Hierarchical Clustering in Medical Document Collections: the BIC-Means Method

Nikos Hourdakis[1], Michalis Argyriou[1], Euripides G.M Petrakis[1], Evangelos E. Milios[2]
[1]Department of Electronic and Computer Engineering
Technical University of Crete (TUC)
Chania, Greece
{hourakis,micharg,petrakis}@intelligence.tuc.gr

[2]Faculty of Computer Science
Dalhousie University
Halifax, Canada
e-mail: eem@cs.dal.ca

Journal of Digital
Information Management

ABSTRACT: *Hierarchical clustering of text collections is a key problem in document management and retrieval. In partitional hierarchical clustering, which is more efficient than its agglomerative counterpart, the entire collection is split into clusters and the individual clusters are further split until a heuristically-motivated termination criterion is met. In this paper, we define the BIC-means algorithm, which applies the Bayesian Information Criterion (BIC) as a domain independent termination criterion for partitional hierarchical clustering. We evaluate the effectiveness of BIC-means in clustering and retrieval on medical document collections and we propose a dynamic version of the BIC-Means algorithm for adapting an existing clustering solution to document additions.*

Categories and Subject Descriptors
H.3.3 [Information Search and Retrieval]; Retrieval models Bayesian: **I.1.2 [Algorithms]**

General Terms: Information clustering, Hierarchical clustering, Text mining, Medical information processing

Keywords: Document Clustering, Termination Criterion, Bayesian Information Criterion (BIC), Dynamic Clustering, Document Retrieval, Performance

Received: May 21 2009; Revised August 11 2009; Accepted 12 October 2009

1. Introduction

Document clustering has been widely applied in various scientific fields for supporting search engines, text mining, and Information Retrieval [1]. 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. Hierarchical clustering, in particular, is commonly used to generate topic hierarchies. It is also a way to circumvent the requirement of flat clustering algorithms to provide the number of clusters as input.

Based on the underlying algorithmic methodology, hierarchical clustering algorithms can be categorized into agglomerative (bottom-up) and divisive (top-down) [2,3]. Divisive approaches start with all documents in the same root cluster and work by iteratively splitting each cluster into a number of smaller ones until a termination criterion is met for each cluster. Partitional clustering algorithms generate a flat set of clusters. *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 advance. Partitional clustering algorithms are well-suited for clustering large data sets due to their low computational requirements (linear in the number of documents) [2]. Furthermore, partitional techniques have been found to lead to better clustering solutions than agglomerative algorithms [2,4].

Partitional clustering methods can also be used to obtain a hierarchical clustering solution via repeated application of the flat partitional algorithm. Bisecting *K*-Means is such an approach [2]. 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 each produced cluster until singleton clusters are obtained at the leafs or until *K* clusters have been produced. The outcome is structured as a binary tree.

The first contribution of this work is the introduction of the Bayesian Information Criterion (BIC) [5] as a termination criterion for a bisecting version of the "Incremental *K*-Means" approach. The Incremental *K*-Means (which is repeatedly applied during the bisection process), updates each cluster centroid after each document is assigned to a cluster, rather than after each batch iteration of document assignment to clusters. 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. We terminate the divisive procedure when there is no separable leaf cluster according to the BIC function. Our approach is similar to the X-means algorithm [6], a variant of K-Means, which first used BIC to estimate the best K in a given range of values.

The performance of BIC-Means clustering is evaluated in terms of clustering quality on OHSUMED, a standard medical document collection. As the document representation for OHSUMED, we explored Medical Subject Headings (MeSH) [7]. Each document is represented as a vector of MeSH terms (multi-word composite terms) rather than by vectors of single word terms (thestandard approach) leading to a more compact representation (of lower dimensionality). The MeSH-based document representation is demonstrated to be superior to a pure word-based representation, for the BIC-Means algorithm. Experiments using the *F*-measure criterion [2] show that BIC-Means clustering performs better than both the Bisecting Incremental K-means and the X-Means clustering algorithm.

A second contribution is speeding up retrieval by exploiting the cluster structure. Given a query, instead of searching the entire

document collection (which can be slow), only clusters relevant to the query are searched. Relevance of a cluster is determined by the presence in its centroid vector of all query terms. Our experiments on the OHSUMED data set demonstrate that only about 14% of the data set needs to be searched in cluster-based retrieval, while the results are almost as good as those obtained by exhaustive search of the entire data set.

Typically, clustering applies to static document collections (i.e., new documents are not inserted and old ones are not deleted). As new documents are inserted (or deleted) the quality of clustering declines and the clusters need to be recomputed. Ideally, clustering should never require total re-organization. Maintaining good quality of clustering after insertion or deletion of documents, is of crucial importance.

