

## Reverse Tree Clustering

Casey R. Bartman, and Jamal Alsabbagh, Grand Valley State University

**Abstract**—Common document clustering algorithms utilize models that either divide a corpus into smaller clusters or gather individual documents into clusters. Hierarchical Agglomerative Clustering, a common gathering algorithm runs in  $O(n^2)$  to  $O(n^3)$  time, depending on the linkage of documents. In contrast, Bisecting K-Means Clustering has been shown to run in linear time with respect to the number of documents to cluster, although other factors significantly affect run time. We propose a clustering algorithm bases on an inverted-index matrix of terms and an inverted term tree model.

### I. INTRODUCTION

As the number of documents available from the web and other sources increase, there has developed a need for robust document clustering algorithms. Most clustering algorithms represent documents as vectors in  $n$ -dimensional space, where  $n$  is a set of terms that occur in the documents. Documents are then clustered based on the similarity of their vector representation [1]. Some shortcomings of this approach have been well documented [2]:

1. Many dimensions for a given document are null. This results in a large number of multiplications that have no affect when utilizing methods such as cosine similarity. Operational overhead is therefore introduced for dimensions that are only significant when comparing a limited number of documents in the corpus.
2. *K-means* variations may require fore knowledge of the number of clusters that should be created. This value may be unknown, especially in unsupervised clustering.
3. The evaluation of outlying documents. Are they secondary to a need for an additional cluster per two above? Do they represent a need for larger variation in the clusters?

This paper presents an implementation of an algorithm based upon Mock's Tree Clustering technique [3], whereby the vector model is not used. Instead, documents are associated with

terms and clustering is performed based upon them intersection. The advantage of this approach is that the overhead due to null dimensions (in the vector model) is eliminated. This work also implements Mock's proposal for generating overlapping clusters. In addition, cluster generation was accomplished by utilizing an inverted-index matrix as described by Salton [2].

This preliminary study demonstrated a robust method for classifying short text documents. Our goal was to develop clusters of reasonable significance to the end user, rather than to seek tight cohesion among clusters. Ongoing evaluation is underway to compare Tree Clustering with Bisecting K-Means Clustering regarding actual run times against the same corpus and the quality of the clusters formed.

### II. THE TREE CLUSTER MODEL

The basis of Mock's Tree Cluster technique is an Apriori algorithm that used an item set lattice to a depth of three terms. Candidate clusters were generated by brute force utilizing 100 terms. Terms were selected by evaluating the *tfidf* values of terms. Terms that had a *tfidf* value of between .05 and 1 were chosen.

The largest clusters were evaluated first, and documents in those clusters were removed from further consideration. Once all three term candidate clusters were considered, the process was repeated with two term candidate cluster sets. Mock also proposed an extension that allowed overlapping clusters. In this model terms were not eliminated until all clusters were created at a given level. Our implementation utilizes overlapping clusters.

### III. THE REVERSE TREE MODEL

As in Mock's model an Apriori algorithm with an item set lattice to a depth of three terms was utilized. Term selection was performed by selecting a term threshold, or minimum support, and removing stop words. Candidate clusters were again generated by a brute force method.

Candidate clusters were then evaluated at the three term level. While a minimum cluster size was selected for a candidate cluster set to be considered valid, all documents were considered at a given lattice depth. This allowed cluster overlap or fuzziness among clusters.

IV. DATA SOURCE AND PREPERATION

The standardized collection of 21578 Reuter’s news articles, as compiled by Dr. David D. Lewis, Ph.D. [4], was utilized in this study. This collection of articles represents a subset of Reuter’s articles that were published in 1987. This subset consists of articles from three time periods; Spring, Summer, and Fall. The articles are contained in a set of 21 files.

A goal of this study was to develop a clustering system that would lend itself to rapid clustering of documents or web pages. It was therefore elected to develop a clustering system based solely on the title data. This corpus consisted of few common terms. Only 11 terms after limited stemming occurred in more than 5% of the titles. As the title data consist of a short set of terms, this would simulate the terms that may be listed for a web page meta content under keywords or description.

