Incremental Construction of Topic Hierarchies using Hierarchical Term Clustering*

Ricardo M. Marcacini and Solange O. Rezende
Mathematical and Computer Sciences Institute - ICMC
University of São Paulo - USP - São Carlos, SP, Brazil
{rmm,solange}@icmc.usp.br

Abstract

Topic hierarchies are very useful for managing, searching and browsing large repositories of text documents. The hierarchical clustering methods are used to support the construction of topic hierarchies in a unsupervised way. However, the traditional methods are ineffective in scenarios with growing text collections. In this paper, an incremental method for the construction of topic hierarchies are presented, allowing the update of a topic hierarchy without repeating the clustering process. The experimental results on several benchmark text collections show that our method obtains topic hierarchies with quality similar to traditional non-incremental algorithms.

1. Introduction

The online platforms for publishing digital content has contributed significantly to the growth of several text repositories. Due to the need to extract useful knowledge from these repositories, methods for automatic organization of text collections have received great attention in the literature [7, 5, 9]. The use of topic hierarchies, such as the Yahoo Directory! and Dmoz, is one of the most popular approaches for this organization because they allow users to explore the collection interactively by topics that indicate the contents of the documents available.

The construction of topic hierarchies using supervised methods usually requires an intense human effort to bulding a model. Moreover, those methods become impracticable for growing text collections. Thus, in these situations, one possible solution is to use unsupervised hierarchical clustering methods, which allow the construction of topic hierarchies in a automatic way.

Hierarchical clustering methods organize a collection text on a hierarchy of cluster and subclusters. The most generic knowledge is represented by clusters of higher hierarchy levels while their details, or more specific knowledge by clusters of lower levels. Thus, users can explore the text collection in various levels of granularity, finding the desired information more easily and quickly [4].

Unfortunately, most existing methods for text clustering have limitations that affect the construction and maintenance of topic hierarchies [5]. A serious limitation is the fact that the clustering is done statically, i.e., it is necessary that all text documents are available before starting the clustering process. Thus, the construction of a topic hierarchy should be repeated whenever there are changes in the text collection, which greatly increases the computational cost and can make the solution unfeasible in large volumes of data. Another limitation of traditional methods for text clustering is the difficulty in interpreting the results because there are no labels associated with the clusters. In general, it is necessary to apply a specific algorithm for cluster labeling after the hierarchy construction, further increasing the computational cost of the process.

In view of this, this paper presents a incremental method for the construction of topic hierarchies in text collections. Thus, it is possible to update a topic hierarchy in a growing text collection without repeating the clustering process. The method is based on hierarchical clustering of terms (words) that finds similar terms using the document distribution in which they occur. The clusters of similar terms are used as labels for the set of retrieved documents, helping the interpretation of results. The experimental evaluations demonstrate that the proposed method presents good results and provides a competitive alternative for building dynamic topic hierarchies with understandable description of the document clusters.

To describe the proposed method, this paper is organized as follows. In Section 2, a brief review of concepts about text clustering and the well-know algorithms in the literature is provided. In Section 3, the incremental method for hierarchical term proposed in this work is presented.

*This work was sponsored by FAPESP and CNPq.

An experimental evaluation using several benchmark text collections is carried out and the results are discussed in Section 4. Finally, in Section 5, an overview of the work is presented along with a discussion of some directions for future work.

2. Background and Related Works

In this work, we consider the automatic construction of topic hierarquies using non-supervised hierarchical text clustering.

To perform a process of text clustering it is necessary to define a representation model of textual data, a similarity measure among the documents and a strategy for the cluster formation [4]. The space-vector model [10] is the most used for the representation of texts. In this model, each text document d is represented by a vector of terms, $d = (t_1, ..., t_m)$, where each term t_i has an associated value, for example, the frequency of occurrence. In order to obtain the similarity between two documents the cosine measure is commonly used. Given vectors of two documents d_i and d_j, the cosine measure is defined as follows:

$$cos(d_i, d_j) = \frac{d_i \cdot d_j}{\|d_i\|\|d_j\|} \quad (1)$$

Finally, we define a clustering strategy. The agglomerative hierarchical clustering methods are widely used, and start by considering each document as a separate cluster. Then, the pairs of the most similar documents are merged iteratively until all the documents belong to only one cluster. There are three classic algorithms of this strategy: Single-Link, Complete-Link and UPGMA [2].