A third contribution is a dynamic version of BIC-Means, refereed to henceforth as "Dynamic BIC-Means" that adjusts clusters, supporting insertions in the document collection. It requires no re-organization, while maintaining the good quality of clustering of its static counterpart. Dynamic BIC-Means would be suitable for clustering 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. The Dynamic BIC-Means algorithm will keep dynamic corpora or databases organized in the presence of insertions. Inspired by BIC-Means, Dynamic BIC-Means works by deciding which leaf cluster to insert a document into and updating the BIC value of this cluster and of its ancestor nodes. This process might incorporate splitting of existing leaf clusters.

The remainder of this paper first presents related work in the field of data clustering along with resources and processes used in the definition of BIC-Means in Section 2. Subsequently, we present the BIC-Means algorithm along with its dynamic counterpart in more detail in Section 3. Then, experimental results are presented in Section 4 followed by conclusions and issues for future work in Section 5.

2. Related Work and Background

In this section, we provide an overview of the document clustering techniques that are important for this work.

2.1 Document Representation

The various clustering algorithms are described and implemented based upon the Vector Space Model (VSM) [8] for measuring document similarity. In this model, each document d is represented as a vector in a multi-dimensional term space. Each dimension of the space corresponds to a unique term 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, after stop-word removal. 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. The weight w_t for each term t in document d_i is defined as:

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

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 term t occurs. To account for documents of different lengths, we scale the length of each document vector so that it is of unit length. A cluster, which is a set of documents, is represented by its centroid vector (i.e., the mean vector of its documents). We measure the similarity between two documents and (or between a document and a centroid vector) using the cosine formula [8]:

$$\cos(\vec{d}_i,\vec{d}_j) = \frac{\vec{d}_i \bullet \vec{d}_j}{\left\|\vec{d}_i\right\|\left\|\vec{d}_j\right\|} \quad (2)$$

Since document vectors are of unit length, the above formula is simplified to $\cos(\vec{d}_i,\vec{d}_j) = \vec{d}_i \bullet \vec{d}_j$.

2.2 Basic *K*-Means algorithm

K-Means takes as input a data set and K and creates a flat, non-hierarchical clustering solution that consists of K clusters. Initially, the algorithm picks K documents (at random) as initial centroids and computes the similarity between each document and all cluster centroids according to Eq. 2. Each document is assigned to its closest cluster centroid. The next step is the "centroid re-computation": All documents assigned to the same centroid are averaged to compute a new centroid. The clusters (and their centroids) are adjusted iteratively by the algorithm until convergence (i.e., the centroids do not change significantly).

In most implementations, K-Means algorithm continues iterating 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 (or maximum cluster cohesion), the algorithm runs at most *ITER* iterations, where *ITER* is user defined. Our experimental results indicated that maximal cluster cohesion is obtained for *ITER*=5 or 6. Cluster cohesion (overall cluster similarity) is defined as the sum of the average pairwise similarities between all documents assigned to a cluster and equals to $\left\|c^2\right\|$. The overall cohesion of a clustering solution is then computed as:

$$\text{Overall Clustering Cohession} = \sum_{r=1}^{K}\left\|\vec{c}\right\|^2 \quad (3)$$

Notice that even though the document vectors are of length one, the centroids vectors are not.

2.3 Incremental *K*-Means

In K-Means during an iteration the centroids remain fixed. New centroids are computed after all documents have been reassigned to clusters. Incremental K-Means updates centroids after each document is assigned to a cluster. This way the clusters adjust 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 incrementally update the centroid of a cluster after a new document is assigned to the cluster.

$$\vec{C}_{updated} = \frac{\left(\vec{C}_{old}\left(|S|-1\right)\right)+\vec{d}_{assigned}}{|S|} \quad (4)$$

where $|S|$ is the number of documents, $\vec{C}_{updated}$ is the updated centroid, $\vec{C}_{old}$ is the centroid before the assignment of the new document, $\vec{d}_{assigned}$ is the document which is added to the cluster and $|S|$ is the new size of the cluster. The time required to

update the centroid is constant. After the algorithm has stopped iterating, the final centroids are used to generate the final clusters, by assigning each document to its closest centroid. Incremental K-Means requires fewer iterations to produce an acceptable clustering result (i.e., one or two iterations are typically sufficient). Furthermore, Incremental K-Means is not as sensitive to the initialization of the algorithm [9].