A sample subset of 2% of the articles was created that demonstrated the following inconsistencies with the data:

1. Most articles were formatted in uppercase, but not all.
2. A large number of numeric values existed (i.e. 10, 1,000,000, 5.3, etc). Furthermore, the formatting of the numeric values was not consistent.
3. The tense of verb was not consistent.
4. Plurals of nouns occurred.
5. Multiple spellings for the same word occurred.
6. Words that referred to similar topics (i.e. Japan and Japanese).
7. Non consistent combination of words (i.e. German-mark, German/mark).

Review of the numeric values contained within the titles demonstrated that in many cases the numeric values were nouns and not adjectives. Salton, in his review on term selection recommended the use of nouns and certain verbs as ideal for use in clustering. Numeric values were therefore translated into string values for given ranges that were consistent with how those

values were utilized as nouns. This is summarized in Table I.

TABLE I.  
String conversions of numeric values.

Numeric Range	Replacement Term
Values < 10	INT
Values < 100, Value ≥ 10	TENS
Value<1000, Value ≥ 100	HUNDREDS
Value<100,000, Value≥1,000	THOUSANDS
Value ≥100,000	BIG

The stemming of the titles consisted of translating ranges of numeric values into terms (Table I), removing stop words, and limited term consolidation. In doing so, no formal stemmer algorithm was utilized on the terms.

An evaluation of the titles of the entire set of 21,578 articles revealed that they contained 13,744 distinct terms, with a total count of 138,337 terms.

It was elected to utilize a document frequency of 30 to 1,079 documents for significant terms. Only 11 terms occurred in more than 1,079 documents. This is believed to reflect the limited length of any given title in the corpus. The final terms set consisted of 643 terms.

A minimum document frequency of 30 was selected based on a graphical review of term selection. At document frequencies of less than 30, there was an exponential increase in the number of terms selected. Figure 1 demonstrates the number of terms selected versus the document frequencies.

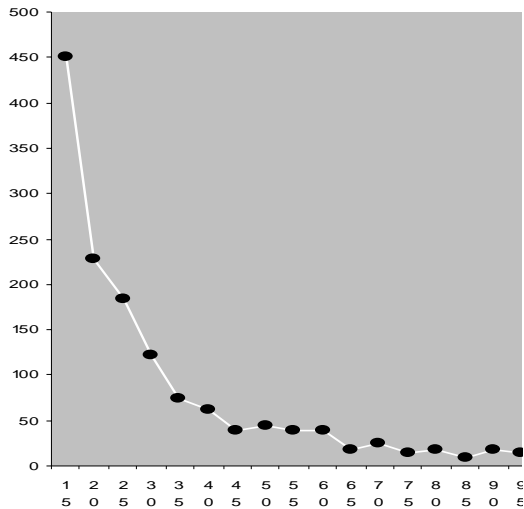


Figure 1. Term Selection v. Document Freq.

## V. REVERSE TREE ALGORITHM STRUCTURE

The characteristics of the proposed algorithm are based on a Boolean determination as to whether a document contains a key term. Documents were not weighted as to the number of occurrences of a term within them or the document length. If a document contained a key term, the id number of the document was associated with the term in a Java LinkedList Object. For example a terms set is represented as:

LinkedList={ $d_a, \dots, d_z$ }

And a cluster is represented as:

$$Cluster = LinkedList_1 \cap LinkedList_2$$

Intersections of the LinkedList object were performed at the three term cluster level. The number of documents present in a candidate set, to be considered valid, was varied in the evaluation as discussed below. A document could be present in more than one candidate set. Allowing documents to be present in more than one candidate set, there by introducing fuzziness into the algorithm.

Documents that existed in a least one valid three term cluster were then removed from further consideration.

The remaining documents were evaluated in a similar manner at the two term level. Every three

term cluster had three possible two term roots and these two term clusters were added to the three term leaf. Two term clusters that did not have a three term leaf formed new clusters. The same number of documents, or minimum support, had to be present in the two term candidate sets as in the three term sets for that set to be considered valid.

Documents that existed in a least one valid two term cluster were then removed as before. The remaining documents were then evaluated at the one term level. These were processed as above. Any document not in a cluster after this phase was considered an outlying document.

