High Quality, Efficient Hierarchical Document Clustering using Closed Interesting Itemsets

Hassan H. Malik and John R. Kender
Department of Computer Science
Columbia University
New York, NY 10027
{hhm2104, jrk}@cs.columbia.edu

Abstract

High dimensionality remains a significant challenge for document clustering. Recent approaches used frequent itemsets and closed frequent itemsets to reduce dimensionality, and to improve the efficiency of hierarchical document clustering. In this paper, we introduce the notion of "closed interesting" itemsets (i.e. closed itemsets with high interestingness). We provide heuristics such as "super item" to efficiently mine these itemsets and show that they provide significant dimensionality reduction over closed frequent itemsets.

Using "closed interesting" itemsets, we propose a new hierarchical document clustering method that outperforms state of the art agglomerative, partitioning and frequent-itemset based methods both in terms of FScore and Entropy, without requiring dataset specific parameter tuning. We evaluate twenty interestingness measures on nine standard datasets and show that when used to generate "closed interesting" itemsets, and to select parent nodes, Mutual Information, Added Value, Yule's Q and Chi-Square offers best clustering performance, regardless of the characteristics of underlying dataset. We also show that our method is more scalable, and results in better run-time performance as compare to leading approaches. On a dual processor machine, our method scaled sub-linearly and was able to cluster 200K documents in about 40 seconds.

1. Introduction and Related Work

Organizing data into a tree-like hierarchy has many applications. A hierarchy provides a view of the data at different levels of abstraction, helping users deal with the common problem of information overload. As the user expands nodes at different levels in the hierarchy, the structure within the broad topic becomes more apparent as parent and child nodes are organized in a general to specific fashion. These benefits make hierarchies a logical choice to organize large collections of documents and during last few decades, various approaches were proposed to produce cluster hierarchies from document collections.

Agglomerative and partitioning-based approaches represent two most popular categories of hierarchical document clustering techniques. Agglomerative approaches start with a singleton cluster for each document and build the hierarchy bottom-up by applying various pair-wise similarity measures on clusters, and merging the cluster pair with highest similarity at each step, until only one cluster remains. Agglomerative methods generally suffer from their inability to perform adjustments once a merge is performed, resulting in lower clustering accuracy. They also have a very high computational cost [5], making them infeasible for large document datasets. On the other hand, partitioning approaches obtain hierarchical clustering solutions via a sequence of repeated bisections [5] and are generally scalable and efficient. Steinbach et al. [1] showed that Unweighted Pair Group Method with Arithmetic Mean (UPGMA) [2] and bisecting k-means, a variant of standard k-means are the most accurate agglomerative and partitioning methods, respectively [4]. Furthermore, Zhao and Karypis [5] recently showed that the I_2 criterion function outperforms other criterion functions when used with bisecting k-means.

A recent trend in hierarchical document clustering is to use frequent itemsets to produce cluster hierarchies. HFTC [6] was the first algorithm in this class and achieves accuracy comparable to 9-secting k-means, and worst than bisecting k-means. Fung et al. [3] showed that HFTC is not scalable for large document collections and proposed FIHC; a frequent itemset based clustering approach that claims to outperform

HFTC and the best-known agglomerative and partitional methods (i.e. UPGMA and bisecting k-means) both in terms of accuracy and scalability. More recently, Yu et al. [7] proposed TDC that uses only closed frequent itemsets and further reduces dimensionality, while improving the clustering quality and scalability over FIHC. To our surprise, a more fair comparison (section 5.6) revealed that both FIHC and TDC actually perform worse than UPGMA and bisecting k-means.

Based on the observation that higher frequency does not necessarily mean higher quality, and combining ideas from research in selecting the most interesting association rules, and closed frequent itemset mining, we introduce the notion of "closed interesting" itemsets in this paper. We provide a simple, parallelizable algorithm, and necessary heuristics to efficiently mine these itemsets. We present results from extensive experiments performed on standard datasets of varying characteristics and sizes, and show that using the same support threshold for first level (single word) itemsets results in significantly smaller number of "closed interesting" itemsets as compare to the number of closed frequent itemsets generated. Even so, when used for hierarchical document clustering, we show that "closed interesting" itemsets outperform state of the art clustering algorithms, indicating their superior quality.

We present a hierarchy assembling approach that supports soft clustering and prunes unwanted itemsets on the way. In order to make the hierarchy more compact, existing approaches [3, 7] use agglomerative clustering to merge the first-level nodes. Although significantly less expensive than applying agglomerative clustering on the whole dataset, this step is still very expensive. We used bisecting k-means to reduce computational complexity of this step. Finally, we propose various implementation-level optimizations throughout the paper. Figure 1 provides an overview of our hierarchical document clustering process.

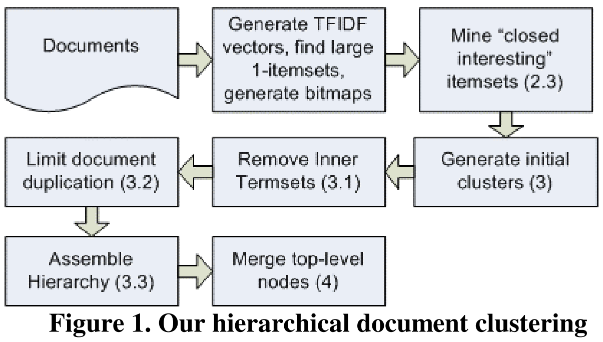

Documents
Generate TFIDF vectors, find large 1-itemsets, generate bitmaps
Mine "closed interesting" itemsets (2.3)
Generate initial clusters (3)
Remove Inner Termsets (3.1)
Limit document duplication (3.2)
Assemble Hierarchy (3.3)
Merge top-level nodes (4)

Figure 1. Our hierarchical document clustering process; numbers refer to sections in this paper

The rest of this paper is organized as follows. Section 2 introduces "closed interesting" itemsets and provides a simple mining algorithm. Section 3 covers various pruning steps and details on hierarchy generation. Section 4 provides details on our approach of merging first level nodes. In section 5, we present experimental results on several popular datasets. Finally, we conclude and present our ideas for future work in section 6.

