Hierarchical Star Clustering Algorithm for Dynamic Document Collections

Reynaldo Gil-García and Aurora Pons-Porrata

Center for Pattern Recognition and Data Mining
Universidad de Oriente, Santiago de Cuba, Cuba
{gil,aurora}@cerpamid.co.cu

Abstract. In this paper, a new clustering algorithm called *Dynamic Hierarchical Star* is introduced. Our approach aims to construct a hierarchy of overlapped clusters, dealing with dynamic data sets. The experimental results on several benchmark text collections show that this method obtains smaller hierarchies than traditional algorithms while achieving a similar clustering quality. Therefore, we advocate its use for tasks that require dynamic overlapped clustering, such as information organization, creation of document taxonomies and hierarchical topic detection.

Keywords: hierarchical clustering, dynamic clustering, overlapped clusters.

1 Introduction

The World Wide Web and the number of text documents managed in organizational intranets continue to grow at an amazing speed. Managing, accessing, searching and browsing large repositories of text documents require 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.

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, dynamic 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.

Hierarchical clustering algorithms have an additional interest, because they provide data-views 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.

In the context of hierarchical document clustering, six major challenges must be addressed: 1) Very high dimensionality of the data: the computational complexity should be linear with respect to the number of dimensions (terms).

2) Very large size of text collections: the algorithms must be efficient and scalable to large data sets. 3) Documents often have several topics: it is important to avoid confining each document to only one cluster. Thus, overlapping between document clusters should be allowed. 4) Dynamic data sets: the algorithms must be able to update the hierarchy when documents arrive (or are removed). 5) The insensitivity to the input order: the generated set of clusters must be unique, independently on the arrival order of the documents. It is one of the major issues in incremental and dynamic algorithms, and 6) The number of clusters is unknown prior to the clustering: it is difficult to specify a reasonable level of the hierarchy. Instead of that, it makes more sense to let the clustering algorithm find it out by itself.

Agglomerative and Divisive are two general categories of hierarchical clustering algorithms. Both of them have been applied to document clustering. UPGMA [1] of agglomerative algorithms and Bisecting K-Means (BKM) [2] of divisive methods are reported to be the most accurate one in its category [3]. These hierarchical methods neither can deal with dynamic data sets nor allow overlapping between clusters.

There are some incremental algorithms that update the cluster hierarchy when new documents arrive, such as DC-tree [4] and IHC [5]. They are based on a tree structure and obtain disjoint document hierarchies. In DC-Tree the document assignments to clusters are irrevocable, whereas IHC is relatively not sensitive to the presentation of input ordering. DC-Tree defines also several parameters, thus its tunning is problematic.

On the other hand, several static hierarchical algorithms have been proposed for overlapped clustering of documents, including HFTC [6] and HSTC [7]. HFTC algorithm attempts to address the hierarchical document clustering using the notion of frequent itemsets. Each cluster consists of a set of documents containing all terms of each frequent term set. HSTC algorithm provides the methodology for organizing the base clusters identified by STC algorithm [8] into a navigable hierarchy. A base cluster consists of a set of documents that share a common phrase. Like STC, the time complexity of HSTC is quite high with respect to the number of terms.

To the best of our knowledge, there are no hierarchical algorithms for document clustering that combine both processing of dynamic data and obtaining of overlapped clusters.

In this paper, we present a novel overlapped hierarchical algorithm, called *Dynamic Hierarchical Star* algorithm (DHS), for clustering of dynamic document collections. This approach attempts to address the challenges mentioned above. The experimental results on several benchmark text collections show that this method obtains smaller hierarchies than traditional algorithms while achieving a similar clustering quality.

The remainder of the paper is organized as follows: Section 2 describes DHS clustering algorithm. The comparison with traditional hierarchical algorithms is shown in Section 3. Finally, conclusions are presented in Section 4.