2.4 Bisecting *K*-Means

Partitional algorithms can be used to obtain hierarchical clustering solutions via a sequence of repeated applications. Bisecting K-Means algorithm starts with a single cluster of all the documents and works by applying K-Means to split this cluster into two sub-clusters (bisecting step). Then, each of the two sub-clusters is further bisected until a pre-defined criterion is met (e.g. a total of K clusters is produced) or until leaf clusters are singletons (i.e., they contain a single document). At each iteration, we can select for further clustering 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 [2] indicated small differences between these possible methods. The run time of the algorithm is $O(N \log_2 N)$ where N is the number of documents in the collection. Thus it is an efficient technique for hierarchically clustering large data sets.

2.5 Bayesian Information Criterion (BIC)

As mentioned before, Bisecting K-Means terminates when a stopping criterion is met. If the entire hierarchy is required, bisecting continues until singleton leafs are obtained. Other termination criteria include: partitioning until the desired number of leaf clusters is reached [2]; stopping the splitting of a cluster if it contains less that 5% of the total number of documents [10]; stopping the splitting when a min-max cut criterion is met [11].

The Bayesian Information Criterion (BIC) or Schwarz Criterion is a parametric measure of how well a given model predicts the data. It represents a trade-off between the likelihood of the data under the model and the complexity of the model. A model with more parameters can predict the data better, but it may result in over-fitting. BIC has been used in the X-Means algorithm to choose the optimal number of clusters in a given range of values according to intrinsic properties of a given data set [6]. An equivalent technique called Minimum Description Length (MDL) is applied in [12]. Below, we propose the use of BIC in deciding when to stop splitting in hierarchical clustering and we describe how BIC is computed.

Let D be a set of documents $\left\{\vec{x}_1, \vec{x}_2, ..., \vec{x}_n\right\}$. D can be partitioned into disjoint subsets $D_1, D_2, ..., D_K$. In the case of Bisecting K-Means $K = 2$. Documents in each cluster are assumed to follow a Gaussian distribution $N\left(\vec{\mu}, \sigma^2\right)$. Let $\vec{\mu}_j$ 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, $\vec{\mu}_{(i)}$ 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 $\vec{\mu}_j$ 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 a model M_j (i.e., in our case the parent cluster or the two children clusters together) is:

$$BIC(M_j) = \hat{l}_j(D) - \frac{p_j}{2}\log R \quad (5)$$

where $\hat{l}_j(D)$ is the log-likelihood of the data according to the model M_j, while p_j is the number of independent parameters in M_j. The BIC contains two components. The first term (log-likelihood of the documents) can be used as a measure of the cohesion of a cluster used to decide whether a cluster should split or not. We estimate how close to the centroid are the documents of the cluster. More specifically, given a cluster of points, and a spherical Gaussian distribution $N\left(\vec{\mu}, \sigma^2\right)$, log-likelihood is the probability of the set of data points, if they are assumed to be independent samples of this distribution. The second term penalizes the complexity of the model. In the case of the parent cluster, K is set to 1 while in the case of the two children clusters, K is set to 2. M is the number of dimensions (terms) in the representations of documents. The standard deviation of the assumed spherical Gaussian distribution must be estimated from the data points themselves. The maximum likelihood estimate for the variance is given by:

$$\hat{\sigma}_n^2 = \frac{1}{M\left(R-1\right)}\sum_i \left\| x_i - \vec{\mu}_{(i)} \right\|^2 \quad (6)$$

The maximum log-likelihood of the data in a single cluster D_n is computed as:

$$\hat{l}(D_n) = -\frac{R_n}{2}\log(2\pi) - \frac{R_n M}{2}\log\left(\hat{\sigma}_n^2\right) - \frac{M(R_n - 1)}{2} + R_n \log R_n - R_n \log R \quad (7)$$

where R_n denotes the number of documents in cluster D_n. The maximum log-likelihood must be computed separately for the parent cluster and for the two children clusters. In Eq. 7, $K = 1$, as it pertains to the log-likelihood of a single cluster. The maximum likelihood estimate (MLE) of the variance $\hat{\sigma}^2$ is computed separately for each cluster according to Eq. 6. To apply Eq. 5 to the model with two children clusters, we use the fact that the overall log-likelihood is the sum of the log-likelihood of the individual clusters. Thus, Eq. 5 can be re-written as:

$$BIC(M_2) = \sum_{i=1}^{2}\left(\hat{l}_i(C_i)\right) - \frac{p_2}{2}\log R \quad (8)$$

where $C_i, i = 1,2$ are the two children clusters. The number of free parameters p_2 in model M_2 is the sum of $K-1$ class probabilities, MK centroid coordinates, and K variance estimates ($K = 2$). According to Eq. 7, as $\hat{\sigma}^2$ increases, the likelihood decreases and therefore the BIC score (see Eq. 5) decreases. As a result the parent cluster must be partitioned. We conclude that $\hat{\sigma}^2$ is the key determinant of the BIC score of a cluster, which makes sense if interpreted as a measure of the cohesion of the cluster.

