Dynamic Hierarchical Compact Clustering Algorithm

Reynaldo Gil-García[1], José M. Badía-Contelles[2], and Aurora Pons-Porrata[1]

[1] Center of Pattern Recognition and Data Mining,
Universidad de Oriente, Santiago de Cuba, Cuba
{gil, aurora}@app.uo.edu.cu
[2] Universitat Jaume I, Castellón, Spain
badia@icc.uji.es

Abstract. In this paper we introduce a general framework for hierarchical clustering that deals with both static and dynamic data sets. From this framework, different hierarchical agglomerative algorithms can be obtained, by specifying an inter-cluster similarity measure, a subgraph of the β-similarity graph, and a cover algorithm. A new clustering algorithm called *Hierarchical Compact Algorithm* and its dynamic version are presented, which are specific versions of the proposed framework. Our evaluation experiments on several standard document collections show that this algorithm requires less computational time than standard methods in dynamic data sets while achieving a comparable or even better clustering quality. Therefore, we advocate its use for tasks that require dynamic clustering, such as information organization, creation of document taxonomies and hierarchical topic detection.

1 Introduction

Managing, accessing, searching, and browsing large repositories of text documents requires efficient organization of the information. In dynamic information environments, such as the World Wide Web or the stream of newspaper articles, it is usually desirable to apply adaptive methods for document organization such as clustering.

Hierarchical clustering algorithms have an additional interest, because they provide a view of the data at different levels of abstraction, making them ideal for people to visualize and interactively explore large document collections. Besides, clusters very often include subclusters, and the hierarchical structure is indeed a natural constraint on the underlying application domain.

Static clustering methods mainly rely on having the whole collection ready before applying the algorithm. Unlike them, the incremental methods are able to process new data as they are added to the collection. In addition, dynamical algorithms have the ability to update the clustering when data are added or removed from the collection. These algorithms allow us dynamically tracking the ever-changing large scale information being put or removed from the web everyday, without having to perform complete reclustering.

M. Lazo and A. Sanfeliu (Eds.): CIARP 2005, LNCS 3773, pp. 302–310, 2005.

Several incremental clustering algorithms have been proposed (e.g. see [6]). However, these algorithms do not create cluster hierarchies. On the other hand, various hierarchical algorithms have been used for clustering [11], but all of them are static algorithms. Finally, there are a few algorithms that update the cluster hierarchy when a new object arrives, such as GALOIS [3], Charikar's algorithm [5] and DC-tree [10]. These algorithms have several of the following drawbacks: its time complexity is exponential with the dimension of the objects, the number of clusters is fixed a priori, the obtained clusters depend on the data order, they require tuning several parameters and they impose restrictions to the representation space of the objects and to the similarity function.

In this paper we introduce a general hierarchical framework that deals with both static and dynamic data sets. From this framework, different hierarchical agglomerative algorithms can be obtained, by specifying an inter-cluster similarity measure, a subgraph of the β-similarity graph, and a cover algorithm. We also propose a new clustering algorithm called *Hierarchical Compact Algorithm*, which is a specific variant of this framework. This algorithm is compared with other hierarchical clustering methods using four standard document collections. Our evaluation experiments show that this algorithm requires less computational time in dynamic data sets than standard methods while achieving a comparable or even better clustering quality.

2 Static Hierarchical Clustering Algorithm

We call β-similarity graph the undirected graph whose vertices are the clusters and there is an edge from vertex i to vertex j, if the cluster j is β-similar to i. Two clusters are β-similar if their similarity is greater or equal to β, where β is a user-defined parameter. Analogously, i is a β-isolated cluster if its similarity with all clusters is lesser than β.

The clustering algorithms based on graphs involve two main tasks: the construction of a certain graph and a cover routine of this graph that determines the clusters. In this context, a cover for a graph $G = (V, E)$ is a collection $V_1, V_2, ..., V_k$ of subsets of V such that $\cup_{i=1}^{k} V_i = V$, each one representing a cluster.

