HAL open science

Similarity Based Hierarchical Clustering with an Application to Text Collections

Julien Ah-Pine, Xinyu Wang

To cite this version:

Julien Ah-Pine, Xinyu Wang. Similarity Based Hierarchical Clustering with an Application to Text Collections. Intelligent Data Analysis, Oct 2016, Stockholm, Sweden. pp.320 - 331, 10.1007/978-3-319-46349-0_28 . hal-01437124

HAL Id: hal-01437124

https://hal.science/hal-01437124v1

Submitted on 10 Apr 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Copyright

Similarity based hierarchical clustering with an application to text collections

Julien Ah-Pine and Xinyu Wang

University of Lyon, Eric Lab
5, avenue Pierre Mendès France
69676 Bron Cedex, France
Julien.Ah-Pine@univ-lyon2.fr
Xinyu.Wang@univ-lyon2.fr

Abstract. Lance-Williams formula is a framework that unifies seven schemes of agglomerative hierarchical clustering. In this paper, we establish a new expression of this formula using cosine similarities instead of distances. We state conditions under which the new formula is equivalent to the original one. The interest of our approach is twofold. Firstly, we can naturally extend agglomerative hierarchical clustering techniques to kernel functions. Secondly, reasoning in terms of similarities allows us to design thresholding strategies on proximity values. Thereby, we propose to sparsify the similarity matrix in the goal of making these clustering techniques more efficient. We apply our approach to text clustering tasks. Our results show that sparsifying the inner product matrix considerably decreases memory usage and shortens running time while assuring the clustering quality.

Key words: Agglomerative hierarchical clustering, Lance-Williams formula, Scalable hierarchical clustering, Kernel machines, Text clustering.

1 Introduction

Hierarchical clustering is an important member in the clustering family. As it is able to reveal internal connections of clusters, it is more informative than its counterpart, flat clustering. Due to this advantage, it is widely applied in different domains like in documents organization where it makes it possible to highlight the relationships between topics.

There are two types of hierarchical clustering: agglomerative and divisive. Given a dataset $\mathcal{D}$ of N instances, agglomerative hierarchical clustering (AHC) recursively merges two clusters at each step, until that all instances are grouped into one cluster. Whereas, divisive hierarchical clustering (DHC) functions in the opposite way. DHC is computationally demanding, as there are $2^{N-1} - 1$ possible subdivisions into two clusters when splitting a dataset of N. Comparatively, AHC is more efficient, and thus more widely studied and applied. The result of AHC is usually represented by a dendrogram, a binary tree composed of $2N - 1$ nodes, to which a real value called height is assigned. The conventional procedure of AHC, also called the stored dissimilarities approach, takes a

pairwise dissimilarity matrix D of size N as input, initializes a binary tree with N leaves (singletons) with null height values, and iteratively adds new nodes (merged clusters) by fusing a pair of clusters (C_i, C_j) determined as follows:

$$(C_i, C_j) = \underset{(C_k, C_l)}{\arg\min}\, D(C_k, C_l) \tag{1}$$

AHC can be computationally costly. For the usual AHC procedure described above, the time complexity is $O(N^3)$. Other approaches, such as NN-chain based methods [1, 7], have time complexity $O(N^2)$. But the drawback is that, NN-chain based methods are constrained by reducibility property, thus they cannot work with median and centroid methods. Another approach, called SparseHC [9] structures clusters with an adjacency hash map. According to its experiment results, SparseHC does decrease the memory growth, but its time complexity is improved for single link only. Besides this approach is not generic, as it is only applicable for single link, complete link and average link. Another method, CURE [4], reduces data by random sampling and partitioning. Though it decreases time complexity to $O(N^2_{sample} log N_{sample})$, its results are indeterministic due to the random procedures. And for BIRCH [11], its time complexity is $O(N)$, but extra structure like clustering features (CF) tree has to be employed in order to store compact summaries of the original data.

In this paper, we propose an AHC approach which is generic for all usual methods. It allows a better scalability compared to the conventional AHC algorithm both in memory and processing time, and its results are deterministic unlike some aforementioned techniques. Our method is based on a new expression of the Lance-Williams (LW) formula, in which we replace dissimilarities with inner product based similarities. This change provides us with two important advantages: (1) it allows us to easily extend AHC to kernel functions; (2) it enables us to design a suitable thresholding strategy so that we can obtain a sparsified similarity matrix, resulting in a more scalable AHC procedure.

In order to illustrate the properties and benefits of our approach, we applied it to text clustering tasks. Our experiments show that results obtained by our framework are identical to the results obtained by the usual AHC methods, demonstrating their equivalence. Moreover, our experimental results show that on a largely sparsified similarity matrix, our approach is still able to cluster correctly, but with higher speed and less memory usage.

The rest of paper is structured as follows. Section 2 introduces fundamental materials on AHC. Section 3 details our approach and the mathematical proof of correctness. Experimental verification on results quality and performance improvement on real-world text clustering tasks can be found in Section 4. Section 5 concludes our paper with a discussion and presents future work.

2 Conventional AHC methods and the LW formula

There are many AHC techniques and reviews of these algorithms can be found in [8] and [10]. Conventional AHC methods can be classified into graph and geometric methods. Single link, complete link, average link and Mcquitty are graph

methods, in which dissimilarity of two clusters is determined by the dissimilarities of instances from these clusters. Due to this property, these techniques can use graph representations that rely on a pairwise dissimilarity matrix. Whereas, for geometric methods, composed of centroid, median and Ward methods, instances are assumed to be represented in an Euclidean space, clusters are represented by prototypes and Euclidean distances between these representative vectors are used as dissimilarities.