2.6 Dynamic Clustering

Clustering methods such as those described above are static, i.e., they perform well as long as updates are avoided. They can accommodate a small number of updates, e.g. by inserting new documents to the most similar cluster, but their performance in terms of clustering quality drops as the number of updates increases and the clusters need to be recomputed.

The pseudo-dynamic approach [13] suggests re-clustering 5% of the oldest documents in the collection for each document insertion. Document deletions are handled similarly. The algorithm is slow for large data sets. The Incremental Hierarchical Clustering [14] method works by computing the cluster in which to insert a new document and by updating the entire hierarchy upwards to the root. It does not support document deletions.

Recent work on clustering dynamic document collections includes the Dynamic *K*-Windows algorithm [15] that supports both document insertions and deletions. The algorithm works by randomly initializing *k* windows (i.e., *d*-dimensional hypercubes that contain documents) over the dataset and by moving and enlarging the window until the number of included elements does not change significantly. If two windows are considered to belong to the same cluster they are merged. Clusters are not allowed to overlap. The algorithm produces a flat clustering. The main weakness of the approach is that *k* (i.e., number of windows) needs to be specified in advance. The On-line Star Algorithm [16,17] considers the documents in the collection as nodes in a graph and organizes the nodes in star-shaped sub-graphs consisting of centers and "satellites" (i.e., nodes that are not centers). When a vertex is inserted (or deleted) new stars (clusters) may need to be created or existing stars may need to be removed. It supports both insertions and deletions.

3. BIC-Means Clustering Algorithm

The BIC-Means clustering algorithm is based upon the idea of using BIC as the splitting criterion when forming a binary tree of clusters with Incremental *K*-Means. Initially, Incremental *K*-Means bisects the entire collection into two clusters. Then, one of the two clusters is selected and further bisected, leading to a total of three clusters. In the exhaustive version of the algorithm used in this paper, the process of selecting and bisecting leaf clusters continues until leaf clusters are singletons (i.e. they contain a single document). To prevent over-splitting of clusters 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 and the parent node is a terminal leaf cluster in the final cluster hierarchy. Consequently, BIC-Means terminates when there is no separable leaf cluster according to BIC. The proposed BIC-Means algorithm is presented in Fig. 1.

Input: K=2 in Incremental K-Means, document collection $S:\left\{\vec{d}_1,\vec{d}_2,...,\vec{d}_n\right\}$

Output: A hierarchy of clusters.

1. Treat all documents as a single initial leaf cluster;
2. Pick a leaf cluster C to split from the list of leaf clusters. Choose the cluster with the least overall similarity;
3. Bisecting Step: Use Incremental K-Means to split cluster C into two sub-clusters, C_1 and C_2;
4. Compute the BIC scores for two distinct models, one for the initial cluster C and another for the two resulting clusters, C_1 and C_2;
5. If $BIC(C) > BIC(C_1, C_2)$ then accept the split of C and add C_1, C_2 as leaf clusters with C as parent, else keep C as a terminal leaf (no further splitting);
6. Repeat steps 2-5, until there is no separable leaf cluster according to the BIC.

Figure 1. The BIC-Means algorithm

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 efficient technique for hierarchically clustering large datasets.

3.1 Dynamic BIC-Means

After the original document collection has been clustered, a new document is inserted to the leaf cluster C most similar to it. Each cluster is represented by its centroid vector; document to cluster similarity is computed by Eq. 2. The BIC value of cluster C is updated, which, in turn, may cause the splitting C if its BIC value is less than the cumulative BIC of its children clusters. Fig. 2 summarizes this process.

Input: document $\vec{x}$

Output: Updated cluster hierarchy.

1. Insert $\vec{x}$ to the most similar leaf cluster *C*;
2. Split *C* into C_1 and C_2;
3. Add $\vec{x}$ to the closest child cluster of *C*;
4. Update BIC of cluster *C*;
5. Update BIC of ancestor nodes (up to the root cluster);
6. Compute cumulative BIC of children nodes;
7. If $BIC(C) < BIC(C_1, C_2)$ then accept the split and add C_1, C_2 to the list of leaf clusters;
8. Else keep *C* as the terminal leaf.