2. Mining Closed Interesting Itemsets

2.1. Motivation

Frequent itemset mining often results in too many itemsets. Using a faster mining algorithm does not always help as it is fundamentally a combinatorial problem and the mining time exponentially increases as support threshold linearly decreases [7, 8], regardless of the mining algorithm used. Researchers found that a large percentage of frequent itemsets shares support with one or more of their parent (subset) itemsets. These itemsets are considered insignificant as they represent "specialization" of the more general concept represented by the parent itemset. "Closed" frequent itemset mining utilizes this finding and imposes the additional requirement of "closeness" for large itemset generation. Specifically, in addition of meeting the minimum support threshold, closed frequent itemsets must also have support that is different from (practically less than) any of their subset itemsets. Generally, it results in significantly smaller number of closed frequent itemsets, when compared with the number of frequent itemsets found on the same dataset, using the same support threshold. In addition, closed frequent itemsets performed better than frequent itemsets in a number of applications, such as hierarchical document clustering [7].

Finding the most interesting association rules is another significant thread in data mining research. A number of association rules can be generated from each large itemset at each level, which often results in a very large association rule base [12], especially when attributes in the data set are highly correlated [11]. A low support threshold results in too many discovered associations. Increasing the support threshold significantly reduces the number of rules discovered, but risks losing useful associations, especially on uneven datasets. On the other hand, Confidence is criticized because of its asymmetric property and its failure to incorporate the baseline frequency of the consequent [9]. In addition, it is non-trivial to set good values for support and confidence thresholds; it depends on the size of dataset, sparseness of data, and the particular problem under study [10]. Considering these issues, a number of researchers [10, 11, 13, 14] proposed alternate interestingness measures to evaluate and rank discovered associations. Inspired from various statistical and mathematical principles, these measures are considered less sensitive to the properties of specific datasets.

2.2. Overview of "closed interesting" itemsets

We argue that while the "closeness" requirement of closed frequent itemsets is useful and based on solid principals, the other requirement of meeting a minimum support threshold is problematic and difficult to generalize. Combining the stronger aspects of closed frequent itemset mining with research in finding the most interesting association rules, we propose a new kind of itemsets called "closed interesting" itemsets.

Table 1. List of interestingness measures used, section 5.4 provides details on threshold values

#	*Symbol*	*Interestingness Measure*	*Threshold*
1	AV	Added Value	0.4
2	c	Symmetric Confidence	0.6
3	F	Certainty Factor	0.4
4	χ^2	Chi-Square	$unit = 50$, $p = 3000$
5	S	Collective Strength	1.45
6	V	Conviction	1.7
7	Φ	Correlation Coefficient	0.35
8	IS	Cosine	0.33
9	G	Gini Index	0.017
10	I	Interest	12
11	Z	Jaccard	0.23
12	J	J-Measure	0.02
13	κ	Kappa	0.35
14	K	Klosgen's	0.068
15	L	Laplace	0.6
16	M	Mutual Information	0.1
17	α	Odds Ratio	25
18	RI	Piatetsky Shapiros Interest	0.02
19	Q	Yule's Q	0.85
20	Y	Yule's Y	0.65

These itemsets retain the "closeness" property of closed frequent itemsets, but replace the minimum support requirement with meeting minimum threshold of a symmetric, statistically inspired objective interestingness measure. Table 1 lists the measures used in our experiments. Computational details of these measures can be found in [9, 10, 12, 13, 15, 16, 17, 18]. Note that some of these measures are not inherently symmetric and are converted to a symmetric version by calculating the interestingness values for both directions and selecting the maximum value, as proposed by Tan et al. [13]. We compare their relative performance and recommend a small number of measures that we experimentally found least sensitive to the properties of specific datasets in section 5.

Furthermore, most of these measures are meant to calculate correlation or interdependence between two-way contingency tables (i.e. two variables), which makes them unusable for generating "closed interesting" itemsets with more than two items. While measures like log-linear analysis [10] exist to calculate interdependence between multi-way contingency tables, they are computationally expensive. We define a simple greedy heuristic to deal with this problem:

Super item: If an itemset p at level k is used to generate a candidate itemset q at level $k + 1$ (i.e. itemset q contains all k-items from itemset p, and exactly one additional item u), all items in itemset p are used to form a super item S, with support (S) = support (p). Items v and u are used to form a two-way contingency table and to calculate interestingness values.

Example: Considering a dataset of 200 transactions, support (A) = 98, support (B) = 120, support (C) = 65, support (A, B) = 80 and support (A, B, C) = 45. If itemset "A, B" at level 2 is used to generate a candidate itemset "A, B, C" for level 3, a super item S is formed with support (S) = support (A, B) = 80. Since "C" is the additional item in the candidate itemset, a contingency table is formed between S and C, as shown in table 2.

Table 2. A 2 x 2 contingency table between super item "*S*" and item "C"

	C	¬C	Total
S	45	35	80
$\neg S$	20	100	120
Total	65	135	200

Using the contingency table shown in Table 2 and "Correlation Coefficient" as interestingness measure, we get an interestingness value of 0.414, which shows that the super item S and item C are positively correlated [13].

Similar to frequent itemset mining, we prune candidate itemsets for level k if any of their k subsets of size k-1 do not exist in the previous level, with a caveat that frequent itemset mining uses support that has a downward closure property, providing theoretical foundation for this step. We empirically found this step to be useful in increasing the quality and reducing the number of "closed interesting" itemsets generated. We leave the theoretical analysis for future work.

```
1) result = Φ
2) $I_1 = U_1$ = {large 1-itemsets}
3) for (k = 2; $I_{k-1}$ != 0; k ++) do begin
4)         $I_k$ = find-interesting-itemsets($I_{k-1}$, $U_{k-1}$)
5)         append(result, $I_k$)
6)         $U_k$ = get-unique-items($I_k$)
7) end
8) answer = result
```

(a) Algorithm ***CII-MINE***

```
1) find-interesting-itemsets(super_items, unique_items)
2)         interesting_itemsets = Φ
3)         for (i = 0; i < size(super_items); i ++) do begin
4)                 for (j = index-of(get-last-item(super_items[i]), unique_items) + 1; j < size(unique_items); j ++) do begin
5)                         candidate_itemset = super_items[i] U unique_items[j]
6)                         subset_itemsets = find-subset-itemsets(candidate_itemset)
8)                         if (contains($I_{k-1}$, subset_itemsets)) then
9)                                 if (closed(candidate_itemset, subset_itemsets, $I_{k-1}$)) then
10)                                        val = apply-measure(super_items[i], unique_items[j])
11)                                        if (val >= min_interestingness_threshold) then
12)                                                append(interesting_itemsets, candidate_itemset);
13)                                        end
14)                                end
15)                        end
16)                end
17)        end
18)        answer = interesting_itemsets
19) end
```

