SHCoClust, a Scalable **S**imilarity-based **H**ierarchical **Co-clust**ering Method and its Application to Textual Collections

Xinyu Wang
Université de Lyon
Lyon 2, ERIC, EA3083
5, av. Pierre Mendès-France
69500 Bron, France
Email: xinyu.wang@univ-lyon2.fr

Julien Ah-Pine
Université de Lyon
Lyon 2, ERIC, EA3083
5, av. Pierre Mendès-France
69500 Bron, France
Email: julien.ah-pine@univ-lyon2.fr

Jérôme Darmont
Université de Lyon
Lyon 2, ERIC, EA3083
5, av. Pierre Mendès-France
69500 Bron, France
Email: jerome.darmont@univ-lyon2.fr

***Abstract*—In comparison with flat clustering methods, such as K-means, hierarchical clustering and co-clustering methods are more advantageous, for the reason that hierarchical clustering is capable to reveal the internal connections of clusters, and co-clustering can yield clusters of data instances and features. Interested in organizing co-clusters in hierarchies and in discovering cluster hierarchies inside co-clusters, in this paper, we propose SHCoClust, a scalable similarity-based hierarchical co-clustering method. Except possessing the above-mentioned advantages in unison, SHCoClust is able to employ kernel functions, thanks to its utilization of inner product. Furthermore, having all similarities between 0 and 1, the input of SHCoClust can be sparsified by threshold values, so that less memory and less time are required for storage and for computation. This grants SHCoClust scalability, i.e, the ability to process relatively large datasets with reduced and limited computing resources. Our experiments demonstrate that SHCoClust significantly outperforms the conventional hierarchical clustering methods. In addition, with sparsifying the input similarity matrices obtained by linear kernel and by Gaussian kernel, SHCoClust is capable to guarantee the clustering quality, even when its input being largely sparsified. Consequently, up to 86% time gain and on average 75% memory gain are achieved.**

I. Introduction

As an important branch of unsupervised methods, clustering techniques always succeed in drawing interests of researchers in various domains. Clustering data in different manners reveals different information conveyed by the data. For example, hierarchical clustering groups data instances in an iterative fashion and outputs a binary-tree like structure called dendrogram, which explicitly displays the internal connections among sub-clusters. However, the process of generating a dendrogram is computationally costly. Differently, co-clustering is a technique of subspace clustering, instead of grouping only data instances, it performs clustering in both data space and feature space. In a resulted co-cluster, a partition of data instances is described by a partition of features.

Interested in discovering hierarchical structures inside co-clusters and in attenuating the influence of high complexity of hierarchical clustering, in this paper we propose a similarity-based hierarchical co-clustering algorithm, SHCoClust. It is built upon a bipartite spectral co-clustering method [1], and a recent approach of the agglomerative hierarchical clustering [2], which allows sparsifying the proximity matrix to reduce computational cost. SHCoClust is capable to organize co-clusters in a form of dendrogram, and to discover cluster hierarchies inside co-clusters. We believe this property is valuable in understanding the content of data. However, none of the methods that contributes to establish SHCoClust is able to do so. Furthermore, with inner product at base, SHCoClust can employ kernel functions, which offers it the capability to process non-linearly separable datasets more effectively. To our knowledge, SHCoClust is the first method that is able to reveal cluster hierarchies of and inside co-clusters, and to apply kernel functions in co-clustering tasks.

Through this paper, we begin with introducing the methods that we use to establish SHCoClust in Section II. After detailing our approach in Section III, we illustrate the experiments and results in Section IV. Section V concludes our current work and presents further works.

II. State of the Art

A. The Bipartite Spectral Co-clustering

Since introduced in [3], co-clustering has been widely applied in the domains of text mining and bio-informatics. Capable to simultaneously partition data instances and their features, co-clustering is favored in searching for the most representative terms for a set of closely grouped documents, or in searching for the most expressive conditions for a set of highly related genes. Details of many published works on co-clustering can be found in [4], [5], [6].

Based on bipartite spectral graph partitioning, [1] models a collection of documents as an undirected bipartite graph, in which documents and terms are vertices and TF-IDF weights are edges. Discovering co-clusters of documents and terms is to find the best cuts in this graph, eventually resulting in K subgraphs, in which vertices are highly associated and are of equally-sized. To solve this NP-hard discrete optimization problem, an effective heuristic method, the spectral graph

partitioning, is applied to offer a real relaxation. The proposed solutions are the second largest left and right singular vectors of the diagonal-normalized document-term matrix. For convenience, we refer this approach as BSC (bipartite spectral co-clustering).