Figure 2. Dynamic BIC-Means algorithm

The algorithm involves time consuming processes such as (a) Bisecting of a leaf cluster and (b) Computing the BIC values of a leaf cluster and of its descendant nodes which takes time linear to the number of documents in a cluster. To reduce time complexity, the following optimizations are implemented:

- If the split is rejected (steps 7-8 in the algorithm above), the splitting of a cluster is cached, as it may be needed again whenever a new document is inserted to *C*.
- Document insertions may cause nodes higher in the hierarchy to be updated. We have chosen to ignore such updates (as only the leaf nodes take part in the clustering solution).
- The existing BIC value of a node is updated to accommodate the insertion of a new document $\vec{x}$ (rather than computing BIC over the entire set of n+1 documents in the cluster).

The updated variance in the BIC computation for a node with n+1 documents is computed based on Eq. 5 in $O(1)$ time. This requires computing $\hat{l}(D_{n+1})$ from $\hat{l}(D_n)$ where D_n is the set of documents in cluster *C* and set D_{n+1} includes the new document. This in turn requires computing $\hat{\sigma}^2_{n+1}$ from $\hat{\sigma}^2_n$ in $O(1)$ time as follows:

$$\hat{\sigma}^2_{n+1} = \frac{1}{n}\left\{\hat{\sigma}^2_n + (\vec{x}-\vec{\mu}_{n+1})(\vec{x}-\vec{\mu}_n)\right\} \quad (9)$$

where $\vec{\mu}_{n+1} = \vec{\mu}_n + \left(\vec{x}-\vec{\mu}_n\right)/n$ and μ_n is the centroid of D_n

4. Experimental Results

We carried out three different sets of experiments on the OHSUMED collection [18]. OHSUMED is a collection of MEDLINE document abstracts used for benchmarking information retrieval systems evaluation. MEDLINE constitutes the primary medical repository of the U.S. National Library of Medicine. MEDLINE documents are currently indexed by

human experts, based on a controlled list of indexing terms, the MeSH[1] thesaurus. Each document is assigned a set (typically between 10-12) of MeSH terms by human indexers. In our work, MeSH terms are augmented by additional MeSH terms extracted from the title and the abstract field of each document [19]. In the following, we compare BIC-Means (the proposed method) with Incremental Bisecting *K*-Means (the most successful variant of *K*-Means), the benchmark method in all our experiments. The purpose of these experiments is to assess [20]:

- The performance of BIC-Means: BIC-Means performs at least as well as Incremental Bisecting *K*-Means (producing exactly the same clustering results, only faster).
- The performance of Dynamic BIC-Means: Dynamic BIC-Means approaches the clustering solutions of the standard *K*-Means approaches such as the Incremental Bisecting *K*-Means.
- The performance of cluster-based retrieval: It is possible to exploit the results of clustering for speeding-up the retrieval response times in medical document collections.

4.1 Document Clustering Experiments with BIC-Means

This first set of experiment focuses on the evaluation of the clustering quality of BIC-Means compared with Bisecting Incremental *K*-means. *F*-Measure was used to estimate the overall "goodness" of the generated clusters [2]. All clustering techniques are tested on a subset of the OHSUMED collection (10902 documents into 10 classes) that comes with a predefined categorization of its content [4]. We carried out 10 separate runs for each document clustering evaluation with randomly selected initial cluster centroids. Therefore, the experimental results reported here correspond to the average *F*-Measure over ten runs.

We compare the performance of Bisecting Incremental *K*-Means using a vector space representation of documents defined using (a) single words terms, as it is typical in retrieval experiments, and (b) vectors of MeSH terms. In the Incremental *K*-Means method ITER was set to 1 and the divisive procedure terminated when each leaf cluster contained a single document. We observe that our Bisecting Incremental *K*-Means method yields better *F*-Measure when the OHSUMED documents are represented by MeSH terms than by single word terms (i.e., 0.69 and 0.61 respectively). In terms of clustering time, the Bisecting Incremental *K*-means algorithm needs 14 minutes to run for the MeSH representation, whereas, for the single word terms representation, the clustering hierarchy was obtained in 97.6 minutes. This happened due to the size of the document vectors. In the case of single word term representation, a document contains about 80-100 terms, while in the case of MeSH based representation each vector contains about 20 distinct MeSH terms. Based on this result, we use vectors of MeSH terms as document representations in all experiments below.

Next, we compare BIC-Means against Bisecting Incremental *K*-Means, using the MeSH representation for documents. BIC-Means achieved *F*-Measure 68.73% versus 69.03% for the Bisecting Incremental *K*-Means method. The difference is only 0.3%. In terms of clustering time, BIC-Means took 9.5 minutes, whereas Bisecting Incremental *K*-Means required 14 minutes.