The Single-Link is one of the simplest hierarchical clustering algorithms and uses the technique of the nearest neighbor, where the similarity of two clusters is the similarity of their most similar members. Unlike the Single-Link, in the Complete-link clustering the similarity of two clusters is the similarity of their most dissimilar members. In the UPGMA algorithm, the similarity between two cluster is defined as the average of the similarities between all pairs of documents in each cluster. The UPGMA eliminates many problems related to dependence on the size of the cluster, and is considered one of the best algorithms in textual collections in terms of clusters quality [12]. The clusters obtained by these three algorithms do not have labels to describe their content, doing it hard to interpret the results.

In recent years, alternative methods have been proposed for text clustering in order to obtain clusters with associated labels [3, 5, 7]. These methods aim the clustering of the most similar terms (words) in the collection, ie, terms with some value of co-occurrence. Each cluster of terms has a set of documents associated, resulting in document clusters with labels. The FIHC [5] is one of such method, and uses the Apriori algorithm [1] to find the terms with co-occurrence in the text collection, called frequent itemsets. The text collection is organized hierarchically and the frequent itemsets are used as cluster labels. One disadvantage of the FIHC algorithm is that the parameters used to find the frequent itemsets are difficult to define in practice [2].

The method proposed in this paper also performs the document clustering using the co-occurrence between terms. A technique called co-occurrence graphs [11] is used, where terms are organized in a graph and two terms are connected if there is a significant co-occurrence value between them. From the graph, the similarity between two terms, t_i and t_j, is calculated by the SNN (Shared Nearest Neighbor) measure:

$$SNN(t_i,t_j) = \frac{|\ N(t_i) \cap N(t_j)\ |}{|\ N(t_i) \cup N(t_j)\ |} \quad (2)$$

where $N(t)$ represents the set of neighbors of t. This similarity measure uses only the topology of the graph, and has been considered very useful when trying to identify topics in a given corpus. In our method, the Incremental Hierarchical Term Clustering (IHTC) algorithm is presented, which constructs the co-occurrence graph and clustering similar terms in an incremental way. The hierarchical term clustering is then used for the construction and management of topic hierarchies.

One focus of this paper is to evaluate the topic hierarchies generated by the incremental algorithm proposed and to compare the results with the traditional non-incremental methods.

3. Incremental Hierarchical Term Clustering

This section presents an incremental method for constructing topic hierarchies based on a hierarchical term clustering. The text collection is represented in a graph, where the vertices are terms from the text collection and the edges indicate significant co-occurrence between them. The most similar terms are clustered using the *SNN* measure, resulting in a hierarchical structure known as dendrogram. To illustrate, consider the co-occurrence graph in Figure 1. A possible result for the hierarchical term clustering is the dendrogram in the Figure 2.

The dendrogram is a binary tree that represents the sequence of the clustering and the similarity between the terms. Each node of the dendrogram is a possible topic of the collection, represented by a set of terms. The set of documents of a topic is obtained by the union of subsets of documents retrieved by each term in the topic. This structure allows a user to explore the textual collection by browsing the topics that describe the information of interest.

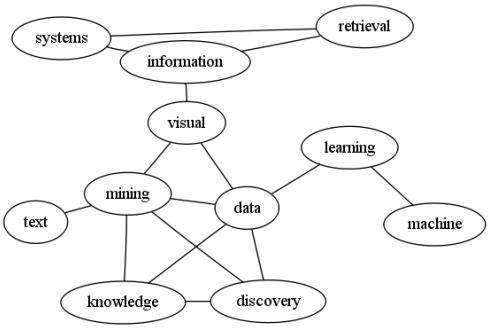

Figure 1. Co-occurrence term graph

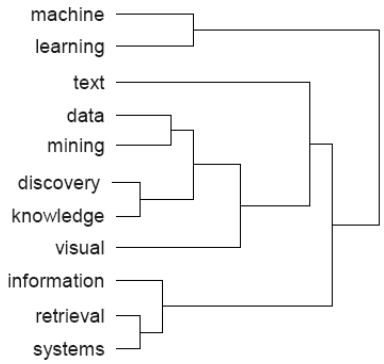

Figure 2. Dendrogram representation of the hierarchical term clustering

Below, we present the algorithm IHTC, which implements the proposed method. The algorithm performs texts preprocessing, constructs a graph of co-occurrence of terms and their related dendrogram. This whole process is performed incrementally.