2 Dynamic Hierarchical Star Algorithm

In this paper, we introduce a new dynamic hierarchical clustering algorithm derived from the general hierarchical framework proposed in [9]. We will call it *Dynamic Hierarchical Star* algorithm. This method updates a hierarchy of overlapped clusters when new documents arrive (or are removed).

As our algorithm is derived from the framework, it is an agglomerative method based on graph. 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. The process in each layer involves two steps: construction of a graph and obtaining a cover for this graph. In this context, a cover for a graph $G = (V, E)$ is a collection $V_1, V_2, ..., V_k$ of (not necessarily disjoint) subsets of V such that $\cup_{i=1}^{k} V_i = V$, each one representing a cluster.

DHS algorithm uses two graphs. The first one is the β-similarity graph, which is an undirected graph whose vertices are the clusters and there is an edge between vertices i and j, if the cluster j is β-similar to i. Two clusters are β-similar if their similarity is greater than 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 less than β. Like UPGMA clustering method, we use group-average as inter-cluster similarity measure.

The second graph relies on the maximum β-similarity relationship (denoted as *max-S* graph) and it is a subgraph of the first one. The vertices of this graph coincide with vertices in the β-similarity graph, and there is an edge between vertices i and j, if i is the most β-similar cluster to j or vice versa.

Given a cluster hierarchy previously built by the algorithm, each time a new document arrives (or is removed), the clusters at all levels of the hierarchy must be revised (see Figure 1). When a new document arrives (or is removed), a singleton is created (or deleted) and the β-similarity graph at the bottom level is updated. Then, the *max-S* graph is updated too, which produce (or remove) a vertex and can also produce new edges and remove others. The star cover routine is applied to the *max-S* graph in order to update the clusters. When clusters are created or removed from a level of the hierarchy, the β-similarity graph at the next level must be updated. This process is repeated until this graph is completely disconnected (all vertices are β-isolated). It is possible that

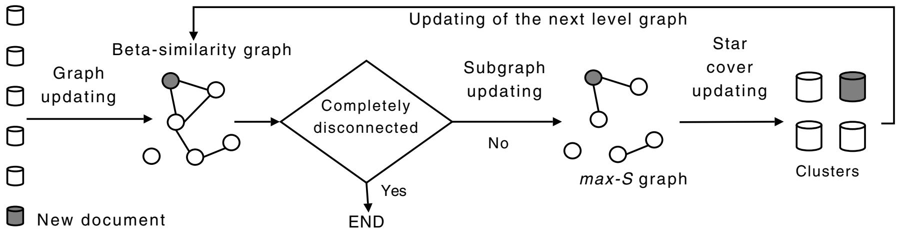

Fig. 1. Dynamic Hierarchical Star algorithm

Algorithm 1. Dynamic Hierarchical Star method

```
1. Arrival of a document to cluster (or to remove).
2. Put the new document in a cluster on its own (or remove the single cluster to
   which the document belongs).
3. $level = 0$ and update the $\beta$-similarity graph at the bottom level, $G_0$.
4. While $G_{level}$ is not completely disconnected:
   (a) Update the max-S graph at $level$.
   (b) Update the star cover for the max-S graph.
   (c) Update the $\beta$-similarity graph at the next level, $G_{level+1}$.
   (d) $level = level + 1$
5. If there exist levels greater than $level$ in the hierarchy, remove them.
```

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. Notice that the algorithm uses the same β value in all hierarchy levels.

The DHS method is summarized in Algorithm 1. As it can be noticed, the dynamic algorithm comprises the updating of both the graphs and the star cover at each level of the hierarchy. The updating of the β-similarity graph is trivial. The steps of the *max-S* graph updating are shown in Algorithm 2.