Overall, BIC-Means gives similar *F*-Measure values to Bisecting Incremental *K*-Means, with significantly less computation time. In addition, BIC-Means generates meaningful clusters as compared to Bisecting Incremental *K*-Means method that exhaustively generates a cluster hierarchy down to singleton clusters. A similar set of experiments on the Reuters-21578[2] data set confirmed the efficiency of BIC-Means [20].

4.2 Retrieval using Document Clusters

A number of studies [1,21,22] have suggested applying clustering to improve retrieval results. Some experimental results [1,23] have demonstrated that retrieval from the clusters most relevant to the query is more effective than retrieval from the entire document corpus, while others [24] have indicated that exhaustive retrieval is generally more effective. In most of these early experiments the size of document collections used was limited by the time and space requirements of hierarchical clustering. There are no conclusive results on larger data sets. In this study, we compare cluster-based retrieval with exhaustive retrieval on collections of medical documents of a significant size.

For this experiment the entire OHSUMED collection with 233,445 Medline abstracts was used. For cluster-based retrieval, documents are clustered using the BIC-Means method. Each cluster is represented by its centroid vector. Documents and queries are represented by MeSH term vectors as in the previous experiment and document matching is performed by Eq. 2. The results were evaluated against the TREC provided queries and answers. These constituted the ground truth for evaluating retrieval performance. The query set consisted of 61 queries (after removing 45 queries either because they contained no MeSH terms or had no relevant documents associated with them). Each query contained between 1 and 6 MeSH terms. 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 top 100 ranked answers.

The quality of the retrieval results is measured in terms of *recall (R)* (i.e., ratio of relevant documents that are retrieved) and *precision (P)* (i.e., ratio of retrieved documents that are relevant). For each query, the best 100 answers were retrieved so that, the precision/recall plot of each method contains exactly 100 points. Each point corresponds to average precision and recall of the top *n* results over the 61 queries, where *n* varies from 1 to 100.

Several retrieval strategies are evaluated in [20]. In the most successful variant (presented here), only the leaf clusters which contain all the MeSH terms of the query in their centroid vector are searched. The query is matched with the documents contained in these clusters and the 100 highest ranked documents (ordered by decreasing similarity with the query) are used to compute precision and recall. We also evaluated retrieval strategies that searched the top *n*=15, 30, 50 most similar clusters (to the query) that contained all the query terms, referred to as *AllQinCen_AllClusters, AllQinCen_Top_50Clusters,AllQinCen_Top_30Clusters* and *AllQinCen_Top_15Clusters* respectively. Fig. 3 illustrates the performance of all these methods in addition to the performance of exhaustive search, the benchmark method for this experiment. As expected, exhaustive search, outperforms all other methods. *AllQinCen_AllClusters* is almost as effective as exhaustive search but achieved this performance by searching 83 (out of 633) leaf clusters holding only 14% of the entire data set (approximately 35,000 documents). Notice that, the performance of cluster-based search improves with *n*: As *n* increases, so does the search space and retrieval

[1] http://www.nlm.nih.gov/mesh

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

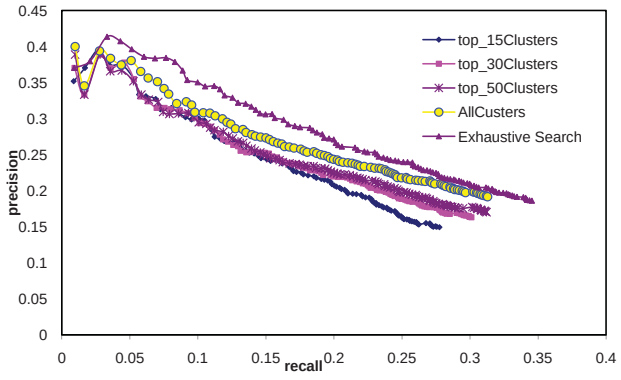

Figure 3. Precision-recall curves of exhaustive and cluster-based search using clusters containing all query terms in their centroid vectors

reaches the performance of exhaustive search. Obviously, the *AllQinCen_AllClusters* is the preferred method for searching in clustered document collections.

5. Document Clustering Experiment with Dynamic BIC-Means

We evaluate the performance of Dynamic BIC-Means in a dynamic environment where clustering is updated after each document addition according to the method of Fig. 2. The term vector of each document is compared with the centroid vectors of all leaf clusters and is inserted into the cluster with the most similar centroid. Leaf clusters may split as their BIC values are updated by Eq. 5.