Our hierarchical clustering algorithm is an agglomerative method and it is based on graph too. It uses a multi-layered clustering to produce the hierarchy. The granularity increases with the layer of the hierarchy, with the top layer being the most general and the leaf nodes being the most specific. At each successive layer of the hierarchy, vertices represent subsets of their parent clusters. The process in each layer has two steps: the construction of a graph and a cover routine of this graph. The general framework is shown in Figure 1.

In our framework, a similarity measure to compare the objects and an inter-cluster similarity measure are required. The algorithm starts with each object being considered a cluster. Then, it constructs a subgraph of the β-similarity graph. The set of vertices of this subgraph must equal to the set of vertices of the graph. A cover routine is applied to this subgraph in order to build the clus-

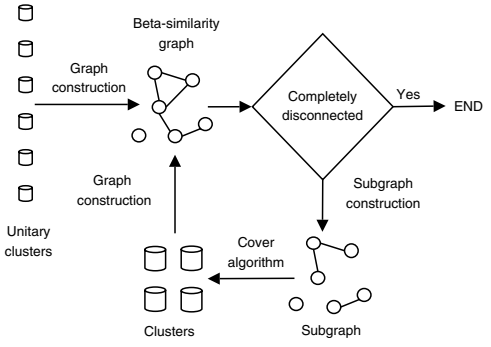

Fig. 1. General framework

ters in the bottom layer. From the obtained clusters, the algorithm constructs a new β-similarity graph and its corresponding subgraph. Then, the cover routine is applied again to obtain the clusters in the next layer. This process is repeated until the β-similarity graph is completely disconnected, that is, all vertices (clusters) of the graph are β-isolated. In our framework, the cover routine should not depend on the order of the incoming objects. This requirement guarantees the order independence of the framework. Notice that we use the same β value and a unique subgraph type in all levels of the hierarchy.

We can obtain disjoint or overlapped clusters at each level of the hierarchy, depending on the cover routine used. It is worth noticing that if we change the type of subgraph, the similarity measures or the cover routine in this general framework, different hierarchical agglomerative algorithms are obtained. In our algorithm, unlike the traditional hierarchical agglomerative algorithms, several clusters can be merged in the same level. Also, since our stopping criterion is that the graph is completely disconnected, the top level of the hierarchy does not necessarily consist of one cluster.

Traditional hierarchical agglomerative algorithms can be obtained as particular cases of the previous general framework if we choose $\beta = 0$, the subgraph should be the mutual nearest neighbour subgraph of the β-similarity graph, and the cover routine should find the connected components in this subgraph.

2.1 Hierarchical Compact Algorithm

In this paper we propose a specific variant of the abovementioned framework. We will call it *Hierarchical Compact Algorithm* (*HCA*). This algorithm assumes the following issues:

1. The group-average as inter-cluster similarity measure.
2. The subgraph is the maximum β-similarity graph [4] disregarding the orientation of its edges (namely undirected $max - S$ graph).
3. The cover routine finds the connected components of the undirected $max - S$ graph, that is, the compact sets [9].

The main steps of the method are shown in the Algorithm 1.

Algorithm 1 Static Hierarchical Compact Algorithm.

1. Put each object in a cluster on its own.
2. $level = 0$.
3. Construct the β-similarity graph, G_{level}.
4. While G_{level} is not completely disconnected:
   (a) Construct the undirected $max - S$ graph (subgraph of G_{level}).
   (b) Find the connected components of this subgraph.
   (c) Construct a new β-similarity graph, $G_{level+1}$, where each vertex represents a connected component and the inter-cluster similarity is group-average.
   (d) $level = level + 1$

The proposed algorithm can produce clusters with arbitrary shapes and the generated set of clusters at each level of the hierarchy is unique, independently on the arrival order of the objects. Also, since we use the maximum β-similarity graph, the algorithm produces cohesive clusters. In our algorithm, the number of clusters is not fixed. Besides, it requires a unique parameter and therefore it reduces the problem of tuning the parameter values to suit specific applications. The computational complexity of the *HCA* algorithm is $O(n^2)$.

