Hierarchical Document Clustering

Benjamin C. M. Fung, Ke Wang, and Martin Ester, Simon Fraser University, Canada

INTRODUCTION

Document clustering is an automatic grouping of text documents into clusters so that documents within a cluster have high similarity in comparison to one another, but are dissimilar to documents in other clusters. Unlike document classification (Wang, Zhou, and He, 2001), no labeled documents are provided in clustering; hence, clustering is also known as unsupervised learning. Hierarchical document clustering organizes clusters into a tree or a hierarchy that facilitates browsing. The parent-child relationship among the nodes in the tree can be viewed as a topic-subtopic relationship in a subject hierarchy such as the Yahoo! directory.

This chapter discusses several special challenges in hierarchical document clustering: high dimensionality, high volume of data, ease of browsing, and meaningful cluster labels. State-of-the-art document clustering algorithms are reviewed: the partitioning method (Steinbach, Karypis, and Kumar, 2000), agglomerative and divisive hierarchical clustering (Kaufman and Rousseeuw, 1990), and frequent itemset-based hierarchical clustering (Fung, Wang, and Ester, 2003). The last one, which was recently developed by the authors, is further elaborated since it has been specially designed to address the hierarchical document clustering problem.

BACKGROUND

Document clustering is widely applicable in areas such as search engines, web mining, information retrieval, and topological analysis. Most document clustering methods perform several preprocessing steps including stop words removal and stemming on the document set. Each document is represented by a vector of frequencies of remaining terms within the document. Some document clustering algorithms employ an extra preprocessing step that divides the actual term frequency by the overall frequency of the term in the entire document set. The idea is that if a term is too common across different documents, it has little discriminating power (Rijsbergen, 1979). Although many clustering algorithms have been proposed in the literature, most of them do not satisfy the special requirements for clustering documents:

- *High dimensionality.* The number of relevant terms in a document set is typically in the order of thousands, if not tens of thousands. Each of these terms constitutes a dimension in a document vector. Natural clusters usually do not exist in the full dimensional space, but in the subspace formed by a set of correlated dimensions. Locating clusters in subspaces can be challenging.
- *Scalability.* Real world data sets may contain hundreds of thousands of documents. Many clustering algorithms work fine on small data sets, but fail to handle large data sets efficiently.
- *Accuracy.* A good clustering solution should have high intra-cluster similarity and low inter-cluster similarity, i.e., documents within the same cluster should be similar but are dissimilar to documents in other clusters. An external evaluation method, the F-measure (Rijsbergen, 1979), is commonly used for examining the accuracy of a clustering algorithm.

- *Easy to browse with meaningful cluster description.* The resulting topic hierarchy should provide a sensible structure, together with meaningful cluster descriptions, to support interactive browsing.
- *Prior domain knowledge.* Many clustering algorithms require the user to specify some input parameters, e.g., the number of clusters. However, the user often does not have such prior domain knowledge. Clustering accuracy may degrade drastically if an algorithm is too sensitive to these input parameters.

Document Clustering Methods

Hierarchical Clustering Methods

One popular approach in document clustering is agglomerative hierarchical clustering (Kaufman and Rousseeuw, 1990). Algorithms in this family build the hierarchy bottom-up by iteratively computing the similarity between all pairs of clusters and then merging the most similar pair. Different variations may employ different similarity measuring schemes (Zhao and Karypis, 2001; Karypis, 2003). Steinbach (2000) shows that Unweighted Pair Group Method with Arithmatic Mean (UPGMA) (Kaufman and Rousseeuw, 1990) is the most accurate one in its category. The hierarchy can also be built top-down which is known as the divisive approach. It starts with all the data objects in the same cluster and iteratively splits a cluster into smaller clusters until a certain termination condition is fulfilled.

Methods in this category usually suffer from their inability to perform adjustment once a merge or split has been performed. This inflexibility often lowers the clustering accuracy. Furthermore, due to the complexity of computing the similarity between every pair of clusters, UPGMA is not scalable for handling large data sets in document clustering as experimentally demonstrated in (Fung, Wang, Ester, 2003).

Partitioning Clustering Methods

K-means and its variants (Larsen and Aone, 1999; Kaufman and Rousseeuw, 1990; Cutting, Karger, Pedersen, and Tukey, 1992) represent the category of partitioning clustering algorithms that create a flat, non-hierarchical clustering consisting of k clusters. The k-means algorithm iteratively refines a randomly chosen set of k initial centroids, minimizing the average distance (i.e., maximizing the similarity) of documents to their closest (most similar) centroid. The bisecting k-means algorithm first selects a cluster to split, and then employs basic k-means to create two sub-clusters, repeating these two steps until the desired number k of clusters is reached. Steinbach (2000) shows that the bisecting k-means algorithm outperforms basic k-means as well as agglomerative hierarchical clustering in terms of accuracy and efficiency (Zhao and Karypis, 2002).