Algorithm 2. Updating of the *max-S* graph

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

2.1 Star Cover Routine

Our star cover routine approaches the minimum dominating set of the *max-S* graph using a greedy heuristic that takes into account the number of neighbors of each vertex. Each cluster is a star-shaped subgraph of $l+1$ vertices. It consists of a single star and l satellite vertices, where there exist edges between the star and each satellite vertex. Since a vertex can be neighbor of several stars, this cover routine obtains overlapped clusters.

Several heuristics for obtaining the star cover for a graph have been proposed. The star-based methods presented in [10, 11] are not able to update the cover when documents arrive or are removed, and therefore can not be used in dynamic algorithms. In contrast, Aslam [12] proposed a dynamic star cover routine but it

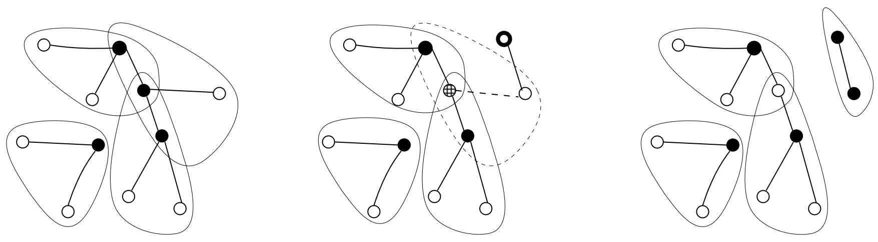
(a) Original star cover (b) New document arrival (c) Updated star cover

Fig. 2. Star cover updating (black circles represent the stars)

depends on the data order, since if two or more neighbors with the same degree exist, only the first of them in the arrangement is considered as a star.

Thus, our proposal follows the same idea of Aslam, but with some differences. The first one is that in our cover routine two stars can be neighbors, and therefore the obtained cover is independent on the data order. We considered a vertex as star if its degree is greater than or equal to that of its neighbors. The second one is that our star cover routine is carried out over the *max-S* graph instead of the β-similarity graph. This difference demands to handle the changes produced by adding and removing edges not connected with the new document.

Algorithm 3. Star cover updating

```
1. Let $N$ be the set of vertices added to the $max{-}S$ graph and $R$ be the set of vertices
   removed from it. Let also $NE$ be the set of edges added to the $max{-}S$ graph and
   $RE$ be the set of edges removed from it.
2. Let $Q$ be a queue with the vertices to be evaluated, $Q = N$.
3. Put into $Q$ all stars $s$ such that $\exists v\,(s, v) \in RE$, and mark them as satellites. Put
   also into $Q$ each vertex $v$ (if $v \notin R$) and all neighbors of $s$.
4. Put into $Q$ all stars $s$ such that $\exists v, v'\,(v, v') \in NE$ and $v$ neighbor of $s$. Mark each
   star $s$ as satellite and put also into $Q$ all neighbors of $s$.
5. While $Q \neq \emptyset$:
   (a) Extract the highest degree vertex $v$ from $Q$.
   (b) If $v$ is a satellite vertex and ($v$ does not have any star neighbors or the degree
       of $v$ is greater than or equal to that of all its star neighbors):
         i. Mark $v$ as star.
        ii. If $v$ has star neighbors with less degree, then mark these stars as satellites
            and put into $Q$ all neighbors of them.
```

The intuition behind the updating of our star cover after a new vertex is added to a graph is depicted in Figure 2. A *max-S* graph and its star cover are shown in Figure 2a. Suppose a new vertex is added to the graph, as Figure 2b. How does the addition of this new vertex affect the star cover? (see Figure 2c). In general, the answer depends on the changes in the *max-S* graph produced by new vertex addition. All stars that decrease its degree and those whose neighbors increase its degree are enqueued to be re-evaluated. Vertices with highest degree are extracted from the queue until it is empty. If the extracted vertex does not have any star neighbors or it has a degree greater than or equal to its star neighbors, the star structure already in place has to be modified to assign the vertex as a star. The most difficult case that destroys the star cover is when the evaluated vertex is adjacent to several stars, each of whose degree is less than that of this vertex. In this situation, the satellite vertices in the stars that are broken as a result have also to be re-evaluated (see Algorithm 3).