B. The Agglomerative Hierarchical Clustering

There are two principle techniques of hierarchical clustering, the divisive (DHC) and the agglomerative (AHC). Given a dataset $\mathcal{D}$ of n instances and m features, DHC begins with splitting $\mathcal{D}$ into two sets iteratively and results in clusters of which each is a data instance. AHC performs in the opposite way, it starts with merging two individual instances and eventually outputs a cluster of size n. They output a binary tree-like structure of $2n-1$ nodes, named dendrogram. Comparing these two techniques in terms of complexity, AHC is practically preferred as it is of $\mathcal{O}(N^3)$ in the worst case. However, finding an optimal split in DHC can be NP-hard, if n is large. [7] surveys many works that improve the performance of AHC. In the scope of this paper, we focus on the conventional AHC and its methods. The computing procedure of AHC is shown in Algorithm 1, where $d(,)$ is a distance metric, and each data instance is initially considered as a cluster, represented by C.

Algorithm 1 Conventional AHC

Require: dataset $\mathcal{D}_{n\times m}$. $\forall x \in [1,\ldots,n], C_x \leftarrow \mathcal{D}_x$
1: distance matrix $\mathbb{D} \leftarrow \emptyset$
2: **for i** $= 2$ to n **do**
3: **for j** $= 1$ to **i** $- 1$ **do**
4: $\mathbb{D}_{\mathbf{ij}} \leftarrow d(\mathcal{D}_{\mathbf{i}}, \mathcal{D}_{\mathbf{j}})$
5: **end for**
6: **end for**
7: **while** num_clusters> 1 **do**
8: $(i,j) \leftarrow \arg min \mathbb{D}_{x,y}$
9: $C_{ij} \leftarrow C_i \cup C_j$
10: **for all** $C_k \in \mathcal{D}$ such that $C_k \neq C_i, C_k \neq C_j$ **do**
11: $\mathbb{D} \xleftarrow{update} d(C_{ij}, C_k)$
12: **end for**
13: **end while**

In the setting of Vector Space Model and the TF-IDF weighting scheme, documents can have a sphere-like projection in the feature space, in such case the Euclidean distance is a suitable choice to compute the distance of two document vectors. In AHC, an essential step, shown as line 11 in Algorithm 1, is to determine $d(C_{ij}, C_k)$. There are seven methods to do so by using either the centroids or the graphic representatives of clusters. The Lance-Williams (LW) formula is a framework that unifies the seven conventional clustering methods [8]. Equation 1 and Table I display the formula and its parameter values. $|C_x|$ is the number of data instances in cluster C_x. With this formula, one can simply switch parameter values to compute $d(C_{ij}, C_k)$ for a given method.

$$\begin{aligned} d(C_{ij}, C_k) =& \alpha_i d(C_i, C_k) + \alpha_j d(C_j, C_k) + \beta d(C_i, C_j) \\ &+ \gamma |d(C_i, C_k) - d(C_j, C_k)|. \end{aligned} \quad (1)$$

TABLE I
LANCE-WILLIAMS FORMULA: METHODS AND PARAMETER VALUES

Methods	α_i	α_j	β	γ
Single	1/2	1/2	0	-1/2
Complete	1/2	1/2	0	1/2
Average	$\frac{\lvert C_i\rvert}{\lvert C_i\rvert+\lvert C_j\rvert}$	$\frac{\lvert C_j\rvert}{\lvert C_i\rvert+\lvert C_j\rvert}$	0	0
Weighted	1/2	1/2	0	0
Centroid	$\frac{\lvert C_i\rvert}{\lvert C_i\rvert+\lvert C_j\rvert}$	$\frac{\lvert C_j\rvert}{\lvert C_i\rvert+\lvert C_j\rvert}$	$-\frac{\lvert C_i\rvert\lvert C_j\rvert}{(\lvert C_i\rvert+\lvert C_j\rvert)^2}$	0
Median	1/2	1/2	-1/4	0
Ward	$\frac{\lvert C_i\rvert+\lvert C_k\rvert}{\lvert C_i\rvert+\lvert C_j\rvert+\lvert C_k\rvert}$	$\frac{\lvert C_j\rvert+\lvert C_k\rvert}{\lvert C_i\rvert+\lvert C_j\rvert+\lvert C_k\rvert}$	$-\frac{\lvert C_k\rvert}{\lvert C_i\rvert+\lvert C_j\rvert+\lvert C_k\rvert}$	0