(b) Method ***find-interesting-itemsets***

Figure 2. A simple "closed interesting" itemset mining algorithm

2.3. Itemset Mining

Figure 2 (a) presents CII-MINE, a simple algorithm to mine "closed interesting" itemsets. The algorithm starts with mining large 1-itemsets (individual words) in a way similar to frequent itemset mining. We experimentally found that using a very low support threshold for this step results in best quality itemsets (section 5.7), adding credence to the claim that using a high support threshold results in pruning useful associations [13]. Each of the kth steps (where k >= 2) form candidate itemsets by considering all "closed interesting" itemsets found in k-1th step as super items, and adding the unique individual items that follow the last item in the super item. Each candidate is checked for downward closure and closeness, and candidates that satisfy both requirements are checked for meeting the interestingness threshold. Candidates that satisfy all three requirements are added to the list of "closed interesting" itemsets for step k. Mining stops when all "closed interesting" itemsets are found.

Example: If mining "closed interesting" itemsets with k = 2 and large 1-itemsets resulted in "closed interesting" 2-itemsets (I_2) in Table 3, U_2 = {a, b, c, d, e, f}. Mining "closed interesting" itemsets for k = 3 would be done as represented in Table 4. Note that super items {b, f}, {d, f} and {e, f} are not considered because there are no items following 'f' in U_2.

Table 3. Interesting 2-itemsets and their support

2-itemset	*Support count*	*2-itemset*	*Support count*	*2-itemset*	*Support count*
{a, b}	150	{b, d}	140	{c, e}	200
{a, d}	280	{b, e}	320	{d, f}	94
{b, c}	120	{b, f}	85	{e, f}	10

Since k = 3, size of each super item = k – 1 = 2. The algorithm first explores all candidate items for super item {a, b}. Since 'b' is the second item in U_2, four candidates {a, b, c}, {a, b, d}, {a, b, e} and {a, b, f} are formed, using items that follow 'b' in U_2. Each candidate is checked for the meeting downward closure requirement, and candidates that do not meet this requirement are pruned (i.e. {a, b, c} is pruned as {a, c} does not exist in I_2). Similarly, candidate itemsets that do not meet the closeness requirement (i.e. {b, c, e}) are pruned. Interestingness values of the remaining candidates are calculated by calling "apply-measure", and passing super and unique items, (i.e. {{a, b}, d} and {{b, d}, f}). Candidates that satisfy the minimum interestingness threshold are added to the result. Note that support for candidate itemsets is only calculated if they meet the downward closure requirement. In addition, we optimized the support calculation performance by using bitmaps that indicate presence / absence of individual, large 1-items in all documents (i.e. where each bit represents a document) and ANDing the bitmaps of all items in an itemset.

Table 4. Mining "closed interesting" 3-itemsets, using 2-itemsets from Table 3, (NC = not calculated)

super_item		***Candidate itemset***	***Supp. Count***	***Comments***
a	b	~~{a, b, c}~~	NC	{a, c} not in I_2
a	b	{a, b, d}	52	Calcul. interest. ({a,b}, d)
a	b	~~{a, b, e}~~	NC	{a, e} not in I_2
a	b	~~{a, b, f}~~	NC	{a, f} not in I_2
a	d	~~{a, d, e}~~	NC	{a, e} not in I_2
a	d	~~{a, d, f}~~	NC	{a, f} not in I_2
b	c	~~{b, c, d}~~	NC	{c, d} not in I_2
b	c	~~**{b, c, e}**~~	120	Not closed: same as (b, c)
b	c	~~{b, c, f}~~	NC	{c, f} not in I_2
b	d	~~{b, d, e}~~	NC	{d, e} not in I_2
b	d	{b, d, f}	72	Calcul. interest. ({b,d}, f)
b	e	~~**{b, e, f}**~~	10	Not closed: same as (e, f)
c	e	~~{c, e, f}~~	NC	{c, f} not in I_2

3. Hierarchical Document Clustering and Itemset Pruning

Our hierarchy construction approach is similar to FIHC [3] and TDC [7], with various important differences, most significant of which relates to how parent nodes are selected. An initial cluster is formed for each "closed interesting" itemset, containing all documents that contain the itemset, with items in the itemset used as the cluster label. In fact, these clusters are readily available as a byproduct of calculating support using the bitmap-based representation discussed in the previous section. These initial clusters are not disjoint, as a document can contain multiple "closed interesting" itemsets of varying sizes. Sections 3.1 and 3.2 discuss our approach to limit document duplication. Section 3.3 presents our hierarchy construction algorithm. This step significantly differs from existing approaches as it allows selecting multiple parents, using the interestingness between parent and child nodes without inspecting cluster contents.

3.1. Inner termset removal

If a document is contained in multiple clusters that are based on itemsets of varying sizes, we reduce document duplication by pruning the document from all but the clusters based on the largest sized itemsets. Later, when these itemsets are used to build the hierarchy, this step results in each document assigned to all applicable nodes at the highest possible (i.e. most specific) level in the hierarchy. Figure 3 presents an algorithm that performs this step in a single pass on discovered "closed interesting" itemsets, without processing individual documents.

```
1) {allocate array global_map}
2) {allocate array lev_maps with size = k}
3) for (i = k; i >= 1; i--) do begin
4)         forall itemsets $t \in I_i$ do begin
5)                 $bitmap_t$ = $bitmap_t$ AND (NOT global_map)
6)                 lev_maps [i] = lev_maps [i] OR $bitmap_t$
7)         end
8)         global_map = global_map OR lev_maps [i]
9) end
```

Figure 3. Inner-termset removal algorithm, where k = size of the largest discovered itemset

The algorithm starts by allocating a global, and individual coverage maps for each level, where number of levels = size of largest discovered itemset. A level coverage map is similar to an itemset bitmap with a difference that an itemset bitmap indicate documents that contain the itemset where as a level coverage (bit) map indicate documents that contain any itemset at that level. Similarly, the global coverage map indicates documents that contain any discovered itemset. Levels are iterated in largest to smallest order and at each level; bitmaps of all itemsets that exist at that level are ANDed with inverse of bits in global coverage map, which results in eliminating documents that already existed at a higher level. The updated bitmap is used to update the current level's coverage map. Finally, after each level, current level's documents are added to the global coverage map. This results in pruning documents from all but their largest-sized itemsets.