Fig. 4 illustrates measurements of clustering quality *F* for the Dynamic BIC-Means, BIC-Means and *X*-Means [6] algorithms. BIC-Means and *X*-Means are incapable of handling updates. They compute a new clustering solution (involving the entire collection) every time the dataset grows by 1,000 documents. Dynamic BIC-Means, accommodates insertion of new documents simply by updating the BIC values of the clusters accepting the new document (which may involve the splitting of the cluster). The dynamic version of BIC-Means approximates the performance of its static counterpart within 15% in the worst case (for large document sizes) and within less than 5%for smaller data sets. Notice that both versions of BIC-Means perform better than *X*-Means.

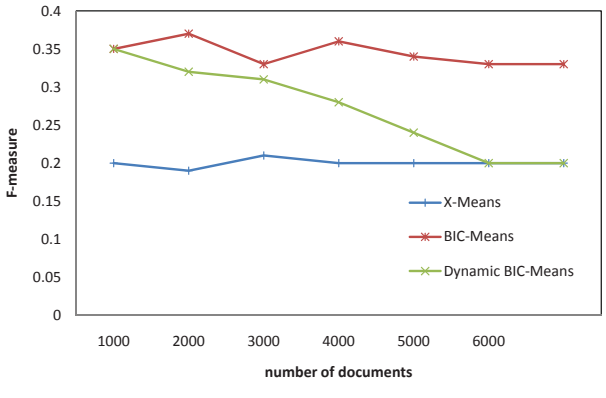

Figure 4. Clustering quality (*F*) of Dynamic BIC-Means, BIC-Means and X-Means as dataset size increases from 0 to 7000 documents

6. Conclusions

The contributions of this work span three directions: The first contribution of this paper is the proposal of BIC-Means, a hierarchical clustering algorithm based on Bisecting Incremental *K*-Means which determines when to stop splitting clusters using the Bayesian Information Criterion (BIC). The BIC score measures the improvement of a cluster when it is split. The divisive procedure terminates when there is no separable leaf cluster according to BIC. As shown in the experiments, BIC-Means achieves almost the same clustering quality as the exhaustive version of Bisecting Incremental *K*-Means.

The second contribution of this paper is the proposal of a cluster-based information retrieval strategy for medical collections, where documents are represented by their MeSH terms and retrieval is carried out only on a selected set of document clusters (the centroids of which contain all MeSH terms of the query). The experimental results demonstrate that the representation of medical documents using MeSH terms is more effective for clustering and retrieval than a representation with single word terms.

The third contribution of this paper is to the proposal of a dynamic version of the BIC-Means algorithm, a heuristic clustering approach for adapting an existing clustering solution to updates (i.e., document insertions). Dynamic BIC-means work by introducing cluster splitting whenever the BIC criterion of cluster cohesion is violated. Dynamic BIC-Means performs almost equally well with its static counterpart (the BIC-Means approach) without requiring cluster reorganization, but much faster.

In future work, other parameter-free algorithms are known to exist, to which BIC-means should be compared. The *G*-means algorithm [25] runs *k*-means with increasing *k* until a statistical test accepts the hypothesis that the clusters are Gaussian. Robust information-theoretic clustering [26] is based on the Minimum Description Length (MDL) principle. It purifies clusters obtained by a standard clustering algorithm from noise and adjusts the clustering to determine both the number and the shape of the clusters. It consists of a robust fitting algorithm and a cluster merging algorithm. Furthermore, it would be interesting to apply BIC-Means on the entire Medline database. Medline contains more that 16 million references (version 2008) to journal articles in life sciences, medicine and bio-medicine and is still expanding. Hierarchical clustering of MEDLINE would improve the effectiveness of retrievals and provide support to intuitive browsing, navigation and summarization of medical information.

Acknowledgements

Special thanks to Angelos Hliaoutakis, Epimenidis Voutsakis, and Giannis Varelas for their input to this work.

References

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

[2] Steinbach, M., Karypis, G., Kumar, V. (2000). A Comparison of Document Clustering Techniques. *In: KDD Workshop on Text Mining*, p. 188–196, Boston, USA, August.

[3] Ding, C., He, X., Zha, H., Gu, M., Simon, H. (2001). A Min-Max Cut Algorithm for Graph Partitioning and Data Clustering. *In: IEEE Intern. Conf. on Data Mining (ICDM01)* p. 107–114, San Jose, California, USA, Nov-Dec.

[4] Zhao, Y., Karypis, G. (2002). Evaluation of Hierarchical Clustering Algorithms for Document Datasets. *In: Int. Conf. on Information and Knowledge Management (CIKM02)* p. 515–524, Chicago, USA.

