

Bidirectional Hierarchical Clustering for Web Mining

ZHONGMEI YAO & BEN CHOI

Computer Science, College of Engineering and Science
Louisiana Tech University, Ruston, LA 71272, USA
zya001@latech.edu, pro@BenChoi.org

Abstract

In this paper we propose a new bidirectional hierarchical clustering system for addressing challenges of web mining. The key feature of our approach is that it aims to maximize the intra-cluster similarity in the bottom-up cluster-merging phase and it ensures to minimize the inter-cluster similarity in the top-down refinement phase. This two-pass approach achieves better clustering than existing one-pass approaches. We also propose a new cluster-merging criterion for allowing more than two clusters to be merged in each step and a new measure of similarity for taking into consideration not only the inter-connectivity between clusters but also the internal connectivity within the clusters. These result in reducing the average complexity for creating the final hierarchical structure of clusters from $O(n^2)$ to $O(n)$. The hierarchical structure represents a semantic structure between concepts of clusters and is directly applicable to the future of semantic net.

1. Introduction

The World Wide Web, with its explosive growth and ever-broadening reach, has become the default knowledge resource for many areas of endeavor. It is becoming increasingly important to devise sophisticated schemes to find interesting concepts and relations between concepts from this resource. Clustering is one of the techniques that can solve this problem. Clustering is an unsupervised discovery process for partitioning a set of data such that the intra-cluster similarity is maximized and the inter-cluster similarity is minimized [1,2]. The application of clustering techniques to web mining has been facing a number of challenges [3,4], such as huge amount of resources, retrieval time, high dimensionality, quality, and meaningful interpretation.

In this paper we propose a new Bidirectional Hierarchical Clustering system in a high dimensional space based in part on the graph partitioning model [5,6]. Our system first uses the all-k-nearest neighbors [7] to sparsify the graph and to eliminate outliers. In our bottom-up cluster-merging phase, we define a new edge matching [5,6] method that takes into consideration not only the inter-connectivity between vertices but also the internal connectivity within the vertices. This edge matching method also discovers the hierarchical structure of clusters much faster than the usual hierarchical clustering. Our top-down refinement processing then eliminates errors that occurred in the greedy cluster-merging phase. The final step is to extract concepts from the clusters organized in the hierarchical structure.

The rest of this paper is organized as follows. Section 2 reviews related work. Our proposed Bidirectional Hierarchical Clustering system is presented in Section 3. Section 4 discusses the computational complexity of our algorithm. Section 5 contains conclusions and future work.

2. Related Work

Numerous clustering algorithms appear in literature [1-4,8-19]. Clustering techniques can be broadly categorized into partitional clustering and hierarchical clustering [1,2] which differ in whether they produce flat partitions or hierarchy of clusters. The k-means is a partitional clustering algorithm which has $O(n)$ time complexity in terms of the number of data points [8,19]. While the k-means is sensitive to outliers, the medoid-based method eliminates this problem typified by PAM and CLARANS [9]. But the k-medoids have $O(n^2)$ time complexity. The limitations of these two partitional schemes are that they are sensitive to initial seeds and they fail when clusters have arbitrary shapes or large different sizes.

This research was supported in part by Center for Entrepreneurship and Information Technology (CEnIT), Louisiana Tech University, Grant iCSe 200123.

Hierarchical clustering creates a nested sequence of clusters. There are variations of hierarchical agglomerative clustering (HAC) algorithms which differ primarily in how they compute the distance between clusters [1,2]. For instance, the single link method can find clusters of arbitrary shape or different sizes, but it is susceptible to noise and outliers. The complete link method is less used because of its $O(n^3)$ time complexity. An efficient method is the group average method which defines the average pair-wise distance as the cluster distance.

Other density-based or grid-based clustering methods were presented, e.g. GDBSCAN [13] and OptiGrid [14]. Nevertheless they do not work effectively in a very high dimensional space [4,15]. Another clustering approach is the probabilistic approach [16]. This approach tends to impose structure on the data and the selected distribution family may not be appropriate [4].

More recently, clustering algorithms for mining large databases have been proposed [10-12]. Most of these are variants of hierarchical clustering, e.g. BIRCH [11], CURE [12], and CHAMELEON [10]. In summary, only k-means methods, HAC methods and graph partitioning algorithms [10] have been applied in very high dimensional datasets. The performances of HAC algorithms have higher quality and are more versatile than the k-means algorithm. The major limitations of HAC methods [8,10,17] are their $O(n^2)$ time complexity and the errors that may occur during the greedy cluster-merging procedure (Figure 1). In the following sections we present our new algorithm that overcomes the limitations of common HAC methods.