Example: Considering a dataset of 10 documents, and itemset *x* at level *i*, with $bitmap_x$ = {0100100001}, and global map updated with all documents that exist on levels i + 1 to k, such as global_map = {0010100101}, we have:

```
$bitmap_x$              = {0100100001}
NOT global_map          = {1101011010} AND
------------------------------------------
$bitmap_x$              = {0100000000}
```

Note that two documents were pruned from $bitmap_x$, as they existed in itemset(s) at a higher level.

3.2. Constraining document duplication

The inner-termset removal algorithm (Figure 3) also prepares coverage maps for individual levels. These coverage maps are used to limit document duplication at the same (their largest) level, as inner-termset removal eliminates documents from all but their largest applicable itemsets, and documents may still exist in multiple itemsets at their largest level. Using level coverage maps, documents that exist at each level are checked for existence in itemsets (clusters) at that level. If a document exists in more than MAX_DOC_DUP (user defined parameter) itemsets, a score is calculated against each matching itemset and the document is assigned to MAX_DOC_DUP itemsets with highest scores. We used a score calculation method similar to TDC [7], which uses the document's TFIDF vector (includes large 1-itemsets only) and adds the term frequencies of items that existed in the itemset.

3.3. Bottom-up hierarchy assembling, constraining node duplication and pruning of itemsets

TDC [7] builds the hierarchy by linking each itemset of size k with all of its (up to k) subsets at level k-1. This approach may result in boosting *FScore*, but would impact the overall clustering quality because of too much node duplication. On the other hand, FIHC [3] applies an expensive similarity calculation method, which first prepares a conceptual document for each node (i.e. by merging the TFIDF vectors of all documents that exist in the node or any of its children) and calculating a score against each of its (up to k) parents. The node is linked to the parent with the highest similarity. This method is expensive because it requires preparing conceptual documents for nodes at all levels in the hierarchy (conceptual documents for first level are not needed by this step, but at the time of merging first level nodes later), and also because the similarity calculation method uses the notion of "cluster frequent items" which requires an additional step to find these items for each node, using the documents that exist in that node and any of its child nodes. It also adds another parameter to the system (i.e. "minimum cluster support") and as discussed earlier, support thresholds are not easy to generalize. Finally, assigning each node to exactly one parent does not support soft clustering, which is an essential element of real-life hierarchies. As an example, a large number of nodes in the "Yahoo Directory" are cross-linked between various categories.

We avoid both extremes (i.e. TDC, which assigns each node to all available parents and FIHC which assigns each node to exactly one parent) and propose a more balanced approach that assigns each node to up to a user-defined number of best matching parents. Our method is also computationally efficient, as it does not prepare conceptual documents for nodes at various levels in the hierarchy and also does not calculate cluster support, and hence, avoids the additional mining step. Instead, we used the same "interestingness" measure that was used to mine "closed interesting" itemsets in the previous step, and our "super item" heuristic to calculate the interestingness between the itemset at level k and its (upto k) parent itemsets at level k-1 (i.e. by considering the parent itemset as super item). A node is linked to up to MAX_NODE_DUP (user defined parameter) parents with the highest interestingness values. This method does not look into the documents contained in the cluster and selects parents solely using the itemsets (i.e. cluster labels).

Figure 4 presents our bottom-up hierarchy construction algorithm. Because of inner termset removal and constraining maximum document duplication, a number of itemsets may no longer have any documents associated to them (i.e. empty clusters). They are pruned on the way unless they were used as parent by a node at level k + 1.

4. Merging First Level Nodes

Generally, itemset mining results in a large number of large 1-itemsets (frequent single words), making the first-level nodes very sparse. Removing inner termsets and constraining document duplication results in a number of empty clusters, which are pruned during the hierarchy construction. Still, there may be a large number of nodes at level 1. Similar to FIHC and TDC, we merge the first level nodes to reduce sparseness of this level.

TDC uses a heuristic to compute pair-wise similarities, and at each step, the pair with highest similarity is merged in a way similar to agglomerative clustering. This heuristic uses the number of common documents between nodes as the primary goodness criteria. We found this heuristic problematic, as it does not support hard clustering (i.e. MAX_DOC_DUP = 1 results in no common docs between nodes), and does not consider the actual similarities between clusters. FIHC, on the other hand, applies agglomerative clustering on first level nodes and uses a similarity function similar to the one it uses for selecting parents during hierarchy construction. This function uses the notion of "cluster frequent items" and inspects the documents assigned to each node, and all of its children to find these items, making it very expensive.

We first prepare conceptual documents for first-level nodes by merging term frequencies of large 1-itemsets from all applicable documents in the cluster. Unlike FIHC, which prepares conceptual documents for nodes at all levels, we do it only for first-level

```
for (i = k; i >= 1; i--) do begin
    forall itemsets $t \in I_i$ do begin
        if (document-count($bitmap_t$) > 0 .OR. contains($parents_{i+1}$, t) then
            if (i = 1) then
                add($children_{root}$, t);
            else
                S = get-k-subsets(t)
                sorted_list = $\Phi$
                forall itemsets super_item $\in$ S do begin
                    if (contains($I_{i-1}$, super_item)) then
                        interestingness_val = apply-measure(super_item, t - super_item)
                        add(sorted_list, super_item, interestingness_val)
                    end
                end
                for (j = 0; j < MAX_NODE_DUP .AND. size(sorted_list) >= j; j++) do begin
                    itemset = get-itemset($I_{i-1}$, $sorted\_list_j$)
                    add($children_{itemset}$, t)
                    if (.NOT. contains($parents_i$, itemset)) then
                        add($parents_i$, itemset)
                    end
                end
            end
        else
            prune(t)
        end
    end
end
```

Figure 4. Hierarchy construction

nodes, which is significantly less expensive. We applied bisecting k-means, using the I_2 criterion function on these conceptual document vectors, reducing the computational complexity of this step from $O(n^2 * \log(n))$ to $O(e * \log(k))$, where n is the number of first-level nodes, and e is the number of non-zero entries in the feature vectors of all conceptual documents. Note that applying bisecting k-means on the conceptual document vectors of first-level nodes is significantly less expensive than applying bisecting k-means on all document vectors in the data set, making this approach more scalable than state of the art approaches including bisecting k-means (Section 5.9).