C. Sim_AHC, the Similarity-based AHC

A new expression of the LW formula is proposed in [2], where distances are replaced by similarities, we refer this approach as Sim_AHC. It is mathematically and experimentally proved to be equivalent to the original LW formula. Input to Sim_AHC is the pairwise similarity matrix $\mathbb{S}$ of a normalized dataset (that each row has norm 1). As $\forall s \in \mathbb{S}, 1 \geq s \geq 0$, $\mathbb{S}$ can be sparsified by a threshold value, so that less memory is demanded to store $\mathbb{S}$ and less time is required to compute on it. This is an advantageous property in terms of efficiency. However, this property cannot be found in the conventional AHC. Because when distances are used, zero and close-to-zero values signify high similarities, which are of the most interest in clustering, thus they have to be stored for computation instead of being ignored. The similarity updating process between a newly-merged cluster C_{ij} and any other cluster C_k in Sim_AHC consists of two steps, updating the cross-cluster similarity and updating the cluster's self similarity:

$$\begin{aligned} s(C_{ij}, C_k) = \alpha_i s(C_i, C_k) + \alpha_j s(C_j, C_k) + \beta s(C_i, C_j) \\ - \gamma |s(C_i, C_k) - s(C_j, C_k)|. \end{aligned} \quad (2)$$

$$s(C_{ij}, C_{ij}) = \delta_i s(C_i, C_i) + \delta_j s(C_j, C_j). \quad (3)$$

Values of parameters α_i, α_j, β and γ in Sim_AHC are kept the same as in Table I. To guarantee the equivalence for each individual clustering method, the values of newly added parameters δ_i and δ_j are set accordingly: for median method, $\delta_i = \delta_j = \frac{1}{4}$; for centroid method, $\delta_i = \frac{|C_i|^2}{(|C_i|+|C_j|)^2}$, $\delta_j = \frac{|C_j|^2}{(|C_i|+|C_j|)^2}$; and for the other five methods, values of δ_i and δ_j can be determined freely as long as their sum is 1.

III. SHCoCLUST

Co-clusters returned by BSC are of the same level, like clusters in K-means, they are flat. A dendrogram returned by AHC or by Sim_AHC displays how data instances are merged step by step, nevertheless, the connections among data features are unknown. Discovering hierarchies of co-clusters, and exploring connections of sub-clusters inside co-clusters surely provide us a better understanding of our data.

For such purpose, we propose SHCoClust, a similarity-based hierarchical co-clustering method, detailed in Algorithm 2.

Algorithm 2 SHCoClust

Require: dataset $\mathcal{D}_{n\times m}$

1: $\mathbf{R}_{n\times n}(i,i) \leftarrow \sum_{j=1}^{m} \mathcal{D}_{ij}$, $\mathbf{C}_{m\times m}(j,j) \leftarrow \sum_{i}^{n} \mathcal{D}_{ij}$
2: $\mathcal{D}_{norm} \leftarrow \mathbf{R}^{-\frac{1}{2}} \mathcal{D} \mathbf{C}^{-\frac{1}{2}}$
3: $k \leftarrow \lceil \log_2 K \rceil + 1$
4: $\mathbf{U}_{n\times k}, \mathbf{S}_{k\times k}, \mathbf{V^T}_{k\times m} \leftarrow SVD(\mathcal{D}_{norm}, k)$
5: **if** $\mathbf{S}_{11} \geq \cdots \geq \mathbf{S}_{kk}$ **then**
6: $\quad \mathbf{U'}_{n\times(k-1)} \leftarrow \mathbf{U}_{n\times[2,\ldots,k]}$
7: $\quad \mathbf{V'}_{m\times(k-1)} \leftarrow \mathbf{V}_{m\times[2,\ldots,k]}$
8: **else if** $\mathbf{S}_{kk} \geq \cdots \geq \mathbf{S}_{11}$ **then**
9: $\quad \mathbf{U'}_{n\times(k-1)} \leftarrow \mathbf{U}_{n\times[1,\ldots,k-1]}$
10: $\quad \mathbf{V'}_{m\times(k-1)} \leftarrow \mathbf{V}_{m\times[1,\ldots,k-1]}$
11: **end if**
12: $\mathbf{Z}_{(n+m)\times(k-1)} \leftarrow \begin{bmatrix} \mathbf{R}_{n\times n}\mathbf{U'}_{n\times(k-1)} \\ \mathbf{C}_{m\times m}\mathbf{V'}_{n\times(k-1)} \end{bmatrix}$
13: similarity matrix $\mathbb{S} \leftarrow \emptyset$, $\forall x \in [1,\ldots,n+m]$, $C_x \leftarrow \mathbf{Z}_x$
14: $\mathbf{Z'} \leftarrow norm(\mathbf{Z})$ such $\sum_{c=1}^{k-1} \mathbf{Z'}_{rc}^2 = 1, \forall r \in [1,\ldots,n+m]$
15: **for i** $= 1$ to $n+m$ **do**
16: $\quad$ **for j** $= 1$ to **i do**
17: $\quad\quad \mathbb{S}_{\mathbf{ij}} \leftarrow s(\mathbf{Z'_i}, \mathbf{Z'_j})$
18: $\quad$ **end for**
19: **end for**
20: **while** num_clusters> 1 **do**
21: $\quad (i,j) \leftarrow \arg max \mathbb{S}_{x,y}$
22: $\quad C_{ij} \leftarrow C_i \cup C_j$
23: $\quad$ **for all** $C_k \in \mathbf{Z'}$ such that $C_k \neq C_i$ and $C_k \neq C_j$ **do**
24: $\quad\quad \mathbb{S} \xleftarrow{update}$ Equations 2 and 3
25: $\quad$ **end for**
26: **end while**