Both the basic and the bisecting k-means algorithms are relatively efficient and scalable, and their complexity is linear to the number of documents. As they are easy to implement, they are widely used in different clustering applications. A major disadvantage of k-means, however, is that an incorrect estimation of the input parameter, the number of clusters, may lead to poor clustering accuracy. Also, the k-means algorithm is not suitable for discovering clusters of largely varying sizes, a common scenario in document clustering. Furthermore, it is sensitive to noise that may have a significant influence on the cluster centroid, which in turn lowers the clustering accuracy. The k-medoids algorithm (Kaufman and Rousseeuw, 1990; Krishnapuram, Joshi, and Yi, 1999) was proposed to address the noise problem, but this algorithm is computationally much more expensive and does not scale well to large document sets.

Frequent Itemset-based Methods

Wang et al. (1999) introduced a new criterion for clustering transactions using frequent itemsets. The intuition of this criterion is that many frequent items should be shared within a cluster while different clusters should have more or less different frequent items. By treating a document as a transaction and a term as an item, this method can be applied to document clustering; however, the method does not create a hierarchy of clusters.

The Hierarchical Frequent Term-based Clustering (HFTC) method proposed by (Beil, Ester, and Xu, 2002) attempts to address the special requirements in document clustering using the notion of frequent itemsets. HFTC greedily selects the next frequent itemset, which represents the next cluster, minimizing the overlap of clusters in terms of shared documents. The clustering result depends on the order of selected itemsets, which in turn depends on the greedy heuristic used. Although HFTC is comparable to bisecting k-means in terms of clustering accuracy, experiments show that HFTC is not scalable (Fung, Wang, Ester, 2003).

A Scalable Algorithm for Hierarchical Document Clustering: FIHC

A scalable document clustering algorithm, Frequent Itemset-based Hierarchical Clustering (FIHC) (Fung, Wang, and Ester, 2003), is discussed in greater detail because this method satisfies all of the requirements of document clustering mentioned above. We use "item" and "term" as synonyms below. In classical hierarchical and partitioning methods, the pairwise similarity between documents plays a central role in constructing a cluster; hence, those methods are "document-centered". FIHC is "cluster-centered" in that it measures the cohesiveness of a cluster directly using frequent itemsets: documents in the same cluster are expected to share more common itemsets than those in different clusters.

A frequent itemset is a set of terms that occur together in some minimum fraction of documents. To illustrate the usefulness of this notion for the task of clustering, let us consider two frequent items, "windows" and "apple". Documents that contain the word "windows" may relate to renovation. Documents that contain the word "apple" may relate to fruits. However, if both words occur together in many documents, then another topic that talks about operating systems should be identified. By precisely discovering these hidden topics as the first step and then clustering documents based on them, the quality of the clustering solution can be improved. This approach is very different from HFTC where the clustering solution greatly depends on the order of selected itemsets. Instead, FIHC assigns documents to the best cluster from among all available clusters (frequent itemsets). The intuition of the clustering criterion is that there are some frequent itemsets for each cluster in the document set, but different clusters share few frequent itemsets. FIHC uses frequent itemsets to construct clusters and to organize clusters into a topic hierarchy.

The following definitions are introduced in (Fung, Wang, Ester, 2003): A *global frequent itemset* is a set of items that appear together in more than a minimum fraction of the whole document set. A *global frequent item* refers to an item that belongs to some global frequent itemset. A global frequent itemset containing *k* items is called a *global frequent k-itemset*. A global frequent item is *cluster frequent* in a cluster C_i if the item is contained in some minimum fraction of documents in C_i. FIHC uses only the global frequent items in document vectors; thus, the dimensionality is significantly reduced.

The FIHC algorithm can be summarized in three phases: First, construct initial clusters. Second, build a cluster (topic) tree. Finally, prune the cluster tree in case there are too many clusters.

1. *Constructing clusters*: For each global frequent itemset, an initial cluster is constructed to include all the documents containing this itemset. Initial clusters are overlapping because one document may contain multiple global frequent itemsets. FIHC utilities this global frequent itemset as the cluster label to identify the cluster. For each document, the "best" initial cluster is identified and the document is assigned only to the best matching initial cluster. The goodness of a cluster C_i for a document doc_j is measured by some score function using cluster frequent items of initial clusters. After this step, each document belongs to exactly one cluster. The set of clusters can be viewed as a set of topics in the document set.

   Example: Figure 1 depicts a set of initial clusters. Each of them is labeled with a global frequent itemset. A document Doc_1 containing global frequent items "Sports", "Tennis", and "Ball" is assigned to clusters {Sports}, {Sports, Ball}, {Sports, Tennis} and {Sports, Tennis, Ball}. Suppose {Sports, Tennis, Ball} is the "best" cluster for Doc_1 measured by some score function. Doc_1 is then removed from {Sports}, {Sports, Ball}, and {Sports, Tennis}.