3.1. Incremental Text Preprocessing and Clustering

Textual collections can easily contain thousands of terms, many of them redundant and unnecessary, that slow the clustering process and decrease the quality of results. In our method, the goals of the text preprocessing is to maintain a subset of representative terms and organize them in a co-occurrence graph.

Each document inserted in the text collection is pre-processed individually. Thus, for a new document *d*, the stopwords are removed and a stemming technique applied to transform the terms in its primitive form. Finally, the most significant terms of the document using a technique of extracting keywords are selected [3]. In this work, we used a simple technique where the most frequent *k* terms are selected.

The terms selected from document *d* should be added in the co-occurrence graph *G* of the text collection. However, computing the co-occurrence between the terms for each new document using a brute force approach, would make the process very expensive computationally. Thus, we propose an approximate method to obtain the pairs of terms with significant co-occurrence, based on a technique called Top-K Frequent Elements [8]. In this technique, *m* is the number of counters that monitor the occurrence of pairs of terms. For each pair of terms $e = \{t_i, t_j\}$ presented, If the pair is already monitored, then the value of its counter is incremented by 1. Otherwise, the pair is associated with a free counter and receives the initial value 1. If there aren't free counters, the new par *e* replaces the par e_{min} that has the lowest counter F_{min}. The value for the counter *e* is updated with F_{min+1}, in order to allow the emergence of new pairs.

When the value of a counter is increased, an event to update the co-occurrence graph is called. Then the pair of terms is inserted into the dendrogram, which is updated dynamically in order to maintain the correct values of similarity between the clusters terms. An overview of IHTC algorithm is illustrated in Figure 3. The process of updating the dendrogram is detailed as follow.

```
Algorithm IHTC
Parameters:
  S: TextSource, G: Graph, H: Dendrogram, List: Counters

for each new document D in S{
   Stopwords Removal
   Stemming
   TermSet = getKeywords(D)
   for each different pair of term e={t_i,t_j} in TermSet{
      if e is monitored {
         Increment the counter of e
         updateGraph(e, G)
         updateDendrogram(e, H)
      } else {
         if there are available counters
            Assign counter for e with value equal to 1
         else {
            Let e_min be the element with least value, F_min
            Replace e_min with e and assign F_min+1 for the counter of e
         }
      }
   }
}
```

Figure 3. Overview of the IHTC algorithm

3.2. Update Dendrogram Tree

The structure of the dendrogram has important information about the hierarchical clustering. The height of the nodes that merge two clusters indicates the similarity between them. The lower the height, the more similar the clusters. In a valid dendrogram, it is necessary that the height of the parent clusters is greater than or equal to that of their children. This property is exploited in the construction and dynamic update of the dendrogram.

When a pair of terms t_i and t_j is presented, the value of similarity between them *sim* = *SNN*(t_i,t_j) is computed. Thus, we need to update this new value of similarity in the dendrogram. There are four possible scenarios for this update:

1. If t_i and t_j does not exists in the dendrogram, that is, both are new examples, then a new node with similarity equal to *sim* is created in the dendrogrma, merging t_i and t_j;
2. If t_i exists in the dendrogram and t_j is a new example, then a new node with the similarity equal to *sim* is created in the dendrogram, merging t_j and the ancestor of t_i;
3. If t_i is a new example and t_j exists in the dendrogram, then a new node with the similarity equal to *sim* is created in the dendrogram, merging the ancestor of t_j and t_i;
4. Finally, if both t_i and t_j exists in the dendrogram and there is no common ancestor between them, then a new node with the similarity equal to *sim* is created, merging the ancestors of t_i and t_j. Otherwise, the similarity of the first common ancestor between t_i and t_j is updated to *sim*, if *sim* is greater than the current value of this ancestor.