The above procedure of SHCoClust outputs a dendrogram of $n+m$ leaves and $n+m-1$ internal nodes. If input $\mathcal{D}_{n\times m}$ is a doc-term matrix, the dendrogram grows by merging a pair of doc-doc, term-term or doc-term in each iteration. Cutting the dendrogram by a desired number of clusters K results in flattened co-clusters, each contains a sub-dendrogram of documents and terms. In Section IV-E, visualization of a sampled dataset is presented.

Shown as line 14 in Algorithm 2, $\mathbf{Z}$ is normalized so that if the similarity function $s(,)$ in line 17 takes a form of inner product, values in matrix $\mathbb{S}$ are cosine similarities. The diagonal entries of $\mathbb{S}$ are constantly 1. This nature allows $s(,)$ to be extended to a kernel function, $\mathcal{K}$, such that $\mathcal{K}(x,y) = \langle \phi(x), \phi(y) \rangle$, where $\phi : \mathcal{I} \rightarrow \mathcal{F}$ is a mapping from $\mathcal{I}$ to $\mathcal{F}$, with $dim(\mathcal{I}) \ll dim(\mathcal{F})$, and $x, y \in \mathcal{D}$. To generalize $\mathcal{K}$, we apply $\forall x, y \in \mathcal{D}$: $s(x,y) = \mathcal{K}(x,y)/\sqrt{\mathcal{K}(x,x)\mathcal{K}(y,y)}$. In our experiments, linear kernel and Gaussian kernel are tested. Having $\forall s \in \mathbb{S}, 1 \geq s \geq 0$, given a threshold $\tau \in (0,1)$ on $\mathbb{S}$, we define that $s = 0$ if $s \leq \tau$ to sparsify $\mathbb{S}$. Consequently, less memory is required to store $\mathbb{S}$ and thus less time is needed to compute the while loop of lines 20-26 in Algorithm 2.

IV. Experiments

A. Datasets, Preprocessing and Evaluation Metrics

Three well-known corpora that have been widely tested in text clustering benchmarks are used in our experiments. These collections are Reuters-21578[1] [9], SMART [1] and 20NewsGroups (20NG)[2] [10]. Table II details six experimented datasets that are sampled from the three collections. In the rest of the paper, theses datasets are referred by their indexes shown in column "Ind.". Based on the bag-of-words assumption and the Space Vector Model, each dataset is preprocessed into a doc-term matrix, whose rows are document vectors and columns are terms. A few steps are involved in preprocessing, in detail, stop words are removed, terms that have document frequency higher than 20% and lower than 1% are removed, and TF-IDF weighting strategy is applied. "nb.docs" and "nb.terms" in Table II are the numbers of documents and of terms after preprocessing, "nb.K" is the ground-truth number of clusters. "ARPACK" is used as solver in the function of Singular Value Decomposition that forms $\mathbf{Z}$. ARI (Adjusted Rand Index) [11] and NMI (Normalized Mutual Information) [12] are used to evaluate clustering quality.

TABLE II
Experimented Datasets

Dataset	Ind.	nb.K	nb.docs	nb.terms	Z shape
Reuters	R5	5	500	652	(1150, 3)
	R7	7	2100	2133	(4277, 3)
	R10	10	2450	5075	(7525, 4)
SMART	S0	3	1500	2272	(3772, 2)
	S1	3	3893	6812	(10705, 2)
20NG	NG8	8	3200	1118	(4277, 3)
	NG20	20	2000	1104	(3079, 5)

B. Comparisons of Clustering Quality

Our first experiment consists of four tests to compare clustering quality among conventional AHC, BSC, SHCoClust with and without sparsification. These tests are:

1) SHCoClust without sparsification v.s. conventional AHC
2) SHCoClust with sparsification v.s. without
3) SHCoClust without sparsification v.s. BSC
4) SHCoClust with sparsification v.s. BSC

In tests that concern sparsification, threshold values τ at percentile ranks {10, 25, 50, 75, 90}% of similarities in $\mathbb{S}$ are used to sparsify $\mathbb{S}$. In this experiment, we apply linear kernel to obtain $\mathbb{S}$, which is in fact filled with cosine similarities. The results of this experiment are illustrated in Figure 1. From top to down, each row maps to one test that is marked by its index. Each column lists the corresponding results for a tested dataset. Graph in each cell is a bar chart that is highlighted by a vertical line at x=0. The y-axis in each graph is labeled by the abbreviations of seven clustering methods. Bars in these graphs indicate the "difference" of values for ARI (the red, or the dark bars in a gray-scale printed paper) and for NMI (the

[1]Distribution 1.0, the ApteMod version.