3 Dynamic Hierarchical Clustering Algorithm

Our dynamic general framework can be defined in a similar way to the static framework explained above. The main difference is that the construction of the graphs and the cover routine must be dynamic. Given a hierarchy of clusters previously built by the algorithm, each time a new object arrives (or is removed), the clusters at all levels of the hierarchy must be revised. The steps of the method are shown in Algorithm 2.

As it can be noticed, the dynamic algorithm comprises the updating of the graphs and the updating of the cover at each level of the hierarchy. The updating of the β-similarity graph is trivial. The details of the other updating processes are described below, focusing in the *Dynamic Hierarchical Compact Algorithm* (DHCA). In this particular case, we need to update the undirected $max - S$ graph and its connected components at each level of the hierarchy.

When a new object arrives, a new unitary cluster is created and the β-similarity graph of the bottom level is updated. Then, the undirected $max - S$ graph is updated too, which can produce a new vertex and can also produce new edges and remove others (see Algorithm 3). Every time an edge is removed from the undirected $max - S$ graph, the cluster (connected component) to which the vertices of this edge belong can become unconnected. Therefore, that cluster must be reconstructed. On the other side, every time an edge is added to the undirected $max - S$ graph, the clusters of its vertices are merged if they are

Algorithm 2 Dynamic general framework.

1. Arrival of an object to cluster (or to remove).
2. Put the new object in a cluster on its own (or remove the cluster to which the object belongs).
3. $level = 0$.
4. Update the β-similarity graph, G_{level}.
5. While G_{level} is not completely disconnected:
   (a) Update the subgraph of G_{level}.
   (b) Update the cover of this subgraph.
   (c) Update the β-similarity graph, $G_{level+1}$.
   (d) $level = level + 1$
6. If there exist levels greater than $level$ in the hierarchy, remove them.

different. The updating of the connected components produces new clusters and removes others (see Algorithm 4). When clusters are created or removed from a level of the hierarchy, the β-similarity graph of the next level must be updated. This process is repeated until this graph is completely disconnected. It is possible that the β-similarity graph became completely disconnected before the top level of the hierarchy is reached. In this case, the next levels of the hierarchy must be removed.

Algorithm 3 Undirected $max - S$ graph updating.

1. Let N be the set of vertices to add to the undirected $max - S$ graph and R the set of vertices to remove from it.
2. Let M be the set of vertices for which a vertex of R is its most β-similar vertex.
3. Remove all vertices of R from the undirected $max - S$ graph and add all vertices of N to it.
4. Find the most β-similar vertices of each vertex of $M \cup N$ and add the corresponding edges to the $max - S$ graph.
5. Find the vertices for which a vertex of N is its most β-similar vertex and update the corresponding edges.

4 Experimental Results

The performance of the Dynamic Hierarchical Compact Algorithm has been evaluated using four document collections, whose general characteristics are summarized in Table 1. Human annotators identified the topics in each collection. The smallest of these datasets contains 695 documents and the largest contains 10369 documents. To ensure diversity in the datasets, we obtained them from different sources.

The AFP collection is from the TREC-5 conference [1] and it contains some articles published by the AFP agency in 1994 year. The ELN collection contains

Algorithm 4 Connected component updating.

1. Let N be the set of vertices added to the undirected $max - S$ graph and R the set of vertices removed from it. Let, also, NE be the set of edges added to the undirected $max - S$ graph and RE the set of edges removed from it.
2. Let Q be a queue with the vertices to be processed, $Q = \emptyset$.
3. Remove all vertices of R from their clusters. Put the remaining vertices of these clusters into Q. Put, also, all vertices of the clusters where at least one edge of RE is incident to a vertex of the cluster into Q. Remove these clusters from the list of the existing clusters.
4. Put all vertices of N into the queue Q.
5. Build the connected components from the vertices in Q and add them to the list of existing clusters.
6. For each edge of NE, merge the clusters to which its vertices belong.