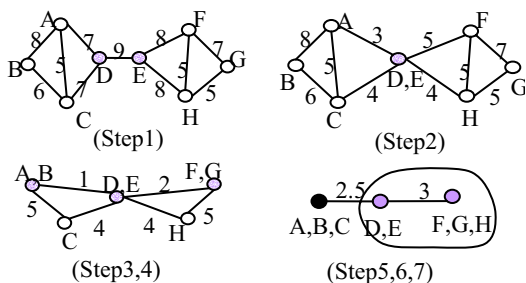


Figure 1. An example of HAC. Note that the greedy decision can lead to an incorrect solution. The correct solution in this case is (ABCD) and (EFGH).

3. Our New Bidirectional Hierarchical Clustering (BHC) System

In this section we propose our new BHC approach. This approach consists of the following five major steps: (1) representing web pages by vector-space model; (2) generating the matrix of k-nearest neighbors of web

pages; (3) bottom-up cluster merging phase; (4) top-down refinement phase; and (5) extracting concepts of clusters.

3.1. Representing Web Pages

We convert a web page into a vector of features and only text in the web page is represented: $(w_{1i}, \dots, w_{ki}, \dots, w_{mi})$ where w_{ki} is the weight of the term t_k in the i^{th} web page, and m is the number of distinct terms (dimensionality) in the dataset. Hereafter m denotes the dimensionality and n denotes the number of web pages.

A major difficulty of text clustering is the high dimensionality of the feature space [20,21]. After removing stopping-terms and stemming terms in web pages, we remove those terms whose document frequencies [21] are less than a threshold in order to reduce dimensionality. The document frequency thresholding can be reliably used, because it eliminated 90% or more unique terms with either an improvement or no loss in accuracy of performance and it also has lowest cost [21].

We then use the term frequency inverse document frequency (tf-idf) [19] to determine w_{ki} ($1 \leq k \leq m$, and $1 \leq i \leq n$):

$$w_{ki} = tf_{ki} \times \log(n / df_k) \times s_i,$$

where tf_{ki} is the frequency of the term t_k in the i^{th} web page, df_k is the document frequency of term t_k , and s_i is the normalization component. Finally, the length of each web page vector is normalized to have unit L^2 norm [19], that is,

$$s_i = \left(\sum_{k=1}^m (tf_{ki} \log(N / df_k))^2 \right)^{-1/2}.$$

The normalization ensures that web pages dealing with the same subject matter, but differing in length lead to similar web page vectors [19]. The cosine measure is then applied to compute similarity between vectors, d_i and d_j :

$$\cos(d_i, d_j) = \frac{d_i \cdot d_j}{\|d_i\| * \|d_j\|}$$

where “ \cdot ” denotes the dot-product of vectors and $\|d\|$ is the length of the vector. Since the length of the web page vector is normalized to have unit length, the above formula is simplified to $\cos(d_i, d_j) = d_i \cdot d_j$. Cosine measure has been tested to be one of the best similarity measures compared to Dice Coefficient, extended Jaccard, Euclidean, Pearson correlation measures in web page domain [4,20,22].

3.2. Generating All-k-nearest-neighbor Matrix

Finding k-nearest-neighbor for each web page can be solved by brute force using $O(n^2)$ similarity computations. Fortunately, there are fast algorithms to solve the all-k-

nearest-neighbor (Aknn) problem. We apply the fast algorithm presented in [7]. The fast Aknn algorithm starts with a rough guess of the set of k-nearest neighbors and refines it when more information is available throughout the process. A pivot-based index is used to index the set of nearest neighbors. The pivot-based indexing algorithm [7] works on the triangle inequality. However, general similarity functions do not obey the triangle inequality. Thus we have to transform the similarity s into distance $t = \sqrt{-\log(s)}$ [4] just for the Aknn problem. The algorithm

has $O(n^{\frac{2+\alpha}{1+\alpha}})$ ($1 \leq \alpha \leq 2$) time complexity. The value of α depends on how good the index is to search in the vector space.

The $n \times k$ Aknn matrix is used to construct a sparse graph, in which a vertex represents a web page and each vertex is connected with its k-nearest neighbors. Edges in the graph are weighted by the pair-wise similarities among vertices. We denote the maximum edge weight as Max , which will be used to determine thresholds in the following phase. This k-nearest-neighbor graph approach reduces redundancy, outliers and overall execution time.