[2]We used the same dataset as in http://qwone.com/~jason/20Newsgroups/.

blue, or the light ones) using a clustering method, obtained by subtraction of two tested algorithms (A v.s. B, $A - B$). In the case of sparsification, we chose the highest value of ARI or NMI obtained in each clustering method through τs, then subtract it with the ARI or NMI value obtained by the approach in comparison.

In Figure 1, we can discovery a few interesting findings:

- Compared to conventional AHC, clustering quality is largely improved using SHCoClust for all clustering methods, except for single link and the Ward method. In datasets of S0 and S1, the increase is tremendous, ARI and NMI are raised up to 0.8 and 0.7 at maximum.
- When sparsification is applied in SHCoClust, the clustering quality is further enhanced, and this enhancement does not compromise SHCoClust's efficiency. Additionally, as most highest ARI and NMI are obtained when τ becomes very close to 1 (values of τ are not shown), the efficiency of SHCoClust is considerably improved, thanks to a substantially sparsified input.
- SHCoClust (without sparsification) obtains close results to BSC for all clustering methods, except for single link. Though improvements can be found in R5, R7 and NG8, other datasets do not display any. However, when sparsification is applied in SHCoClust, some noticeable amelioration can be observed. Comparing graphs of test 3) and of test 4), bars generally exhibit a left-to-right drift, diminishing their heights in the left side of the vertical line at x=0, and growing their heights in the right side. This drift is very apparent in R7 and in NG20 datasets, where ARI and NMI bars are noticeably pulled towards the right side. For other datasets, single link seems to be the least affected.

To summarize, SHCoClust significantly improves clustering quality compared to the conventional AHC for most clustering methods. Without sparsification it obtains close results to BSC. However, when sparsification is applied, SHCoClust demonstrates some obvious improvements over BSC.

C. Extension to Kernel Functions

In our second experiment, we extend the similarity function $s(,)$ in SHCoClust to Gaussian kernel, given by $\mathcal{K}(x, y) = \exp(-\gamma\|x - y\|^2)$ for $x, y \in \mathcal{D}$, $\gamma = 1/dim(\mathcal{I})$ by default[3]. Three representative datasets R5, S0 and NG8 are experimented and are compared to their results obtained by linear kernel. The input kernel matrix is sparsified in the same fashion as in the previous experiment. For each threshold value, ARI, NMI, relative memory usage and relative running time are recorded. Figure 2 illustrates the results of this experiment. Indexed by the clustering methods in column and the names of datasets in row, each graph in Figure 2 contains four lines: the solid line with "+" sign denotes the NMI values, the dashed line with triangle sign is for ARI, while the solid line with "o" sign and the solid line with "x" sign represent the relative memory usage and the relative running time, respectively. The x-axis corresponds to the percentile ranks that define the threshold values τ (not shown). At each $\tau \in (0, 1)$, exact memory usage and running time are recorded, and are divided by those at $\tau = 0$, i.e., when no thresholding is applied, to obtain the relative memory usage and the relative running time.

In Figure 2 we can see that, as the percentile rank increases, memory usage and running time decrease correspondingly, however, ARI and NMI tend to preserve their values at some level until the percentile rank approaches closely to 1. More precisely, before the percentile rank crosses 75%, in most cases, we can obtain ARI and NMI values as high as (or higher than) using the full-sized input. In some other cases, ARI and NMI even boost after percentile rank is over 75%, like in S0 (of linear kernel using average link). Among seven clustering methods, single link is the most peculiar, its ARI and NMI are almost invariant to the effect of sparsification. Exceptionally, in R5 of linear kernel and Gaussian kernel, ARI of single link boosts when percentile rank $= 90\%$, at which point memory usage and running time are largely reduced. Comparing the two kernel functions, overall similar behavior can be observed regarding the four curves. In some cases, Gaussian kernel returns higher metric values than linear kernel, and vice versa. Among the three experimented datasets, results of S0 are globally better, while R5 and NG8 display limited difference in their results.

Principle conclusions we can draw from this experiment are that (1) SHCoClust is capable to guarantee the clustering quality even when its input is significantly sparsified, requiring considerably reduced memory usage and running time. τ at percentile rank of 75% is a heuristically good threshold in our experiment (90% for single link). (2) SHCoClust can be easily employed with a kernel function, though only linear and Gaussian kernels are experimented, we believe that its compatibility with kernel functions enables it to handle non-linearly separable datasets more effectively in other tasks.