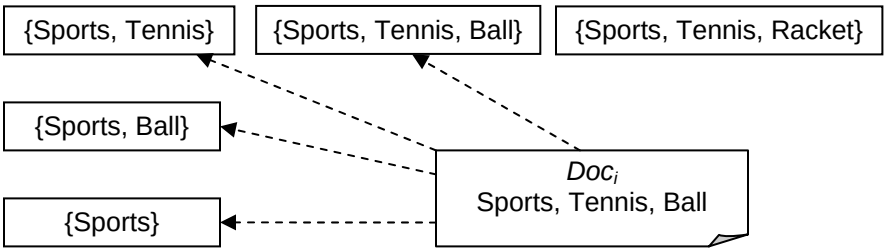

Figure 1 Initial Clusters

2. *Building cluster tree*: In the cluster tree, each cluster (except the root node) has exactly one parent. The topic of a parent cluster is more general than the topic of a child cluster and they are "similar" to a certain degree (see Figure 2 for an example). Each cluster uses a global frequent *k*-itemset as its cluster label. A cluster with a *k*-itemset cluster label appears at level *k* in the tree. The cluster tree is built bottom up by choosing the "best" parent at level *k-1* for each cluster at level *k*. The parent's cluster label must be a subset of the child's cluster label. By treating all documents in the child cluster as a single document, the criterion for selecting the best parent is similar to the one for choosing the best cluster for a document.

   Example: Cluster {Sports, Tennis, Ball} has a global frequent 3-itemset label. Its potential parents are {Sports, Ball} and {Sports, Tennis}. Suppose {Sports, Tennis} has a higher score. It becomes the parent cluster of {Sports, Tennis, Ball}.

3. *Pruning cluster tree*: The cluster tree can be broad and deep, which becomes not suitable for browsing. The goal of tree pruning is to efficiently remove the overly specific clusters based on the notion of inter-cluster similarity. The idea is that if two sibling clusters are very similar, they should be merged into one cluster. If a child cluster is very similar to its parent (high inter-cluster similarity), then replace the child cluster with its parent cluster. The parent cluster will then also include all documents of the child cluster.

Example: Suppose the cluster {Sports, Tennis, Ball} is very similar to its parent {Sports, Tennis} in Figure 2. {Sports, Tennis, Ball} is pruned and its documents, e.g., Doc_1, are moved up into cluster {Sports, Tennis}.

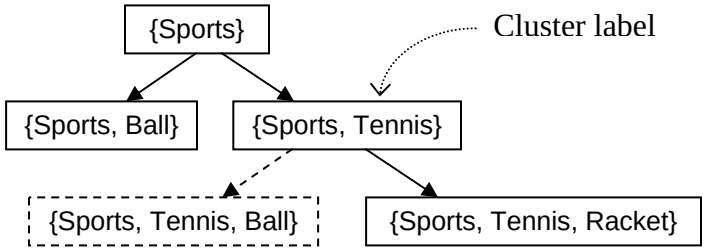

Figure 2 Sample Cluster Tree

Evaluation of FIHC

The FIHC algorithm was experimentally evaluated and compared to state-of-the-art document clustering methods. See (Fung, Wang, Ester, 2003) for more details. FIHC uses only the global frequent items in document vectors, drastically reducing the dimensionality of the document set. Experiments show that clustering with reduced dimensionality is significantly more efficient and scalable. FIHC can cluster 100K documents within several minutes while HFTC and UPGMA cannot even produce a clustering solution. FIHC is not only scalable, but also accurate. The clustering accuracy of FIHC consistently outperforms other methods. FIHC allows the user to specify an optional parameter, the desired number of clusters in the solution. However, close-to-optimal accuracy can still be achieved even if the user does not specify this parameter.

The cluster tree provides a logical organization of clusters which facilitates browsing documents. Each cluster is attached with a cluster label that summarizes the documents in the cluster. Different from other clustering methods, no separate post-processing is required for generating these meaningful cluster descriptions.

Related Links

The followings are some clustering tools on the Internet:
Tools: FIHC implements Frequent Itemset-based Hierarchical Clustering.
Website: http://www.cs.sfu.ca/~ddm/

Tools: CLUTO implements Basic/Bisecting K-means and Agglomerative methods.
Website: http://www-users.cs.umn.edu/~karypis/cluto/

Tools: Vivísimo® is a clustering search engine.
Website: http://vivisimo.com/

FUTURE TRENDS

Incrementally Updating the Cluster Tree

One potential research direction is to incrementally update the cluster tree (Guha, Mishra, Motwani and O'Callaghan, 2000). In many cases, the number of documents is growing continuously in a document set, and it is infeasible to rebuild the cluster tree upon every arrival of a new document. Using FIHC, one can simply assign the new document to the most similar

existing cluster. But the clustering accuracy may degrade in the course of the time, since the original global frequent itemsets may no longer reflect the current state of the overall document set. Incremental clustering is closely related to some of the recent research on data mining in stream data (Ordonez, 2003).