3 Experimental Results

The performance of the Dynamic Hierarchical Star algorithm has been evaluated using five benchmark text collections, whose general characteristics are summarized in Table 1. Human annotators identified the topics in each collection. Notice that these topics are overlapped. In our experiments, the documents are represented using the traditional vector space model. The terms of documents represent the lemmas of the words appearing in the texts (stop words are disregarded). We use the traditional cosine measure to compare the documents.

Table 1. Description of document collections

Collection	Source	Documents	Terms	Topics	Overlapping
AFP	TREC-5	695	12575	25	1.02
ELN	TREC-4	5829	83434	50	1.2
TDT	TDT2	9824	55112	193	1.01
REU	Reuters-21578	10369	35297	120	1.26
REU2	RCV1-v2 Train	23150	47152	101	3.18

There are several measures to evaluate the quality of hierarchical clustering. We adopt a widely used Overall F-measure [13], which compares the system-generated clusters with the manually labeled 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 comparing the quality of the clustering produced by UPGMA and BKM methods[1] against DHS algorithm.

The obtained results 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 result is chosen. We can do that, because of the well-defined stop condition of our method. On the contrary, in the other algorithms we evaluated each level of the hierarchy and considered the flat partition produced by the best one. As it can be noticed, our method not only obtains similar quality results, but it also offers a hierarchy easier to browse, since the less number of clusters and levels it has.

[1] We make use of the CLUTO-2.1 Clustering Toolkit to generate these results.

Table 2. Overall F results for document collections (x=not scalable to run)

Data	Algorithm	Levels	Clusters in hierarchy	Clusters in best level	Overall F
AFP	UPGMA	694	694	40	0.84
	BKM	694	694	10	0.75
	DHS ($\beta = 0.13$)	5	421	42	0.82
ELN	UPGMA	5828	5828	400	0.47
	BKM	5828	5828	50	0.47
	DHS ($\beta = 0.13$)	7	3965	76	0.46
TDT	UPGMA	9823	9823	100	0.75
	BKM	9823	9823	20	0.58
	DHS ($\beta = 0.16$)	8	6571	155	0.77
REU	UPGMA	10368	10368	100	0.53
	BKM	10368	10368	10	0.14
	DHS ($\beta = 0.11$)	9	7142	101	0.51
REU2	UPGMA	x	x	x	x
	BKM	23149	23149	10	0.35
	DHS ($\beta = 0.02$)	9	16156	9	0.34

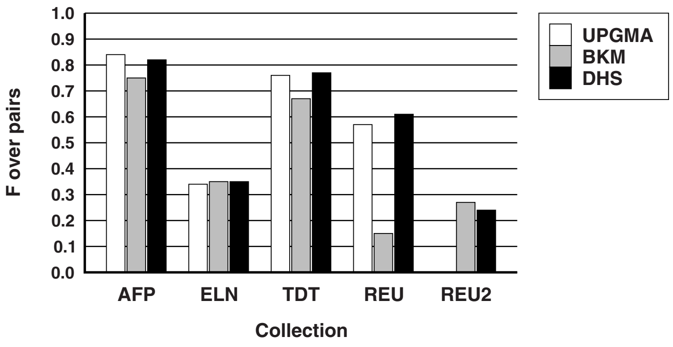

Fig. 3. F-measure over pairs

The higher the overlapping, the better Overall F-measure can be obtained since, by definition, power set achieves the best F value. For that reason, we have also considered F-measure calculated over pairs of documents [14], which is proposed to evaluate overlapped clusters. In this case, precision is calculated as the fraction of pairs correctly put in the same cluster and recall is the fraction of actual pairs that were identified. The performance of the algorithms w.r.t. F-measure over pairs is shown in Figure 3. Notice that our method obtains similar quality results than representative hierarchical algorithms again. Thus, we can conclude that our algorithm does not profit from the overlapping.