Table 1. Description of collections

Collection	Source	Documents	Terms	Topics
AFP	TREC-5	695	12575	25
ELN	TREC-4	5829	84344	50
TDT	TDT2	9824	55112	193
REU	Reuters-21578	10369	35297	120

a set of "El Norte" newspaper articles dated from 1994. Both collections are in Spanish. We also use the TDT2 dataset, version 4.0 [2]. This corpus consists of 6 months of news stories from the January to June 1998. The news stories were collected from six different sources. Human annotators identified a total of 193 topics in the TDT2 dataset. 9824 English stories belong to one of these topics, the rest are unlabeled. Finally, from Reuters-21578 [8] we selected the documents that are assigned one or more topics and have <BODY> and </BODY> tags.

In our experiments, the documents are represented using the traditional vectorial model. The terms of documents represent the lemmas of the words appearing in the texts. Stop words, such as articles, prepositions and adverbs are disregarded from the document vectors. Terms are statistically weighted using the term frequency (TF). To account for documents of different lengths, the vector is normalized using the document length. We use the traditional cosine measure to compare the documents.

There are many different measures to evaluate the quality of clustering. We adopt a widely used external quality measure: the *Overall F-measure* [7]. This measure compares the system-generated clusters with the manually labelled topics and combines the precision and recall factors. The higher the overall F-measure, the better the clustering is, due to the higher accuracy of the clusters mapping to the topics.

Our experiments were focused on evaluating the quality of the clustering produced by other well known hierarchical clustering methods: Average-link,

Complete-link and Bisecting K-Means. We compare these algorithms with our Dynamic Hierarchical Compact Algorithm.

The results for the various document collections and methods are shown in Table 2. In our algorithm we only evaluated the top level of the hierarchy and the parameter β that produced the best results was chosen. On the contrary, in the other algorithms we consider the flat partition produced by the best level of the hierarchy.

As it can be noticed, our method is either the best or always near to the best solution. It is worth mentioning that our algorithm obtains these results with both less total number of clusters and levels.

Table 2. Quality results obtained by different clustering algorithms

Data	Algorithm	Levels	Clusters in hierarchy	Clusters in best level	F-Overall
AFP	Average-link	695	1389	40	0.84
	Complete-link	695	1389	29	0.83
	Bisecting K-Means	695	1389	13	0.69
	DHCA($\beta = 0.12$)	3	226	45	0.82
ELN	Average-link	5829	11658	170	0.41
	Complete-link	5829	11658	80	0.41
	Bisecting K-Means	5829	11658	42	0.36
	DHCA($\beta = 0.10$)	4	1033	73	0.46
TDT	Average-link	9824	19645	165	0.77
	Complete-link	9824	19645	255	0.50
	Bisecting K-Means	9824	19645	122	0.40
	DHCA($\beta = 0.12$)	4	2636	136	0.76
REU	Average-link	10369	20737	100	0.53
	Complete-link	10369	20737	180	0.37
	Bisecting K-Means	10369	20737	101	0.23
	DHCA($\beta = 0.12$)	4	2095	101	0.52

Figure 2 shows the time spent by the DHCA algorithm and the three classical hierarchical algorithms mentioned above. Each curve represents the time spent to cluster the document sub-collections of size 1000, 2000 and so on. Since the dynamic nature, our algorithm needs to update the cluster each time a new document arrives, which clearly increases its cost (see curve DHCA-T). However, in a dynamic environment we have a collection partially clustered and some new documents arrive. In this case the static algorithms have to cluster the whole collection again, whereas the DHCA algorithm only needs to update the existing clusters. DHCA-P represents the time spent by our algorithm to update the clusters when adding 1000 documents every time. For example, the value shown with 7000 documents represents the time spent to add 1000 new documents to the clustering when we have already clustered 6000 documents. As we can observe, in this case the DHCA clearly overcomes the static algorithms.