## VI. ALGORITHM IMPLEMENTATION

The implementation of the algorithm required two passes through the document data. The first pass compiled a set of terms present in the set and their frequency. The second pass associated documents with the generated candidate terms. Total time for execution of these two passes was 80 seconds. All processing times for implementation utilizing Sun Java 1.4.2, an AMD 2400Mhz processor and 1GB or ram.

The candidate sets at the three term level were then evaluated by a brute force method. This did require the potential generation of 6,099,006 candidate sets. The number of candidate sets is only a potential, as the intersection of the LinkedList was performed in series. This method was felt to be more robust than support based pruning for there was little memory overhead in the execution. Secondary to the above, two term clusters had to be evaluated de novo after creating the three term clusters.

Various minimum cluster sizes were evaluated. The degree of overlap of clusters or their fuzziness was inversely proportional to the minimum number of documents required in a candidate set to be considered valid. The result of varying the minimum support for cluster formation is demonstrated in Table II.

TABLE II  
Cluster Formation v. Minimum Support

Min. Cluster Size	3	20
1 Term Clusters	601	518
2 Term Clusters	1867	352
3 Term Clustes	6949	109
Docs. Not Clustered	932	967

## VII. CONCLUSION AND FUTURE EVALUATION

The Reuter's document set of 21578 articles presents a minimum number of documents that a algorithm must be capable of processing in a robust manner as current business needs extend to document sets of 2-3 million [5].

The K-means algorithm is frequently used as a tool in clustering of data. There exist several limitations in utilizing the standard *K-Means* algorithm to evaluated text data.

A primary problem that was appreciated when evaluating the Reuter's data was how to determine the number of clusters that exist in the data.

Two traditional methods of solving the problem are hierarchical agglomerative clustering and bisecting k-means. Both of these algorithm solve the problem, but at a high operational overhead. This limits the usefulness of either algorithm when presented with a large document set.

A secondary problem is the requirement of document clustering algorithm to allow overlap of clusters of a Fuzzy Algorithm. While Fuzzy K-Means variants exist, they require apriori knowledge of the number of clusters to create.

The Reverse Tree algorithm presented addresses both of these issues. No foreknowledge is required of the number of clusters to create and fuzziness is allowed. While the Reverse Tree algorithm is not a vector space model, a comparison can be made.

A random evaluation of the clusters formed did reveal suitable clustering. Neither the cohesion nor the F values based on previous classification of the documents were considered.

While the above results demonstrated that this implementation of a Reverse Decision Tree algorithm provided a suitable method of clustering of the Reuter's news articles, it was not found to be satisfactory for human interface. A report of clusters generated, with one article number per line resulted in a document over 1900 pages in length.

This problem could be addressed with a tree type GUI interface. As this was a preliminary evaluation of the algorithm, such a GUI was not generated.

One solution to the above was to generate a program in which the user could enter terms to form custom searches. Using the same term LinkList Objects that were used to generate the clusters, the user could generate a set of clusters that match their inquiry. Run times for this user guided clustering were less than one second.

A reverse tree rule set generated a satisfactory classification algorithm for a large data source in reasonable time. Supervised learning was required to generate a list of synonyms and stop words for document terms, as was it required in creating a threshold number of occurrences for a term to be significant. The use of an array of term LinkedLists allowed for rapid execution of the clustering and a workable user interface. On going evaluation of the reverse tree algorithm is currently under way on data sets of larger text documents. Other models of term selection are under review that allows robust run times on the document set as well.

### Referemces

- [1] M. Steinbach, G. Karypis, and V. Kumar, "A Comparison of Document Clustering Techniques", University of Minnesota, Technical Report #00-034 (2000). [http://www.cs.umn.edu/tech\\_reports/](http://www.cs.umn.edu/tech_reports/)
- [2] Gerard Salton, *Automatic Text Processing*, chapter 5, Massachusetts: Addison-Wesley, 1989.
- [3] K. Mock, "A Comparison of Three Document Clustering Algorithms: TreeCluster, Word Intersection GQF, and Word Intersection Hierarchical Agglomerative Clustering". Intel Technical Report 9/23/1998.
- [4] Lewis, D., Available: <http://www.daviddlewis.com/resources/testcollections/reuters21578/>
- [5] Person communication—Compulit, Grand Rapids, MI.