Scenarios 2, 3 and 4 can make the dendrogram invalid, i.e., the value of similarity of a cluster parent is greater than the similarity of a cluster child. In this case, a correction procedure is executed to correct the node N_{new} responsible for the state invalid: (i) remove N_{new} from dendrograma; (ii) promote the child node with lowest similarity (N_{child1}); (iii) the other child node (N_{child2}) must be reinserted in the dendrogram. If N_{child2} is ancestor of t_i then N_{child2} is inserted as child of the first ancestor of t_j that has similarity greater than N_{child2}. Otherwise, N_{child2} is inserted as child of the first ancestor of t_i that has the similarity greater than N_{child2}. Thus, the dendrogram back to its valid state and the terms remain linked according to their similarity values. This updating process is detailed in the Figure 4.

To exemplify this process, in Figure 5 is illustrated an example of dendrogram updating. In (a) there is a valid dendrogram with 5 terms: A, B, C, D and E. Next there is an updating in the co-occurrence graph and the similarity between the terms B and C is updated to 0.7 (b). Thus, the similarity of the node {ABCD}, first common ancestor of B and C, is updated to the value 0.7. After this update, the dendrogram is in an invalid state, because the value of similarity of the parent node {ABCD} is greater than the value of the child node {AB}. In (c), the correction procedure is executed on the node {ABCD}, where the child node with lowest similarity {AB} is promoted and the

```
Algorithm IHTC::AdjustDendrogram
Parameters
  TermPair: e={ti,tj}, Node: Nnew, H: Dendrogram

Nchild1 = child of Nnew with lowest similarity;
Nchild2 = child of Nnew with highest similarity;
sim = Nnew.similarity();
Remove(Nnew);
Promote(Nchild1);

if( Nchild2.contains(ti) )  starting = tj;
else starting = ti;

newParent = starting.getParent();
while( newParent .getParent() != null  ){
   if(newParent .similarity() < sim) break;
   newParent= newParent .getParent();
}
newParent1 = child1 of newParent;
newParent2 = child2 of newParent;
if(newParent1.contains(starting)) brother = newParent1;
else brother = newParent2;

/* Reinserting Nchild2 */
newParent.removeChild(brother);
n = new dendrogramNode();
n.insertChilds(brother, Nchild2);
n.setSimilarity(sim);

newParent.insertChild(n);
AdjustDendrogram(n, e);
```

Figure 4. Dendrogram Adjustment

child node {CD} is reinserted. In (d), the dendrograma is in a valid state again.

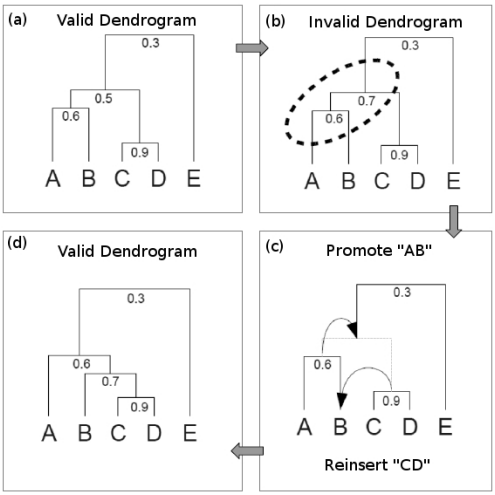

Figure 5. Example of Updating Dendrogram

The updates of the dendrogram is based on the nearest neighbors strategy, like the Single-Link. However, as only pairs of terms that co-occurrence has increased are updated, the resulting dendrogram is an approximation,

trying to focus on pairs of terms with co-occurrence more significant. The order of insertion of the terms can change the final structure of the dendrogram. The experimental evaluation shows that the algorithm is robust, i.e., the order of presentation of terms does not affect the quality of results significantly.

The next section presents an evaluation of the IHTC in benchmark text collections, and we compared the results with other algorithms in the literature.

4. Experimental Evaluation

The IHTC was evaluated experimentally in 8 textual collections. These collections have predefined topics, allowing to measure the precision and recall of topic hierarchies and make comparisons with other algorithms in an objective way.