3.3. The Bottom-up Cluster Merging Phase

Our bottom-up cluster merging approach operates on the idea of matching [5,6] in graph partitioning. If the edge between two vertices in the graph $G_i = (V_i, E_i)$ (V_i is the set of vertices and E_i is the set of edges) has been matched, it is collapsed and a multi-node consisting of these two vertices is created. A coarser graph G_{i+1} is obtained by collapsing the matched adjacent vertices in G_i (Figure 2 [5,6]). Each vertex in the original graph G_0 is regarded as a single cluster. A hierarchical structure of clusters is created in the graph coarsening procedure.

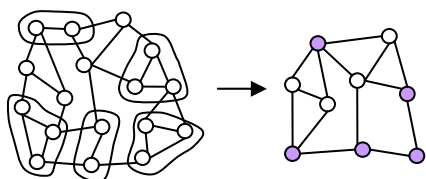


Figure 2. Matching vertices to coarsen a graph.

We define a new matching method called **Heavy Connectivity Matching (HCM)** for allowing more than two vertices to be merged in each stage. We also define a new similarity measure called **edge connectivity** for taking into consideration the inter-connectivity between vertices and the internal connectivity within the vertices.

Edge connectivity between vertices u and v is defined:

$$\frac{in_edge(u) + in_edge(v) + cr_edge(u, v)}{|u \cup v|}$$

where $in_edge(u)$ is the sum of the weights of edges connecting sub-vertices *in* vertex u if u containing more than one vertices; otherwise it is 0; $cr_edge(u, v)$ is the weight of the edge *crossing* between vertices u and v ; and $|u \cup v|$ is the number of edges in the union of u and v .

Our Heavy Connectivity Matching method proceeds by visiting vertices in an arbitrary order. If a vertex u has not been matched yet, we select its unmatched adjacent vertices such that the edge connectivity between u and its unmatched adjacent vertices is larger than a threshold δ . Vertex u and its matched adjacent vertices are then combined to form a multi-node for the next coarser graph.

In order to preserve the connectivity information in the coarser graph, we update edge weights after each stage of coarsening the graph. Let V_i^v be the set of vertices of G_i combined to form vertex v of the next coarser G_{i+1} . We compute $in_edge(v)$ to be the sum of weights of edges connecting the vertices within V_i^v . In the case where more than one vertex of V_i^v contain edges to another vertex u , the weight of the $cr_edge(v, u)$ is updated as the sum of the weights of edges connecting v and u .

HCM is applied successively to coarsen the graph. In each stage the threshold δ is divided by a decay factor, β ($\beta > 1$) [18]. δ equals Max/β for the first stage. During the i^{th} stage, the edges which have weights in the range of $(Max/\beta^i \sim Max/\beta^{i+1})$ are matched and collapsed. β controls the speed of coarsening and guarantees a certain number of edges are matched and collapsed during each stage. The hierarchical structure of clusters is thus created during this merging phase. The merging procedure stops when the highest edge connectivity in the coarsest graph is below a stopping factor that is a function of Max/β^i .

3.4. The Top-down Refinement Phase

After grouping vertices in the greedy hierarchical way, we successively refine the clusters as we project the coarser graph G_{i+1} down to the larger finer graph G_i . Obtaining the larger G_i from the coarser G_{i+1} is done simply by transforming multi-node v of G_{i+1} back to individual vertices, V_i^v , of G_i . Since G_i is finer, it provides more degrees of freedom that can be used to refine clustering.

The refinement algorithm is used to reduce the inter-connectivity between clusters (or multi-nodes). The inter-connectivity between clusters A and B is defined as:

$$gain = \frac{\sum_{i \in A, j \in B} weight(i, j)}{|A| \times |B|},$$

where vertex i belongs to cluster A (or multi-node A), vertex j belongs to cluster B and $|A|$ is the size of cluster A . If a vertex in A is swapped to cluster B and *decreases* the value of $gain$, then the vertex should be moved to

cluster B. The gain is similar to the ratio-cut heuristic in [17].

Given the definition of gain, for each vertex u , we compute improvement if u is moved from the cluster it belongs to, to one of the other clusters that u is connected to. The improvement is indicated by the heuristic value of (*gain-before-swap* – *gain-after-swap*). The Kernighan-Lin algorithm (KL) [5,6] then proceeds by repeatedly selecting a vertex u with the highest heuristic value and moving it to the desired cluster. After moving u , u won't be moved again and the heuristic values of the vertices adjacent to u are updated to reflect the change. In each finer graph, the KL algorithm is terminated when no more vertex moving will decrease the inter-connectivity between clusters. This refinement algorithm is applied at each successive finer graph.