D. Discussion of Complexity and Scalability

In terms of storage and time complexity, computation with the input of a full-sized pairwise similarity matrix $\mathbb{S}$ of N instances and features corresponds to storage complexity of $\mathcal{O}(N^2)$ and time complexity of $\mathcal{O}(N^3)$ (lines 15-25 in Algorithm 2). Sparsifying $\mathbb{S}$ with τ results in M non-zero values stored for computation, $M < N^2$, eventually reduces storage complexity to $\mathcal{O}(M)$ and time complexity to $\mathcal{O}(NM)$. The linear relationship between the storage and time complexity is demonstrated by the lines of relative memory usage and running time in Figure 2. Let τ^* define a threshold value, at which clustering quality is preserved as high as possible, meanwhile $\mathbb{S}$ is sparsified to have M as small as possible. As stated previously, τ at percentile rank of 75% is the τ^* for most cases in our experiment. Table III exhibits the conservation of computing resources measured by memory gain and time gain at τ^*, at which highest ARIs are obtained (marked by *).

[3]This default setting is used in popular SVM packages. Besides, when γ is low, Gaussian kernel provides close-to-one values and higher similarities between pairs of points.

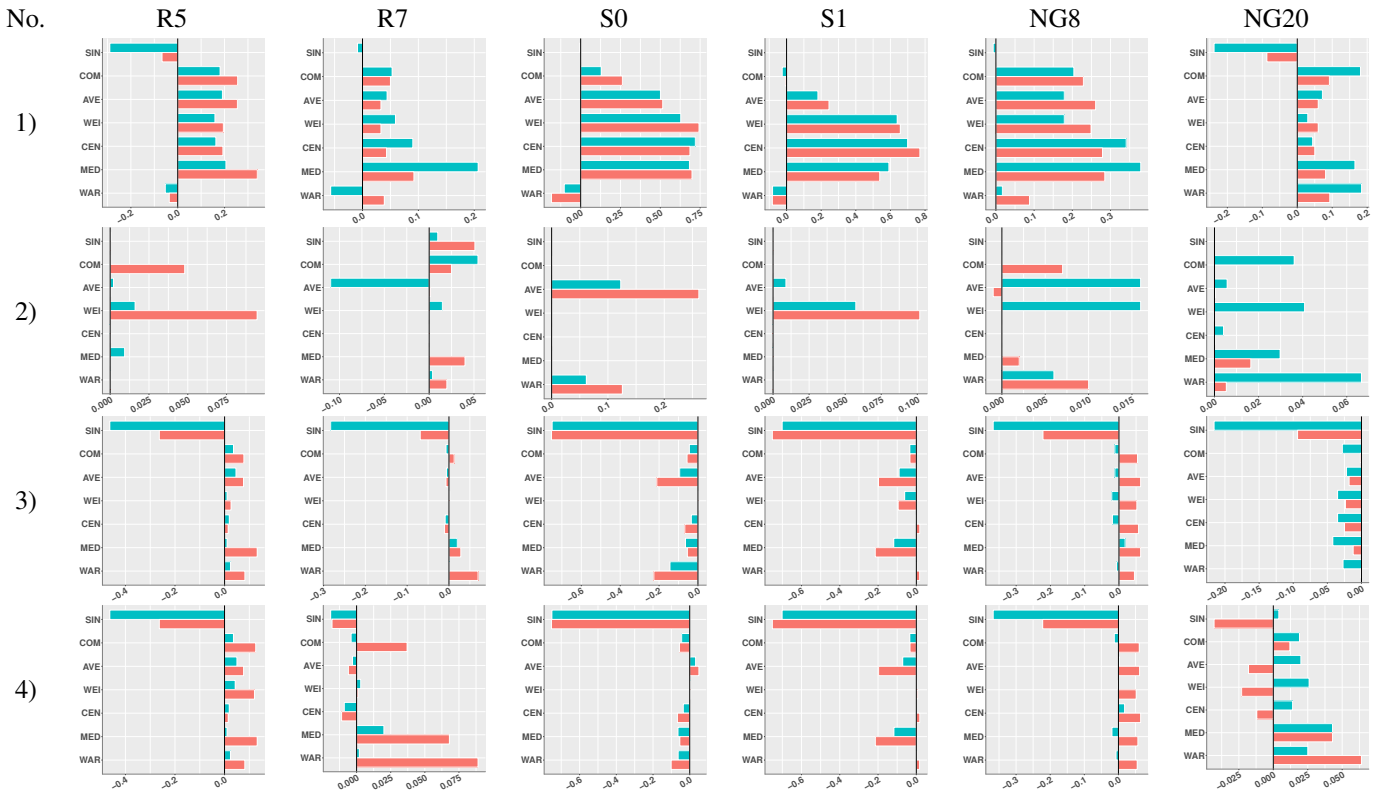

Fig. 1. Comparisons of clustering quality among conventional AHC, BSC, SHCoClust with and without sparsification

The clustering methods that output ARIs* and the ARI values of BSC are listed. Values of $-x$ in Mem% and in Time% indicate the relatively reduced memory usage and running time compared to using a full-sized input.