Table 1 presents a summary of the text collections used in experimental evaluation. The collections *la1*, *la2*, *tr31* and *tr41* were obtained from the repository of TREC (Text Retrieval Conference) and the documents are newspaper articles. The collections *k1b* and *wap* were provided by the WEBACE project, and consist of documents from Yahoo! Directory. Finally, the collections *re0* and *re1* were obtained from Reuters-21578. All collections are available for download in the site of CLUTO project [12].

Table 1. Summary of Textual Collections

Textbase	Source	# Docs	# Terms	# Topics
k1b	WebACE	2340	13879	6
la1	LA Times (TREC)	3204	21604	6
la2	LA Times (TREC)	3075	21604	6
re0	Reuters-21578	1504	2886	13
re1	Reuters-21578	1657	3758	25
tr31	TREC	927	10128	7
tr41	TREC	878	7454	10
wap	WebACE	1560	8460	20

The results obtained with the IHTC were compared with algorithms Single-link (SL), Complete-Link(CL), UPGMA and FIHC. Some details of the experiment are described as follow.

- In the experiments with the algorithms SL, CL, FIHC and UPGMA, textual collections were preprocessed with stopwords removal, stemming and selecting only the terms that occur in two or more documents. The IHTC uses a particular technique for text preprocessing.
- For the algorithms SL, CL and UPGMA, the collections were represented using the vector-space model and the hierarchical clustering obtained with the cosine measure.
- The algorithm FIHC have two parameters to obtain the frequent itemsets. We selected the best parameters for each text collection, performing several runs of the algorithm.
- For the algorithm IHTC, we use *k=20* most frequent terms of each document. The co-occurrence graph was constructed using *m=1000* counters to monitor the occurrence of pairs of terms. These parameters were used for all textual collections. Parameter values were obtained by preliminary experiments and used in all textual collections.

We obtained topic hierarchies for each text collection, using the algorithms IHTC, SL, CL, UPGMA and FIHC. The hierarchies were evaluated with the FScore measure [6, 12], which treats each cluster as the result of a query and each predefined topic as the set of relevant documents of this query. Thus, it is possible to calculate values of precision and recall of each cluster of the hierarchy. In general, the higher the value of the FScore, the better the quality of topic hierarchy solution.

The Table 2 shows the values FScore obtained by each method in each text collection. The FScore values of the IHTC algorithm was estimated by the average value of 100 different runs for each collection, to measure the effect of the order of presentation of data.

Table 2. The FScores of the topic hierarchies

	IHTC	SL	CL	FIHC	UPGMA
k1b	0.729 ± 0.014	0.655	0.764	0.750	0.892
la1	0.526 ± 0.021	0.369	0.364	0.453	0.654
la2	0.535 ± 0.022	0.365	0.449	0.489	0.709
re0	0.649 ± 0.006	0.465	0.495	0.693	0.584
re1	0.685 ± 0.017	0.445	0.508	0.472	0.695
tr31	0.783 ± 0.018	0.532	0.804	0.765	0.816
tr41	0.691 ± 0.021	0.674	0.758	0.713	0.826
wap	0.536 ± 0.017	0.435	0.569	0.561	0.640

The results indicated that our algorithm obtained topic hierarchies with quality similar to existing algorithms in the literature. In some cases, the results were better. Table 3 shows the comparison between IHTC and the other four algorithms. The Rank Difference column shows the difference between the algorithms based on the ranking of the FScore values. Similarly, the Mean Difference column shows the difference based on mean FScore of each algorithm. Positive differences indicate advantage of our algorithm against the algorithm compared and negative differences show when our algorithm obtained inferior performance.

The Rank and Mean Differences allow the application of the statistical tests of Friedman (nonparametric) and SNK (parametric), respectively. These tests analyze whether the differences in results are statistically significant. The value

"NO" indicates that there is no evidence to ensure that the compared algorithms are different, otherwise the value would be "YES". Both tests were performed with 95% confidence level (p-value = 0.05).

Table 3. Statistical analysis of the results

	Rank Difference	Friedman Test	Mean Difference	SNK Test
IHTC x SL	16	NO	0.15	YES
IHTC x CL	1	NO	0.05	NO
IHTC x FIHC	0	NO	0.02	NO
IHTC x UPGMA	-13	NO	-0.09	YES

Based on the Friedman test, it is not possible to affirm that the our algorithm has better results than the other four algorithms. However, the SNK test indicates that our algorithm is more efficient than the SL but inferior than UPGMA. There is no evidence to indicate that the IHTC performance is superior or inferior than the CL and FIHC algorithms.