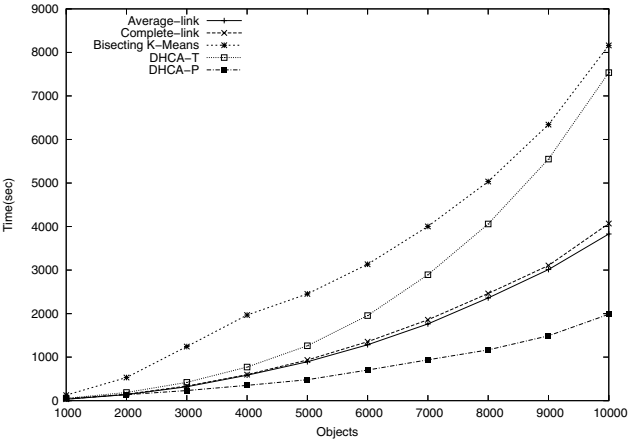

Fig. 2. Time performance

5 Conclusions

In this paper a hierarchical clustering framework, both static and dynamic, has been introduced. This framework is based on the β-similarity graph (relies only on pair-wise document similarity information). Different hierarchical agglomerative algorithms can be obtained from it, by specifying an inter-cluster similarity measure, a subgraph of the β-similarity graph, and a cover algorithm of this subgraph. The traditional hierarchical agglomerative methods can be seen as particular cases of this general framework.

Since in our framework several clusters can be merged at the same level and it stops when the graph is completely disconnected, we can obtain a cluster hierarchy composed by few levels.

A specific variant of the proposed framework, called Hierarchical Compact Algorithm is also introduced. This algorithm obtains cohesive clusters with arbitrary shapes. Another advantage of HCA is that the number of clusters is not fixed and the algorithm requires a unique parameter.

The dynamic version of the Hierarchical Compact Algorithm can be used to organize dynamic data, such as the creation of document taxonomies and the hierarchical topic detection task. Its most important novelty is that it is a dynamic clustering algorithm able to build a cluster hierarchy independent on the data order.

This algorithm was compared with other clustering algorithms in four standard document collections. The experimental results show that our algorithm achieves a comparable or better clustering quality. Moreover, the algorithm achieves better time performance than other traditional hierarchical clustering algorithms in dynamic collections.

Finally, though we employ our algorithm to cluster document collections, it can be also applied to any problem of Pattern Recognition with mixed objects.

Acknowledgements. This work has been partially supported by the research projects Bancaixa (PI-1B2001-14) and CICYT (TIC2002-04400-C03-01).

References

1. Text REtrieval Conference (TREC). http://trec.nist.gov.
2. TDT2 collection, version 4.0, 1998. http://www.nist.gov/speech/tests/tdt.html.
3. C. Carpineto and G. Romano. A lattice conceptual clustering system and its application to browsing retrieval. *Machine Learning 24*, (2):95–122, 1996.
4. J. A. Carrasco-Ochoa, J. Ruiz-Shulcloper, and L. A. De la Vega-Doria. Sensitivity analysis for beta0-compact sets. In *VI Iberoamerican Symposium on Pattern Recognition*, pages 14–19, 2001.
5. M. Charikar, C. Chekuri, T. Feder, and R. Motwani. Incremental clustering and dynamic information retrieval. In *29th Annual Symposium on Theory of Computing*, pages 626–635, 1997.
6. K. M. Hammouda and M. S. Kamel. Efficient phrase-based document indexing for web document clustering. *IEEE Transactions on Knowledge and Data Engineering*, 16(10):1279–1296, 2004.
7. B. Larsen and C. Aone. Fast and effective text mining using linear-time document clustering. In *KDD'99*, pages 16–22, 1999.
8. D. Lewis. Reuters-21578 text collection, version 1.2. http://kdd.ics.uci.edu.
9. A. Pons-Porrata, R. Berlanga-Llavori, and J. Ruiz-Shulcloper. On-line event and topic detection by using the compact sets clustering algorithm. *Journal of Intelligent and Fuzzy Systems*, 3-4:185–194, 2002.
10. W. Wai-chiu and A. Wai-chee Fu. Incremental document clustering for web page classification. In *IEEE 2000 International Conference on Information Society in the 21st Century: Emerging technologies and new challenges*, 2000.
11. Y. Zhao and G. Karypis. Evaluation of hierarchical clustering algorithms for document datasets. In *International Conference on Information and Knowledge Management*, pages 515–524, 2002.