5. Experimental Evaluation

We performed extensive experiments on nine standard datasets of varying characteristics (Table 5) and compared our approach against state of the art agglomerative (UPGMA), partitional (bisecting k-means with I_2 criterion function), frequent itemset based (FIHC) and closed frequent itemset based (TDC) approaches using multiple hierarchical clustering evaluation matrices. With an exception of *Reuters* [19], all datasets can be found as part of the Cluto clustering toolkit [21], which was also used to generate clustering solutions for UPGMA and bisecting k-means, and to merge our top-level conceptual document vectors. For *Reuters*, we did not remove documents assigned to multiple categories but removed documents without category assignment.

Table 5. Datasets used in our experiments

Dataset	*Source*	*#docs*	*#classes*	*#attrs*
Hitech	San Jose Mercury News	2301	6	13170
Re0	Reuters-21578	1504	13	2886
Wap	WebACE	1560	20	8460
Classic4	SMART Project	7094	4	41681
Reuters	Reuters-21578	10787	90	19127
LA12	TREC	6279	6	31472
Ohscal	Ohsumed-233445	11162	10	11465
K1a	WebACE	2340	20	13879
K1b	WebACE	2340	6	13879

5.1. Parallelization

A large percentage of modern computer systems contain multiple processors, processors with multiple cores, or processors that offer hyper-threading capabilities. We utilized SIMD parallelism to take advantage of these features, and to increase run-time performance of various steps used in the clustering process. The "closed interesting" itemset mining algorithm was extended by creating N threads with IDs 0 to N-1, with each thread using its ID to independently explore a subset of possible candidate itemsets, without requiring any intra-step synchronization. This was achieved by simply replacing line 3 of find-interesting-itemsets method with:

```
for (i = thread ID; i < size(super_items); i = i + N) do begin
```

A barrier was added between lines 4 and 5 of CSII-MINE to wait for all threads to finish, and append() on the next line was called on all threads.

Similarly, each thread handled a subset of itemsets at each level of inner termset removal, with updates to the current level's bitmap synchronized. For constraining maximum document duplication, each thread independently handled a level as there are no inter or intra-step dependencies in this step. We parallelized the hierarchy generation step by having each thread handle a subset of itemsets at each level, with updates to parent nodes synchronized. Finally, feature vectors for individual first-level nodes were generated in parallel by a number of threads.

5.2. Evaluation matrices

We used two standard hierarchical clustering evaluation matrices to compare the quality of clustering results produced by our approach with other, state of the art approaches. The first measure, *FScore*, evaluates the overall quality of hierarchical tree using a small number of its nodes [5]. On the other hand, *Entropy* takes into account the distribution of documents in all nodes of the tree. We used *FScore* and *Entropy* as defined by Zhao and Karypis [5]. Note that FIHC and TDC also used *FScore* in the same way.

5.3. Setting the initial support threshold for first-level itemsets

A major issue with any support-based approach, like FIHC [3], is to find the optimal support threshold. Yu et al. [7] proposed to dynamically probe the support threshold by starting with a high value and decreasing the threshold until full coverage of the dataset is achieved. We believe that this approach is problematic, as even a single noisy document would cause this approach to determine 'zero' as support threshold. We addressed this issue by applying a very low support threshold (i.e. 0.2% for the three largest, and 1% for all the other datasets used in this paper) to generate first level itemsets and generating a "miscellaneous" top-level node containing documents not represented by any itemset for visualization purposes. Typically, the number of such documents is very small (i.e. less then 0.1% of the dataset). Note that the support threshold is not used beyond the first level (i.e. to find individual frequent words), and second level and higher itemsets use more principled statistical interestingness measures.

5.4. Using cross validation to determine thresholds for interestingness measures

As explained in sections 2 and 3.3, our approach uses an "interestingness threshold" to prune itemsets and to select parent nodes while assembling the hierarchy. This threshold heavily impacts both the efficiency, and the quality of clustering, which makes it the most important parameter in our system. While it is often possible to tune parameters and achieve good results on individual datasets, it causes the problem of over-fitting, and has little practical value. One of our most important goals was to find measures, and corresponding threshold values that could generalize well and work on datasets with varying characteristics. We achieved this goal by applying global cross-validation, i.e. by randomly selecting a dataset, and trying a number of threshold values for each interestingness measure. The value that resulted in best results on the randomly selected dataset was blindly used across all datasets. In addition, since the Chi-Square test is known to depend on the number of transactions in the dataset, and to overestimate the interestingness of itemsets in large datasets [10], we used a simple heuristic to calculate the Chi-Square threshold values for each of the datasets used in our experiments:

$$chiSquareThreshold = unit * (1 + ceil(\frac{dataset\_size}{p}))$$

This heuristic results in a minimum threshold value of (2 * *unit*) which linearly increases in *unit* increments for each *p* documents. In order to maintain consistency, a number of values for *unit* and *p* were applied on our randomly selected dataset and the values that resulted in a threshold that produced best results on the selected dataset were used to produce Chi-Square thresholds for all other datasets. Table 1 presents the threshold values obtained using this procedure, for all measures. We used these values throughout our experiments.