TABLE III
HIGHEST ARI, RELATIVE GAIN IN MEMORY USAGE AND IN RUNNING TIME

Dataset	Method	τ^* value	Mem%	Time%	ARI*	BSC
R5	Median	0.242	-25	-8	0.395	0.263
R7	Ward	0.998	-75	-61	0.158	0.069
S0	Average	0.996	-90	-86	0.803	0.753
S1	Centroid	0.778	-50	-11	0.770	0.752
NG8	Centroid	0.880	-75	-43	0.287	0.222
NG20	Ward	0.998	-75	-37	0.158	0.094

From this table we can see that all ARIs* are higher than those of BSC, implying that better clustering quality is achieved by SHCoClust. Moreover, ARIs* are obtained with substantially reduced memory usage and running time. Except for R5 dataset, in which median method only gains 8% in time and 25% in memory, other datasets obtain up to 86% time gain and on average 75% memory gain. This proves that SHCoClust is good at economizing computing resources, meanwhile preserving clustering effectiveness. In other words, if given limited computing resources, SHCoClust is able to process relatively large datasets. This reflects its scalability.

E. Exploration of Hierarchical Co-clusters and Structure

We use a sampled dataset $\mathcal{D}_{sam}$ from the SMART collection to illustrate the hierarchies of co-clusters and the connections of sub-clusters inside a co-cluster. $\mathcal{D}_{sam}$ contains 30 documents (10 from each class) and 51 terms (after preprocessing). (a) and (b) of Figure 3 exhibit the doc-term matrix of $\mathcal{D}_{sam}$ before and after being processed by SHCoClust. In (b), we can clearly see that three co-clusters of different size lie along the diagonal. These co-clusters are obtained by the dendrogram cut at a height, shown in (c). Different from the dendrogram obtained by a conventional AHC method, the dendrogram of SHCoClust has negative heights, thus it looks like it grows downwards. In (c), a leave node of the dendrogram is marked by the number of members in it. The three sub-dendrograms above the cutting line (from left to right) map to the bottom-right, the top-left and the middle-situated co-clusters in (b). With the aid of (c), we are able to know that in (b) the bottom-right co-cluster is in fact closer to the top-left one and farther from the middle-situated co-cluster. (d) presents the content of the left-most sub-dendrogram in (c), i.e., the bottom-right co-cluster in (b). Keeping the same structure as it is in the dendrogram, (d) allows us to know which documents and terms co-exit and how they are connected in their co-cluster.

V. CONCLUSIONS AND PERSPECTIVES

In this paper, we proposed a similarity-based hierarchical co-clustering method, SHCoClust. We illustrated its properties, examined its clustering quality, efficiency, discussed its complexity, scalability and visualized its output. We believe that it fills a blank in literature with its unique characteristics.