For the example in Figure 1, we will compute the improvement if any vertex is moved to the other cluster which it is connected to. As we can see, D will be moved to the other cluster (ABC) since its value of (*gain-before-swap* – *gain-after-swap*) is $(\frac{14}{3 \times 5} - \frac{9}{4 \times 4} = 0.37)$. Further moving won't have any improvement. This illustrates our refinement method can improve clustering and obtain the correct solution.

We can see that the top-down (coarsest-finest) refinement approach operates at different representation scales and can easily identify groups of vertices to be moved together. Thus this multi-level refinement approach can climb out of local minima very effectively [5,6].

3.5. Concept Extraction

We extract cluster concept by selecting the most important terms from each cluster. We apply the most frequent and predictive term method to extract the concepts of clusters, since it received the best performance over χ^2 method, most frequent term method, and most predictive term method [24]. The most frequent and predictive word method selects terms based on the product of local frequency and predictiveness:

$$p(\text{term} | \text{cluster}) \times \frac{p(\text{term} | \text{cluster})}{p(\text{term})}$$

where $p(\text{term}|\text{cluster})$ is frequency of the term in the cluster and $p(\text{term})$ is the term's frequency in the whole collection. The k-highest-ranking terms are thus extracted from the cluster to represent the concept.

4. Analysis of Computational Complexity

The overall computational complexity of our new algorithm depends on the time complexity of building the all-k-nearest-neighbor matrix and the amount of time it

requires to perform the bottom-up and top-down phases of the clustering algorithm. The time complexity of finding the Aknn has been discussed in the previous section, which is $O(n^{\frac{2+\alpha}{1+\alpha}})$ ($1 \leq \alpha \leq n$).

The amount of time required by the merging phrase depends on rate in which the size of successively coarser graphs is decreasing. If the size of successively coarse graphs decreases by a constant factor, then the complexity of the algorithm is linear on the number of vertices and the number of edges in the graph [5,6].

In our new edge matching approach, since the *Max* and the decay factor β control the speed of coarsening the graph, an appropriate value of β may guarantee a number of edges are matched and collapsed during each stage. In this case, the bottom-up cluster merging phase has $O(n)$ time complexity, because in an Aknn sparse graph the number of edges is linear on the number of vertices. In the worst case, when the size of successively coarser graphs decreases by only a few vertices at a time, the complexity of the merging algorithm will be quadratic on the number of vertices in the graph. The complexity of refinement phase is same as the merging phase since both of them are multilevel algorithms. (The KL improved by Fiduccia and Mattheyses [23] reduces complexity to $O(|E|)$ by using appropriate data structures.) Therefore the average complexity of overall procedure is determined by constructing the Aknn graph, which takes $O(n^{\frac{2+\alpha}{1+\alpha}})$ ($1 \leq \alpha \leq n$) time.

5. Conclusions and Future Work

In this paper we presented the comprehensive process for clustering web pages and extracting cluster concepts. More importantly, we proposed a new BHC algorithm based in part on multilevel graph partitioning. We defined a new edge matching method that preferred merging the sub-clusters whose edge connectivity was high in the bottom-up cluster-merging phase. We also used an objective function for the top-down refinement procedure that decreased the inter-connectivity between different clusters. Thus the new algorithm tried to maximize the intra-cluster similarity in the bottom-up cluster-merging phase and it ensured to minimize the inter-cluster similarity in the top-down refinement phase. The advantages of our algorithm are that it eliminated the errors occurring in greedy clustering algorithms and its multilevel refinement procedure was very effective in climbing out of local minima. The average time complexity of our new algorithm is $O(n^{\frac{2+\alpha}{1+\alpha}})$ ($1 \leq \alpha \leq n$), which is also faster than the common HAC algorithm ($O(n^2)$).

We believe that the new algorithm will have good performance in near future, since the Aknn algorithm, the multilevel graph partitioning and KL algorithm were well studied and implemented. However, as we can see, the choice of proper objective functions is essential for overall success of our algorithm. Another problem is the method we used to extract concepts of clusters. Using important terms to represent concepts of clusters is the simplest way but more sophisticated methods remain to be developed.

Our future work includes investigating more sophisticated methods for clustering based on contextual meaning of web pages and incorporating them with our proposed classification system [25,26] into our web-page Classification and Search Engine.

