I 46
.B
64 yo black male
.W

occult blood sceening, need for routine screening

.I 48
.B
35 year old with peripheral neuropathy and edema
.W
which peripheral neuropathies have associated edema

.I 50
.B
62 year old with stroke and systolic hypertension
.W
isolated systolic hypertension, shep study

.I 52
.B
74 yo man with post-radiation pericardial effusion and near tamponade
.W
indications for and success of pericardial windows and pericardectomies

.I 54
.B
older male
.W
angiotensin converting enzyme inhibitors, review article

.I 55
.B
24 y. o. w. f. s/p DVT currently on coumadin
.W
course of anticoagulation with coumadin

.I 57
.B
22 yo with fever, leukocytosis, increased intracranial pressure, and central herniation
.W
cerebral edema secondary to infection, diagnosis and treatment

.I 58
.B
65 yo female with a breast mass
.W
diagnostic and therapeutic work up of breast mass

.I 60
.B
28 yr old male with endocarditis
.W
treatment of endocarditis with oral antibiotics

.I 62
.B
26 y o female with bulimia
.W
evaluation for complications and management of bulimia

.I 63
.B
migraine
.W
treatment of migraine headaches with beta blockers and calcium channel blockers

.I 64
.B
30 y o with hypothermia
.W
prevention, risk factors, pathophysiology of hypothermia

.I 66
.B
35 female with pickwickian syndrome
.W
complications of prolonged progesterone

.I 69
.B
70 y o female with left lower quadrant pain
.W
diverticulitis, differential diagnosis and management

.I 71
.B
27 yo with cystic fibrosis and renal failure
.W
cystic fibrosis and renal failure, effect of long term repeated use of aminoglycosides

.I 73
.B
23 YO male with alcolol abuse here for TIPS procedure.
.W
portal hypertension and varices, management with TIPS procedure

.I 74
.B
43 y o female with fevers, increased CPK
.W
neuroleptic malignant syndrome, differential diagnosis, treatment

.I 75
.B

carcinoid tumors of the liver and pancreas
.W
carcinoid tumors of the liver and pancreas, research, treatments

.I 77
.B
30 y o with dehydration, hyperthermia
.W
heat exhaustion, management and pathophysiology

.I 79
.B
25 y o female with anorexia/bulimia
.W
complications and management of anorexia and bulimia

.I 81
.B
48 y o with culture negative endocarditis suspected
.W
culture negative endocarditis, organisms, diagnosis, treatment

.I 82
.B
24 y o with HIV
.W
aids dementia, workup

.I 83
.B
patient s/p renal transplant with fever
.W
infections in renal transplant patients

.I 84
.B
50 year old with copd
.W
theophylline uses--chronic and acute asthma

.I 88
.B
lung cancer
.W
lung cancer, radiation therapy

.I 89
.B
60 year old with lung abscess
.W

surgery vs. percutaneous drainage for lung abscess

.I 90
.B
30 year old female with paroxysmal anaphylaxis
.W
Catamenorrheal Anaphylaxis

.I 92
.B
66 year old male with Guillain-Barre syndrome
.W
Guillain-Barre syndrome, Sensitivity and specificity of nerve conduction velocity tests

.I 94
.B
23 YO W DYSURIA
.W
Urinary Tract Infection, CRITERIA FOR TREATMENT AND ADMISSION

.I 96
.B
41 yo w f here for new visit healthy otherwise
.W
preventive health care for the adult patient

.I 100
.B
32 yo schizophrenic patient with peripheral neuropathy
.W
association of neuroleptics and peripheral neuropathy

.I 103
.B
50 yo woman with breakthrough vaginal bleeding while on estrogen and progesterone therapy
.W
differential diagnosis of breakthrough vaginal bleeding while on estrogen and progesterone therapy

.I 105
.B
68 yo woman with anemia of chronic illness
.W
review of anemia of chronic illness

.I 106
.B
42 yo w/HIV and diarrhea

.W
HIV and the GI tract, recent reviews

A.4.2 MeSH-Based Representations of the 61 OHSUMED Queries

We present the corresponding MeSH-based representations of the 61 queries presented in subsection A.4.1. The number at the left is the identifier of each query.

2. disseminated_intravascular_coagulation male therapeutics
3. lupus prothrombin_time anticoagulants epidemiology
5. hypercalcemia neoplasms etidronic_acid
6. breast_neoplasms female estrogen_replacement_therapy
9. exanthema fever t-lymphocytes lymphatic_diseases
10. gallium hypercalcemia carcinoma male
12. wounds_and_injuries female
14. acquired_immunodeficiency_syndrome pancytopenia male
16. fatigue_syndrome,_chronic therapeutics
17. rh_isoimmunization female
18. endocarditis
19. pregnancy thyrotoxicosis pregnant_women
20. cerebral_palsy depression
22. breast_neoplasms drug_therapy
25. syndrome hypotension hypoaldosteronism aldosterone male hypokalemia
29. pregnancy female thrombocytopenia
30. necrosis male aminoglycosides kidney_failure kidney_failure,_acute therapeutics
31. pain antidepressive_agents,_tricyclic lower_extremity
32. cocaine male
33. dyskinesias endarterectomy,_carotid
35. carcinoma,_hepatocellular risk_factors therapeutics
37. fatigue diagnosis therapeutics
38. gastroparesis therapeutics
39. gastroenteritis
41. diagnosis,_differential ascites

43. back_pain male diagnosis therapeutics world_health_organization prognosis therapeutics
46. mass_screening male occult_blood
48. edema peripheral_nervous_system_diseases
50. hypertension cerebrovascular_accident
52. pericardiectomy pericardial_effusion
54. peptidyl-dipeptidase_a enzyme_inhibitors male
55. warfarin
57. intracranial_pressure brain_edema leukocytosis fever diagnosis infection therapeutics
58. work breast female therapeutics
60. male endocarditis anti-bacterial_agents therapeutics
62. bulimia female evaluation_studies
63. calcium_channels calcium_channel_blockers migraine_disorders therapeutics
64. hypothermia risk_factors
66. obesity_hypoventilation_syndrome progesterone female
68. blood lupus vasculitis rectum
69. pain diagnosis,_differential female diverticulitis
71. cystic_fibrosis aminoglycosides kidney_failure
73. hypertension,_portal methods male varicose_veins
75. pancreas liver research carcinoid_tumor therapeutics
77. fever heat_exhaustion dehydration
79. bulimia female anorexia
81. culture diagnosis endocarditis therapeutics
82. hiv dementia acquired_immunodeficiency_syndrome
83. patients fever infection transplants
84. pulmonary_disease,_chronic_obstructive asthma theophylline
88. lung_neoplasms radiation
89. surgery lung_abscess drainage
90. female anaphylaxis
92. neural_conduction sensitivity_and_specificity guillain-barre_syndrome male
94. urinary_tract urinary_tract_infections therapeutics
100. antipsychotic_agents patients peripheral_nervous_system_diseases association
103. estrogens progesterone diagnosis,_differential hemorrhage women

105. chronic_disease anemia women
106. hiv gastrointestinal_tract diarrhea