5.5. Setting values for MAX_DOC_DUP and MAX_NODE_DUP

Table 7. ***FScore*** **comparison of state of the art hierarchical document clustering approaches with "closed interesting" itemset based hierarchical document clustering, using top 6 interestingness measures**

	UPGMA	bi k-means with I_2	FIHC	TDC	Mutual Information	Conviction	Certainty Factor	Added Value	Chi-Square	Yule'sQ
Hitech	0.499	0.561	0.458	***0.57***	0.540	0.559	0.541	0.531	0.533	0.498
Re0	0.584	0.590	0.529	0.57	0.672	0.641	***0.701***	0.621	0.593	0.614
Wap	0.640	0.638	0.391	0.47	***0.663***	0.619	0.626	0.628	0.634	0.618
Classic	0.848	0.764	0.623	0.61	***0.880***	0.817	0.786	0.793	0.802	0.781
Reuters	0.729	0.793	0.506	0.46	***0.851***	0.771	0.783	0.815	0.775	0.836
LA12	0.700	***0.741***	0.432	N/A	0.661	0.616	0.626	0.709	0.617	0.669
Ohscal	0.399	0.493	0.325	N/A	0.530	0.515	0.507	0.509	***0.547***	0.485
K1a	0.646	0.634	0.398	N/A	***0.654***	0.610	0.626	0.639	0.638	0.622
K1b	0.892	0.890	0.768	N/A	***0.903***	0.869	0.876	0.879	0.881	0.890

Table 8. ***Entropy*** **comparison of state of the art hierarchical document clustering approaches with "closed interesting" itemset based hierarchical document clustering, using top 6 interestingness measures**

	UPGMA	bi k-means with I_2	FIHC	TDC	Mutual Information	Conviction	Certainty Factor	Added Value	Chi-Square	Yule'sQ
Hitech	0.262	0.236	1.258	N/A	0.172	0.210	0.200	0.236	0.153	***0.142***
Re0	0.136	0.136	1.239	N/A	0.077	0.098	0.095	0.117	0.133	***0.064***
Wap	0.131	0.131	1.561	N/A	***0.047***	0.052	0.048	0.067	0.056	0.054
Classic	0.074	0.069	0.886	N/A	0.025	0.069	0.063	0.073	0.029	***0.014***
Reuters	0.101	0.086	1.853	N/A	0.155	0.158	0.149	0.165	0.116	***0.084***
LA12	0.151	0.134	1.076	N/A	***0.062***	0.109	0.102	0.091	0.076	0.072
Ohscal	0.279	0.232	1.775	N/A	0.237	0.300	0.288	0.322	0.230	***0.106***
K1a	0.129	0.126	1.645	N/A	0.045	0.058	0.056	0.077	***0.044***	0.063
K1b	0.043	0.042	0.544	N/A	0.042	***0.033***	0.036	0.056	0.042	0.049

MAX_DOC_DUP controls the maximum document duplication at their most specific level, as explained in section 3.2. Note that the documents are already removed from all, but their most specific level because of inner termset removal (section 3.1). Similarly, MAX_NODE_DUP controls the maximum number of parent nodes allowed. TDC [7] uses a parameter similar to MAX_DOC_DUP, with a value of 10, and does not impose any restrictions on the number of parent nodes. We experimentally found that this approach helps boosting the *FScore*, but degrades the overall clustering quality (i.e. *Entropy*) because of too much duplication. Therefore, we used a value of 2 for both of these parameters, allowing soft clustering, and avoiding unnecessary duplication.

5.6. Clustering quality comparison

Tables 7 and 8 compares the clustering quality of our "closed interesting" itemset based hierarchical document clustering approach against state of the art approaches in terms of *FScore* and *Entropy*. To ensure a fair comparison, we executed our approach with each of the interestingness measures exactly once and recorded the results. The same approach was followed to obtain results for UPGMA and bisecting k-means.

For FIHC, we executed the software several times on each dataset with a number of support thresholds and recorded the best results. We noticed that support thresholds that worked best on one dataset resulted in low-quality clustering on other datasets. In several cases, applying the same threshold results in itemset mining to take an indefinite amount of time. As an example, support threshold of 3% resulted in best *FScore* on *Classic*. When the same support threshold was applied on *LA12*, it resulted in 100,000+ frequent 1, 2, and 3-itemsets, after which the itemset mining did not return for 10+ minutes and the application had to be manually terminated. Since TDC implementation was not available, we used results from [7].

Furthermore, Cluto generates both hierarchical and flat clustering solutions for UPGMA and bisecting k-means. The hierarchical clustering solution does not change with the number of desired clusters, which only impacts the flat clustering solution in a way that the desired number of flat clusters are obtained from the hierarchical tree using cluster analysis techniques. Existing frequent itemset based approaches [3, 7] seem to have compared their hierarchical solutions with flat clustering solutions obtained for UPGMA and bisecting k-means, using 3, 15, 30 and 60 as desired number of clusters. For itemset-based approaches, the number of desired clusters is less significant as it only represents the number of top-level nodes in the hierarchy, and not the total number of clusters in the solution. We used the hierarchical clustering solutions for UPGMA and bisecting k-means instead, and observed that they perform better than both of the existing frequent itemset based approaches. The *FScores* we obtained are also closer to the *FScores* reported by Zhao and Karypis [4] on the same datasets.

For the reason of space, we only report results of the top six measures, as determined by averaging the *FScores* and *Entropies* of each measure on all nine datasets. Note that some measures that are not included in Tables 7 and 8 performed very well on few datasets, but failed to generalize when the same interestingness threshold was applied on other datasets. As an example, jMeasure with threshold from Table 1 resulted in *FScore* of 0.584 on *Hitech*, and *Entropy* of 0.061 on *LA12*. Even though these results are better than all approaches we experimented with, the same threshold did not perform as well on other datasets. Our results (Table 7) indicate that Mutual Information results in best overall *FScore*, followed by Added Value and Chi-Square. On the other hand, Yule's Q results in best overall *Entropy* (Table 8) followed by Mutual Information and Chi-Square. We conclude that Mutual Information offers the best balance as it outperforms all existing approaches (and interestingness measures using our approach) in terms of *FScore* on five out of nine datasets, and performs better than existing approaches on seven out of nine datasets in terms of *Entropy*.

5.7. Comparison of "closed interesting" itemsets with closed frequent itemsets

We compared "closed interesting" itemsets against closed frequent itemsets, by mining closed frequent itemsets at various support levels on *Reuters* dataset, and applying our clustering process on the mined itemsets. When an interestingness measure is used to mine itemsets, the hierarchy generation process uses the same measure for parent selection, as explained in section 3.3. For closed frequent itemsets, we used support for this purpose in a way that up to MAX_NODE_DUP parents that share the most documents with the child node (i.e. parent nodes with lowest support) were selected. We found that this approach achieves better *FScores* as compare to TDC [7], which also uses closed frequent itemsets.

Using the support thresholds that were used to generate closed frequent itemsets, we generated large 1-itemsets and used them to mine "closed interesting" itemsets using few of our top measures. Note that the interestingness thresholds remained constant (i.e. as defined in table 1). The resulting itemsets were used to cluster the *Reuters* dataset. We report the number of level 2 and higher itemsets generated, along with the corresponding *FScores*, for closed frequent itemsets and each of the measure used to generate "closed interesting" itemsets. We omitted the number of 1-itemsets because it remains same for both closed frequent itemsets and "closed interesting" itemsets, when the same minimum support threshold is used.