References

- [1] B. S. Everitt, S. Landua, and M. Leese, *Cluster Analysis*, Arnold, London Great Britain, 2001.
- [2] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data Clustering: A Review", *ACM computing Surveys*, Vol. 31, No. 3, September 1999, pp.255-323.
- [3] O. Zamir and O. Etzioni, "Web Document Clustering: A Feasibility Demonstration", in *Proc. 21st Annu. Int. ACM SIGIR Conf.*, 1998, pp.46-54.
- [4] A. Strehl, "Relationship-based Clustering and Cluster Ensembles for High-dimensional Data Mining", Dissertation, The University of Texas at Austin, May 2002.
- [5] G. Karypis, and V. Kumar, "Multilevel k -way Partitioning Scheme for Irregular Graph", *Journal of Parallel and Distributed computing*, 48(1), 1998, pp96-129.
- [6] G. Karypis and V. Kumar, "A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs", *SIAM Journal of Scientific Computing*, 20(1), 1999, pp359-392.
- [7] E. Chavez, K. Figueroa, and G. Navarro, "A Fast Algorithm for the All K Nearest Neighbors Problem in General Metric Spaces", <http://garota.fisimat.umich.mx/~elchavez/publica/>.
- [8] M. Steinbach, G. Karypis, V. Kumar, "A Comparison of Document Clustering Techniques", *KDD'2000*, Technical report of University of Minnesota.
- [9] R. Ng and J. Han, "Efficient and Effective Clustering Methods for Spatial Data Mining", *VLDB-94*.
- [10] G. Karypis, E.-H. Han, V. Kumar, "CHAMELEON: A Hierarchical Clustering Algorithm Using Dynamic Modeling", *IEEE Computer*, 32(8), August 1999, pp.68-75.
- [11] T. Zhang, R. Ramakrishnan and M. Linvy, "BIRCH: an Efficient Data Clustering Method for Very Large Databases", *Proceedings of the ACM SIGMOD Conference on Management of Data*, Montreal, Canada, 1996, pp. 103-114.
- [12] S. Guha, R. Rastogi, and K. Shim, "CURE: A Clustering Algorithm for Large Databases", *Proceedings of the ACM SIGMOD Conference on Management of Data*, 1998, pp73-84.
- [13] J. Sander, M. Ester, H. P. Kriegel and X. Xu, "Density-based Clustering in Spatial Databases: The Algorithm GDBSCAN and its Applications", *An International Journal* 2(2), Kluwer Academic Publishers, Norwell, MA., June 1998, pp.169-194.
- [14] A. Hinneburg and D. A. Keim. "An Optimal Grid-clustering: Towards Breaking the Curse of Dimensionality in High-dimensional Clustering", *VLDB-99*, 1999.
- [15] B. Liu, Y. Xia, P. S. Yu, "Clustering Through Decision Tree Construction", *SIGMOD 2000*.
- [16] M. Goldszmidt and M. Sahami, "A Probabilistic Approach to Full-Text Document Clustering", Technical Report ITAD-433-MS-98-044, SRI International, <http://citeseer.nj.nec.com/goldszmidt98probabilistic.html>.
- [17] G. Karypis, E.-H. Han, and V. Kumar, "Multilevel Refinement for Hierarchical Clustering", <http://garota.fisimat.umich.mx/~elchavez/publica/>.
- [18] K. Rajaraman and H. Pan, "Document Clustering using 3-tuples", *PRICAI'2000 International Workshop on Text and Web Mining*, Melbourne, Australia, Sep. 2000, p88-95.
- [19] I. S. Dhillon, J. Fan and Y. Guan, "Efficient Clustering of Very Large Document Collections", *Data Mining for Scientific and Engineering Applications*, Kluwer Academic Publisher, 2001.
- [20] C. J. Rijsbergen, *Information Retrieval*, Butterworths, 1979.
- [21] Y. Yang and J. O. Pedersen, "A Comparative Study on Feature Selection in Text Categorization", www.cs.cmu.edu/~yiming/papers.yy/ml97.ps.
- [22] A. Strehl, J. Ghosh, and R. Mooney, "Impact of Similarity Measures on Web-page Clustering", *Proceedings of the AAAI2002 Workshop on Artificial Intelligence for Web Search*, AAAI/MIT Press, Austin, Texas, July 2002, pp58-64.
- [23] C. M. Fiduccia and R. M. Mattheyses, "A Linear Time Heuristic for Improving Network Partitions", *Proceedings 19th IEEE Design Automation Conference*, 1982, pages 175-181.
- [24] A. Popescul and L. H. Ungar, "Automatic Labeling of Document Clusters", <http://www.cis.upenn.edu/~popescul/publications.html>.
- [25] X. Peng & B. Choi, "Automatic Web Page Classification in a Dynamic and Hierarchical Way," *IEEE International Conference on Data Mining*, 2002, pp. 386-393.
- [26] B. Choi, "Making Sense of Search Results by Automatic Web-page Classifications," *WebNet 2001*, 2001, pp.184-186.