[5] Schwarz, G. (1978). Estimating the Dimension of a Model. *Ann. Statistics*, 6:461–464.

[6] Pelleg, D., Moore, A. (2000). X-means: Extending *K*-means with Efficient Estimation of the Number of Clusters. *In: Intern. Conf. on Machine Learning*, p. 727734, San Francisco, USA.

[7] Douglas Johnston, W., Stuart, J. Nelson., Betsy, L. Humphreys. (2001). Relationships in Medical Subject Headings (MeSH).

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

[9] Larsen, B., Aone, C. (1999). Fast and effective text mining using linear-time document clustering. *In: Intl Conference on Knowledge Discovery and Data Mining (ACM SIGKDD)* p. 16–22, San Diego, California, USA, August.

[10] Zhao, Y., Karypis, G. (2004). Soft Clustering Criterion Functions for Partitional Document Clustering . Technical Report TR#04-022, Dept. of Comp. Science, Univ. of Minnesota, Minneapolis.

[11] Ding, C., He, X. (2002). Cluster Merging and Splitting in Hierarchical Clustering Algorithms. *In: IEEE Intern. Conf. on Data Mining (ICDM02)*, Maebashi City, Japan, December.

[12] Nomoto, T., Matsumoto, Y. (2001). A New Approach to Unsupervised Text Summarization. *In: ACM Conf. on Research and Development in Information Retrieval (SIGIR01)* p. 26–34, Orleans, USA, August.

[13] Elmore, M. T., Reed, J. W., Potok, T. E., Patton, R. M. (2005). Real-Time Document Cluster Analysis for Dynamic Data Sets. *In: Proc. IPSI*, p. 586–591, Amalfi, Italy.

[14] Ribert, A., Ennaji, A., Lecourtier, Y. (1999). An Incremental Hierarchical Clustering. *In: Proc. of Vision Interface (VI'99)* p. 586–591, Trois-Rivieres, Canada.

[15] Tasoulis, D. K., Vrahatis, M. N. (2005). Unsupervised Clustering on Dynamic Databases. *Pattern Recognition Letters*, 26 (13) 2116–2127.

[16] Aslam, J., Pelekhov, K., Rus, D. (1999). A practical clustering algorithm for static and dynamic information organization. *In: SODA '99: Proc. of 10th annual ACM-SIAM symposium on Discrete algorithms*, p. 51–60, Philadelphia, PA, USA. Society for Industrial and Applied Mathematics.

[17] Aslam, J., Pelekhov, K., Rus, D. (2004). The Star Clustering Algorithm for Static and Dynamic Information Organization. *Journal of Graph Algorithms and Applications*, 8 (1) 95–129.

[18] Hersh, W., Buckley, C., Leone, T. J., Hickam, D. (1994). OHSUMED: An Interactive Retrieval Evaluation and New Large Test Collection for Research. *In: Int. Conf. on Research and Development in Information Retrieval (SIGIR99)* p. 192–201, Berkeley, California, USA, August.

[19] E.G.M. Petrakis A. Hliaoutakis, Zervanou, K. (2007). Medical Document Indexing and Retrieval: AMTEx vs. NLM MMTx. *In: 12th International Symposium for Health Information Management Research*, Sheffield, UK, July.

[20] Hourdakis, N. (2006). Design and Evaluation of Clustering Approaches for Large Document Collections, the BIC-Means Method. Technical report, Dept. of Electronic and Computer Engineering, Technical Univ. of Crete (TUC), Chania, Crete, Greece. Master's thesis.

[21] Van Rijsbergen, C. J., Croft, W. B. (1975). Document Clustering: An Evaluation of Some Experiments with the Cranfield 1400 Collection. *Information Processing and Management* (11) 171–182.

[22] Van Rijsbergen, C. J. (1979). *Information Retrieval*. Butterworths, London.

[23] Croft, W. B. (1980). A Model of Cluster Searching Based on Classification. *Information Systems* (5)189–195.

[24] Voorhees, E. M. (1985). The Cluster Hypothesis Revisited. *In: ACM Intern. Conf. on Research and Development in Information Retrieval (SIGIR85)* p.188–196, Montreal, Canada.

[25] Hamerly, G., Elkan, C. (2003). Learning the *K* in *K*-means. *In: Advances in Neural Information Processing Systems*, v.17.

[26] Boehm, C., Faloutsos, C., Pan, J-Y., Plant, C. (2007). RIC: Parameter-Free Noise-Robust Clustering. *ACM Transactions on Knowledge Discovery from Data*, 1 (3) 10: 1–10:29, December.