A future improvement that we intend to do for this method

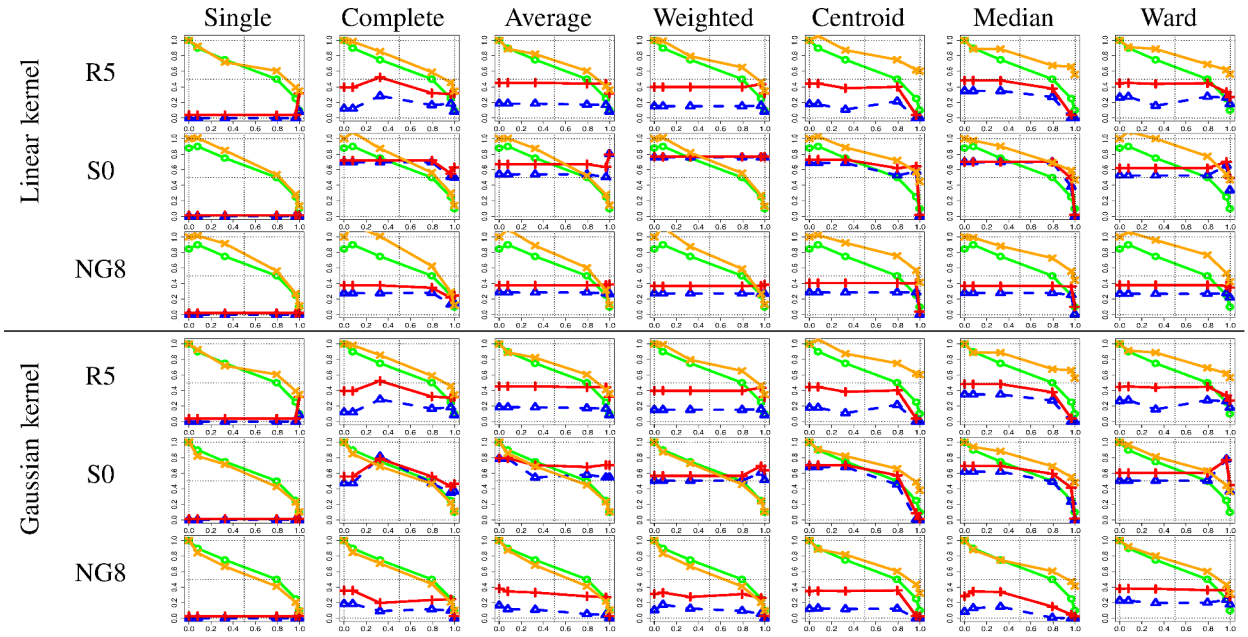

Fig. 2. Results of linear kernel and Gaussian kernel with sparsification

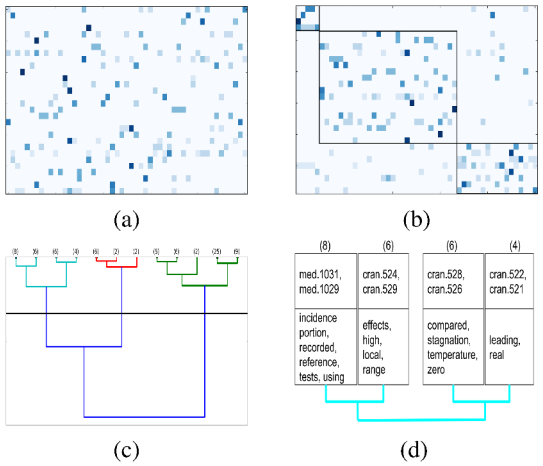

Fig. 3. Doc-term matrix (a) before and (b) after SHCoClust, (c) dendrogram of co-clusters and (d) content of the medium-sized co-cluster

is how to learn a heuristically good threshold value τ in general cases. We would like to find a more efficient way to determine it. In addition, we are interested to test more kernel functions to examine if they present different behaviors from the experimented functions in this paper. Currently, we are working on an implementation of the distributed version of SHCoClust, supported by Apache Spark Engine[4]. We anticipate that further tests with this implementation will enable us to address problems in the scale of big data. For further work, we are also interested to apply SHCoClust to testing the Clustering Hypothesis in the domain of Information Retrieval. We believe that the hierarchical co-clustering structure output by SHCoClust will help us discover interesting facts concerning this topic.

[4]http://spark.apache.org/

ACKNOWLEDGMENT

This work is supported by REQUEST PIA/FSN project.

REFERENCES

[1] I. S. Dhillon, "Co-clustering documents and words using bipartite spectral graph partitioning," in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2001, pp. 269–274.

[2] J. Ah-Pine and X. Wang, "Similarity based hierarchical clustering with an application to text collections," in *International Symposium on Intelligent Data Analysis*. Springer, 2016, pp. 320–331.

[3] J. A. Hartigan, "Direct clustering of a data matrix," *Journal of the american statistical association*, vol. 67, no. 337, pp. 123–129, 1972.

[4] S. C. Madeira and A. L. Oliveira, "Biclustering algorithms for biological data analysis: a survey," *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, vol. 1, no. 1, pp. 24–45, 2004.

[5] A. Tanay, R. Sharan, and R. Shamir, "Biclustering algorithms: A survey," *Handbook of computational molecular biology*, vol. 9, no. 1-20, pp. 122–124, 2005.

[6] S. Busygin, O. Prokopyev, and P. M. Pardalos, "Biclustering in data mining," *Computers & Operations Research*, vol. 35, no. 9, pp. 2964–2987, 2008.

[7] R. Xu, D. Wunsch *et al.*, "Survey of clustering algorithms," *Neural Networks, IEEE Transactions on*, vol. 16, no. 3, pp. 645–678, 2005.

[8] G. N. Lance and W. T. Williams, "A general theory of classificatory sorting strategies ii. clustering systems," *The computer journal*, vol. 10, no. 3, pp. 271–277, 1967.

[9] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," in *European conference on machine learning*. Springer, 1998, pp. 137–142.

[10] K. Lang, "NewsWeeder : learning to filter netnews," in *Proceedings of the Twelfth International Conference on Machine Learning*, 1995, pp. 331–339.

[11] L. Hubert and P. Arabie, "Comparing partitions," *Journal of classification*, vol. 2, no. 1, pp. 193–218, 1985.

[12] A. F. McDaid, D. Greene, and N. Hurley, "Normalized mutual information to evaluate overlapping community finding algorithms," *arXiv preprint arXiv:1110.2515*, 2011.