4 Conclusions

In this paper, a new clustering algorithm called Dynamic Hierarchical Star has been proposed. Its most important novelty is the capability to handle dynamic data sets and to build overlapped cluster hierarchies. Other key features of the proposed algorithm are the insensitivity to the input order, a well-defined stop condition and linear computational complexity w.r.t. the number of dimensions.

The experiments were conducted on five benchmark text collections. Results show that our method not only reaches similar quality than representative hierarchical algorithms, but it also offers a hierarchy easier to browse, since the less number of clusters and levels it has. Thus, we advocate its use for tasks that require dynamic clustering, such as creation of document taxonomies and hierarchical topic detection. Though we employ our method to cluster text data, it can be also applied to any problem of Pattern Recognition with mixed objects.

References

[1] Jain, A., Dubes, R.: Algorithms for Clustering Data. Prentice-Hall, Englewood Cliffs (1988)

[2] Steinbach, M., Karypis, G., Kumar, V.: A comparison of document clustering techniques. In: ACM SIGKDD Workshop on Text Mining, Boston, pp. 109–110 (2000)

[3] Li, Y., Chung, S.M., Holt, J.D.: Text document clustering based on frequent word meaning sequences. Data & Knowledge Engineering 64, 381–404 (2008)

[4] Wong, W., Wai-chee Fu, A.: Incremental Document Clustering for Web Page Classification. In: IEEE Int. Conf. on Information Society in the 21st Century: Emerging technologies and new challenges, Japan (2000)

[5] Widyantoro, D., Yen, J.: An incremental approach to building a cluster hierarchy. In: 2nd IEEE International Conference on Data Mining, Japan, pp. 705–708 (2002)

[6] Beil, F., Ester, M., Xu, X.: Frequent term-based text clustering. In: Knowledge Discovery and Data Mining, pp. 436–442. ACM Press, Canada (2002)

[7] Maslowska, I.: Phrase-based hierarchical clustering of web search results. In: Sebastiani, F. (ed.) ECIR 2003. LNCS, vol. 2633, pp. 555–562. Springer, Heidelberg (2003)

[8] Zamir, O., Etziony, O.: Web document clustering: A Feasibility demonstration. In: 21st SIGIR Conference, pp. 46–54. ACM Press, Melbourne (1998)

[9] Gil-García, R., Baddía-Contelles, J., Pons-Porrata, A.: Dynamic Hierarchical Compact Clustering Algorithm. In: Sanfeliu, A., Cortés, M.L. (eds.) CIARP 2005. LNCS, vol. 3773, pp. 302–310. Springer, Heidelberg (2005)

[10] Gil-García, R., Badía-Contelles, J., Pons-Porrata, A.: Extended Star Clustering Algorithm. In: Sanfeliu, A., Ruiz-Shulcloper, J. (eds.) CIARP 2003. LNCS, vol. 2905, pp. 480–487. Springer, Heidelberg (2003)

[11] Pérez-Súarez, A., Medina-Pagola, J.: A clustering algorithm based on generalized stars. In: Perner, P. (ed.) MLDM 2007. LNCS (LNAI), vol. 4571, pp. 248–262. Springer, Heidelberg (2007)

[12] Aslam, J., Pelekhov, K., Rus, D.: Static and Dynamic Information Organization with Star Clusters. In: CIKM 1998, pp. 208–217. ACM Press, Maryland (1998)

[13] Larsen, B., Aone, C.: Fast and Effective Text Mining Using Linear-time Document Clustering. In: KDD 1999, pp. 16–22. ACM Press, San Diego (1999)

[14] Banerjee, A., Krumpelman, C.: Model based overlapping clustering. In: KDD 2005, pp. 532–537. ACM Press, Chicago (2005)