Table 9. The smaller number, but improved performance of "closed interesting" itemsets over closed frequent itemsets, at various support levels

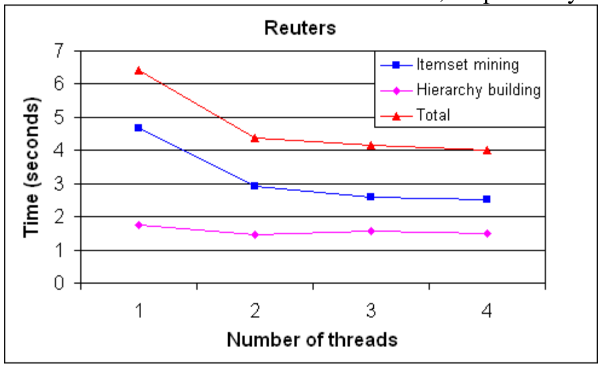
Min Supp	***Closed Frequent***		***Mutual Information***		***Yule's Q***		***Added Value***	
	#	***F***	***#***	***F***	***#***	***F***	***#***	***F***
1%	92880	0.71	1613	0.83	2445	0.80	836	0.78
2%	12246	0.67	842	0.80	548	0.78	397	0.78
3%	4015	0.67	435	0.76	209	0.75	235	0.75
4%	1792	0.64	308	0.70	146	0.68	170	0.71
5%	933	0.62	231	0.68	109	0.68	135	0.69

Table 9 presents the results of this experiment. Clearly, the number of "closed interesting" itemsets found at all support levels is significantly smaller than the number of closed frequent itemsets. Even so, they achieved better *FScores*. Also, the quality of clustering decreases for all itemset types, as the minimum support threshold increases, adding credence to the claim that higher support thresholds result in pruning useful associations [13].

5.8. Parallel processing and hyper-threading

In order to analyze the impact of parallel itemset mining and hierarchy generation, we performed experiments on a system that contains two hyper-threaded, 2.8 GHz Intel Xeon based processors. Each hyper-threaded processor is seen as two logical processors by the OS, resulting in a total of four processors available for executing programs. We started with a single thread and executed the clustering process on two largest datasets (i.e. *Reuters* and *Ohscal*) used in our experiments, a number of times, and averaged the execution times. The same process was repeated with number of threads set to 2, 3 and 4. Note that our run-time environment (i.e. 64-bit Java) mapped individual threads to separate processors.

Figure 5 presents results of this experiment. Using four threads resulted in an average total speedup (computed as the ratio of old and new execution times) of 60% on *Reuters* and 67% on *Ohscal* dataset, when compared with the corresponding single-threaded solutions. Itemset mining enjoyed the most significant performance improvement as threads were added, because of no intra-step synchronization requirements. On the other hand, hierarchy generation performance improved only when a new thread could map to a separate physical processor (i.e. from one thread to two threads) and decreased if more threads were added, requiring execution on a logical processor, because of intra-step synchronization on node modifications, and bitmap updates. This suggests that using a different number of threads for each of these steps could result in better overall performance. Finally, comparing the performance of two-threaded solution with four-threaded solution, we can see that hyper-threading resulted in an average itemset mining speedup of 15% and 13% on *Reuters* and *Ohscal* datasets, respectively.



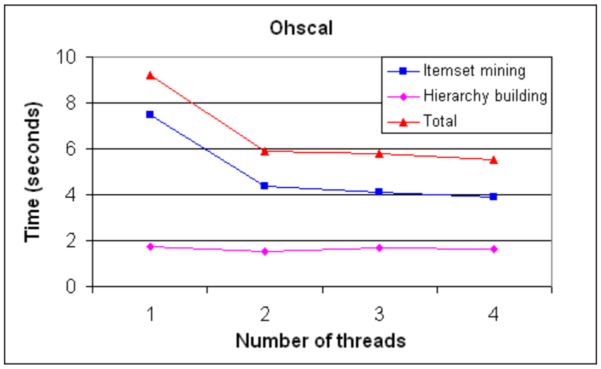


Figure 5. Impact of parallel processing on Reuters and Ohscal datasets with Mutual Information as interestingness measure, and threshold as in Table 1

5.9. Runtime performance and scalability

We used the full *Ohsumed* [20] collection (34,389 unique documents, and 36,250 unique attributes) to evaluate the run-time performance and scalability of our clustering approach. The *Ohsumed* collection was used to generate ten datasets, containing 20K to 200K documents in 20K increments. Each of these datasets was generated by selecting N documents randomly (where N is the size of desired dataset) from existing documents, and replacing approximately 40% of words with other words from the corpus, retaining the frequencies of replaced words. Using Mutual Information as interestingness measure and the threshold value from Table 1, we executed both the parallel (using 4 threads), and single-threaded versions of our clustering process, and also executed bisecting k-means, and FIHC on these datasets. For FIHC, we used the support threshold that resulted in best *FScore* on the full *Ohsumed* collection. In order to ensure a fair comparison, we turned off all cluster analysis, and output options for bisecting k-means, and excluded I/O and reporting times. In addition, the reported times of our approach include execution times of all steps, except offline preprocessing to form document vectors and bitmaps.