As expected, the UPGMA algorithm showed the best FScore values. However, this algorithm is not scalable to large databases [12], because of its high complexity $O(n^3)$. The FIHC obtained good results, but the computational cost is determined by the time of generation of frequent itemsets using the Apriori, which is usually high in textual data. The SL and CL algorithms, with $O(n^2)$ complexity, had inferior results than our algorithm in most textual collections. The complexity of the proposed IHTC algorithm in the worst case is $O(n \times k \times m \times t^2)$, where k and m are the parameters of the algorithm and t the number of different terms in the co-occurrence graph. Moreover, the number of terms of a text collection converge to a constant number when the number of documents is large [5]. Thus, the complexity of IHTC is linear according to the number of documents and scalable in large textual collections.

Based on this experimental evaluation, the IHTC algorithm is an attractive alternative because it provides competitive results and allows the construction of topic hierarchies in a incremental way, with labels to facilitate the interpretation of the results.

5. Conclusions and Future Works

In this paper, we presented an incremental method for the construction of topic hierarchies using hierarchical term clustering. We introduced the algorithm IHTC that is able to preprocess the textual collections, obtain co-occurrence graphs and a dendrogram with the terms of the collection. The results obtained with the IHTC is comparable to traditional off-line clustering algorithms, however, our incremental algorithm has the ability to process growing text collections. Moreover, the proposed algorithm is scalable because it has linear time complexity when the number of documents is large. We provide the IHTC source code, runtime performance analysis, and a online demonstration of the IHTC in the our project page[1].

In the future, we plan to improve our evaluation method. For example, we will evaluate the effect of the parameters k and m used in the IHTC. Moreover, we intend to compare the IHTC with other incremental algorithms, commonly used in text streams.

References

[1] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proceeding of 20th International Conference on Very Large Data Bases, VLDB*, volume 1215, pages 487–499, 1994.

[2] B. S. Everitt, S. Landau, and M. Leese. *Cluster Analysis*. Arnold Publishers, 2001.

[3] R. Feldman, M. Fresko, Y. Kinar, Y. Lindell, O. Liphstat, M. Rajman, Y. Schler, and O. Zamir. Text mining at the term level. In *Proceedings of the PKDD'98*, pages 65–75. Springer Verlag, 1998.

[4] R. Feldman and J. Sanger. *The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data*. Cambridge University Press, 2006.

[5] B. Fung, K. Wang, and M. Ester. Hierarchical document clustering using frequent itemsets. In *Proceedings of the SIAM International Conference on Data Mining*, 2003.

[6] B. Larsen and C. Aone. Fast and effective text mining using linear-time document clustering. In *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 22. ACM, 1999.

[7] Y. Li, S. Chung, and J. Holt. Text document clustering based on frequent word meaning sequences. *Data & Knowledge Engineering*, 64(1):381–404, 2008.

[8] A. Metwally, D. Agrawal, and A. El Abbadi. Efficient computation of frequent and top-k elements in data streams. In *Proceedings of the 10th ICDT International Conference on Database Theory*, pages 398–412. Springer, 2005.

[9] M. F. Moura, R. M. Marcacini, B. M. Nogueira, M. da Silva Conrado, and S. O. Rezende. A proposal for building domain topic taxonomies. In *Proceedings of International Workshop on Web and Text Intelligence - 19th SBIA Brazilian Symposium on Artificial Intelligence (SBIA)*, pages 83–84. São Carlos, Brazil, 2008.

[10] G. Salton. *The SMART Retrieval System—Experiments in Automatic Document Processing*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1971.

[11] R. Sole, B. Murtra, S. Valverde, and L. Steels. Language Networks: their structure, function and evolution. *Trends in Cognitive Sciences*, 12(42):343–352, 2005.

[12] Y. Zhao and G. Karypis. Evaluation of hierarchical clustering algorithms for document datasets. In *Proceedings of the 20th International Conference on Information and knowledge management*, pages 515–524. ACM New York, NY, USA, 2002.

[1] http://sites.labic.icmc.usp.br/marcacini/ihtc/