CONCLUSION

Most traditional clustering methods do not completely satisfy special requirements for hierarchical document clustering, such as high dimensionality, high volume, and ease of browsing. In this chapter, we review several document clustering methods in the context of these requirements, and a new document clustering method, FIHC, is discussed a bit more detail. Due to massive volumes of unstructured data generated in the globally networked environment, the importance of document clustering will continue to grow.

REFERENCES

Beil, F., Ester, M., & Xu, X. (2002). Frequent term-based text clustering. *International Conference on Knowledge Discovery and Data Mining*, KDD'02, Edmonton, Alberta, Canada, 436-442.

Cutting, D. R., Karger, D. R., Pedersen, J. O., & Tukey, J. W. (1992). Scatter/gather: A cluster-based approach to browsing large document collections. *International Conference on Research and Development in Information Retrieval*, SIGIR'92, Copenhagen, Denmark, 318–329.

Fung, B., Wang, K., & Ester, M. (2003, May). Hierarchical document clustering using frequent itemsets. *SIAM International Conference on Data Mining*, SDM'03, San Francisco, CA, United States, 59-70.

Guha, S., Mishra, N., Motwani, R., & O'Callaghan, L. (2000). Clustering data streams. *Symposium on Foundations of Computing Science*, 359-366.

Kaufman, L., & Rousseeuw, P. J. (1990, March). *Finding Groups in Data: An Introduction to Cluster Analysis*. New York: John Wiley & Sons, Inc.

Karypis, G. (2003). Cluto 2.1.1: A Software Package for Clustering High Dimensional Datasets. The Internet <http://www-users.cs.umn.edu/~karypis/cluto/>

Krishnapuram, R., Joshi, A., & Yi, L. (1999, August). A fuzzy relative of the k-medoids algorithm with application to document and snippet clustering. *IEEE International Conference - Fuzzy Systems*, FUZZIEEE 99, Korea.

Larsen, B., & Aone, C. (1999). Fast and effective text mining using linear-time document clustering. *International Conference on Knowledge Discovery and Data Mining*, KDD'99, San Diego, California, United States, 16–22.

Ordonez, C. (2003). Clustering binary data streams with K-means. *Workshop on Research issues in data mining and knowledge discovery*, SIGMOD'03, San Diego, California, United States, 12-19.

van Rijsbergen, C. J. (1979). *Information Retrieval*. London: Butterworth Ltd., second edition.

Steinbach, M., Karypis, G., & Kumar, V. (2000). A comparison of document clustering techniques. *Workshop on Text Mining*, SIGKDD'00.

Wang, K., Xu, C., & Liu, B. (1999). Clustering transactions using large items. *International Conference on Information and Knowledge Management*, CIKM'99, Kansas City, Missouri, United States, 483–490.

Wang, K., Zhou, S., & He Y. (2001, Apr.). Hierarchical classification of real life documents. *SIAM International Conference on Data Mining*, SDM'01, Chicago, United States.

Zhao, Y., & Karypis, G. (2001). Criterion functions for document clustering: Experiments and analysis. Technical report, Department of Computer Science, University of Minnesota.

Zhao, Y., & Karypis, G. (2002, Nov.). Evaluation of hierarchical clustering algorithms for document datasets. *International Conference on Information and Knowledge Management*, McLean, Virginia, United States, 515-524.

TERMS AND THEIR DEFINITION

Cluster Frequent Item: A global frequent item is cluster frequent in a cluster C_i if the item is contained in some minimum fraction of documents in C_i.

Document Clustering: The automatic organization of documents into clusters or group so that documents within a cluster have high similarity in comparison to one another, but are very dissimilar to documents in other clusters.

Document Vector: Each document is represented by a vector of frequencies of remaining items after preprocessing within the document.

Global Frequent Itemset: A set of words that occur together in some minimum fraction of the whole document set.

Inter-cluster Similarity: The overall similarity among documents from two different clusters.

Intra-cluster Similarity: The overall similarity among documents within a cluster.

Medoid: The most centrally located object in a cluster.

Stop Words Removal: A preprocessing step for text mining. Stop words, like "the" and "this" which rarely help the mining process, are removed from input data.

Stemming: For text mining purposes, morphological variants of words that have the same or similar semantic interpretations can be considered as equivalent. For example, the words "computation" and "compute" can be stemmed into "comput".