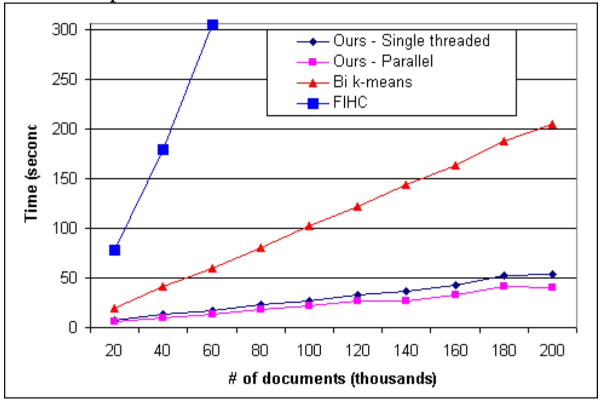


Figure 6. Runtime performance and scalability comparison of our clustering approach, with bisecting k-means, and FIHC

Figure 6 presents results of this experiment. We found that bisecting k-means scaled up linearly, and FIHC scaled worse than linearly, possibly because of frequently accessing document vectors, and agglomerative merging of top-level nodes. The parallel version of our approach outperformed the single-threaded version, as expected. Both versions of our approach scaled sub-linearly, because of significant dimensionality reduction achieved by using "closed interesting" itemsets for clustering, and because our approach reduces the need to refer to full document vectors. These vectors are referred only once: i.e. to generate large 1-itemsets. All interesting k (where k >= 2) itemsets are generated using the bitmaps of large 1-itemsets, and most documents are clustered without ever referring back to the document vectors. The number of such documents increases with the size of the dataset, as our clustering process primarily uses itemsets for forming clusters, and number of words in the corpus does not linearly increase with new documents. Partial vectors (i.e. applicable frequent 1-itemsets) of a small percentage of documents are referred to remove document duplication from clusters at the same level, and to generate conceptual documents for first level nodes. Finally, we expect our approach to scale better than TDC, because the number of "closed interesting" itemsets is significantly smaller than closed frequent itemsets (section 5.7), and other optimizations made throughout the clustering process (i.e. using bisecting k-means to merge the first-level nodes).

6. Conclusions and Future Work

We introduced the notion of "closed interesting" itemsets in this paper. Utilizing the closeness property of closed frequent itemsets, and using ideas from selecting the most interesting association rules, these itemsets provide significant dimensionality reduction over closed frequent itemsets. Using these itemsets, we proposed a new hierarchical document clustering approach that outperforms state of the art approaches, both in terms of *FScore* and *Entropy* on a large number of standard datasets. In addition, our approach scales sub-linearly and was able to cluster 200K documents in less than a minute. A parallel version of our approach achieved the same task in around 40 seconds. We used a principled threshold identification technique and showed that a small number of statistically inspired interestingness measures generalize well to a large number of datasets, without requiring parameter tuning.

We believe that "closed interesting" itemsets can serve as a superior replacement for frequent and closed frequent itemsets, in a variety of application domains. We also believe that existing research in optimized frequent, and closed frequent itemset mining (i.e. FP-Trees) can help improve the performance of mining "closed interesting" itemsets.

In the future, we plan to apply "closed interesting" itemsets in more application domains and work on optimizing various steps used in our clustering process (i.e. parallel bisecting k-means for merging top-level nodes). We also plan to work on finding a more principled way of identifying the Chi-Square thresholds.

7. Acknowledgements

The authors would like to thank Apostol (Paul) Natsev of IBM Research for his valuable suggestions to improve this work. We would also like to thank Prof. Karypis, and Benjamin Fung, for making Cluto and FIHC available, and patiently answering our queries.

8. References

[1] M. Steinbach, G. Karypis, and V. Kumar, "A comparison of document clustering techniques", In *KDD Workshop on Text Mining,* SIGKDD'00, 2000.

[2] L. Kaufman, and P. J. Rousseeuw, "Finding Groups in Data: An Introduction to Cluster Analysis", New York: John Wiley & Sons, Inc, March 1990.

[3] B. Fung, K. Wang, and M. Ester, "Hierarchical document clustering using frequent itemsets", In *Proc. SIAM International Conference on Data Mining,* 2003, pp. 59-70.

[4] Y. Zhao, and G. Karypis, "Evaluation of hierarchical clustering algorithms for document datasets", In *Proc. International Conference on Information and Knowledge Management*, November 2002, pp. 515-524.

[5] Y. Zhao, and G. Karypis, "Hierarchical Clustering Algorithms for Document Datasets", *Data Mining and Knowledge Discovery*, Vol. 10, No. 2, 2005, pp. 141-168.

[6] F. Beil, M. Ester, and X. Xu, "Frequent term-based text clustering", In *Proc. International Conference on Knowledge Discovery and Data Mining*, 2002, pp. 436-442.

[7] H. Yu, D. Searsmith, X. Li and J. Han, "Scalable Construction of Topic Directory with Nonparametric Closed Termset Mining", In *Proc. Fourth IEEE International Conference on Data Mining (ICDM'04)*, 2004, pp. 563-566.

[8] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generations", In *Proc. ACM SIGMOD Int. Conf. Management of Data (SIGMOD'00)*, 2000.

[9] F. Berzal, I. Blanco, D. Sánchez and M.A. Vila, "Measuring the Accuracy and Importance of Association Rules: A New Framework", *Intelligent Data Analysis*, 2002.

[10] T. Brijs, K. Vanhoof, and G. Wets, "Defining interestingness for association rules", In *Int. journal of information theories and applications, 10:4,* 2003.

[11] B. Liu, W. Hsu, and Y. Ma, "Pruning and Summarizing the Discovered Associations", *Proceedings of the SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, 1999.

[12] P. Tan, and V. Kumar, "Interestingness measures for association patterns: A perspective", *KDD Workshop on Postprocessing in Machine Learning & Data Mining*, 2000.

[13] P. Tan, V. Kumar, and J. Srivastava, "Selecting the right interestingness measure for association patterns", In *Proc. of SIGKDD. 32–41*, 2002.

[14] Robert J. Hilderman, and Howard J. Hamilton, "Knowledge Discovery and Interestingness Measures: A Survey", *University of Regina Technical Report, TR 99-04, ISBN 0-7731-0391-0*, 1999.

[15] J. Li, and Y. Zhang, "Direct Interesting Rule Generation", In *Proc. Third IEEE International Conference on Data Mining (ICDM'03),* 2003, *p. 155*.

[16] E. Shortliffe, and B. Buchanan, "A model of inexact reasoning in medicine", *Mathematical Biosciences 23*, 1975.

[17] P. Smyth, and R. M. Goodman, **"**An information theoretic approach to rule induction from databases**",** *IEEE Transactions on Knowledge and Data Engineering,* 1992.

[18] A. J. Viera, and J. M. Garrett, "Understanding Interobserver Agreement: The Kappa Statistic", *Family Medicine 37(5)*, 2005, p. 360.

[19] Reuters. http://kdd.ics.uci.edu/databases/reuters21578.

[20] Ohsumed. ftp://medir.ohsu.edu/pub/ohsumed.

[21] Cluto. http://glaros.dtc.umn.edu/gkhome/views/cluto.