Proposed by Lance G.N and Williams W.T in 1967 [5], the LW formula is a convenient formulation, which unifies the graph and the geometric methods mentioned above. It is used as a generic equation for updating the dissimilarity matrix D at each iteration once the newly formed cluster $C_{(ij)} = C_i \cup C_j$ given by (1) has been added to the dendrogram. According to this approach the dissimilarity between $C_{(ij)}$ and another cluster C_k is given by:

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

Depending on the choice of a certain clustering method, values of parameters α_i, α_j, β and γ change accordingly. Table 1 displays parameter values in (2) that correspond to seven particular methods.

Table 1. 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
Mcquitty	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

3 Our approach

In this section, we introduce our method from three aspects: (1) we propose to renew the original LW formula using cosine similarities instead of squared Euclidean distances; (2) we extend this expression to kernel functions; (3) we introduce a simple sparsification strategy which is applied to the similarity matrix in the goal of reducing memory use and running time.

3.1 An equivalent LW formula using cosine similarities

We suppose that the N instances of dataset $\mathcal{D}$ are represented by vectors in an Euclidean space $\mathcal{I}$ of dimension p. Cosine of the angle between two vectors is

considered as their similarity. Input S is a pairwise similarity matrix of size N. For two data points $x, y \in \mathcal{D}$, with $\langle ., . \rangle$ denoting inner product, their similarity is defined as follows:

$$S(x, y) = \langle \frac{x}{\|x\|}, \frac{y}{\|y\|} \rangle \tag{3}$$

Note that this implies that $S(x, x) = 1$ for all data point $x \in \mathcal{D}$. It is an important condition in our context to establish the new expression. We associate S with a dissimilarity matrix D whose general term is the squared Euclidean distance between normalized vectors:

$$\begin{aligned} D(x, y) &= \| \frac{x}{\|x\|} - \frac{y}{\|y\|} \|^2 \\ &= S(x, x) + S(y, y) - 2S(x, y) \quad (4) \\ &= 2(1 - S(x, y)) \quad (5) \end{aligned}$$

With the above assumptions, we provide an expression of the LW formula using S instead of D. In fact, our approach amounts to work with $-\frac{1}{2}D(C_k, C_l)$ instead of $D(C_k, C_l)$. In order to guarantee the correctness of our reasoning, we proceed by induction. In the first iteration of the AHC algorithm, there are N leaves $\{C_k\}$, each is one data point. In this case, the relationship below is straightforward:

$$\begin{aligned} \underset{(C_k, C_l)}{\arg\min}\, D(C_k, C_l) &= \underset{(C_k, C_l)}{\arg\max} -\frac{1}{2} D(C_k, C_l) \\ &= \underset{(C_k, C_l)}{\arg\max}\, S(C_k, C_l) - \frac{1}{2}(S(C_k, C_k) + S(C_l, C_l)) \quad (6) \end{aligned}$$

Next, for the subsequent iterations, we show that the LW formula can be recast as follows:

$$-\frac{1}{2} D(C_{(ij)}, C_k) = S(C_{(ij)}, C_k) - \frac{1}{2}(S(C_{(ij)}, C_{(ij)}) + (\alpha_i + \alpha_j) S(C_k, C_k)) \tag{7}$$

where:

$$\begin{aligned} S(C_{(ij)}, C_k) &= \alpha_i S(C_i, C_k) + \alpha_j S(C_j, C_k) + \beta S(C_i, C_j) \quad (8) \\ &\quad -\gamma |S(C_i, C_i)/2 - S(C_i, C_k) - S(C_j, C_j)/2 + S(C_j, C_k)| \\ S(C_{(ij)}, C_{(ij)}) &= (\alpha_i + \beta) S(C_i, C_i) + (\alpha_j + \beta) S(C_j, C_j) \quad (9) \end{aligned}$$

Equation (7) together with the recurrence formulas (8) and (9) are respectively the counterparts of (1) and (2) that establish our method. With the condition that $S(C_k, C_k) = 1$ for all N singletons $\{C_k\}$, we show below that our formulation is equivalent to the usual LW formula for each clustering scheme listed in Table 1:

1. For single link and complete link, (8) reduces to $S(C_{(ij)}, C_k) = \alpha_i S(C_i, C_k) + \alpha_j S(C_j, C_k) + \beta S(C_i, C_j) - \gamma |S(C_i, C_k) - S(C_j, C_k)|$, and (9) reduces to

$S(C_{(ij)}, C_{(ij)}) = 1$, as $\alpha_j + \alpha_j + 2\beta = 1$. Since in (7) $\alpha_i + \alpha_j = 1$, then (6) with the reduced updating rules of (8) and (9) are globally equivalent to the conventional procedure. Note that for other remaining methods $\gamma = 0$, so that (8) boils down to $S(C_{(ij)}, C_k) = \alpha_i S(C_i, C_k) + \alpha_j S(C_j, C_k) + \beta S(C_i, C_j)$.

2. If we assume again that $S(C_k, C_k) = 1$ for all N singletons $\{C_k\}$ and if we replace $S(C_{(ij)}, C_{(ij)})$ with (9) as well, then the second term in the right-hand side of (7) becomes $-(\alpha_i + \alpha_j + \beta)$. By this, we can divide the remaining clustering methods into two groups: (a) average link, Mcquitty and Ward which have $-(\alpha_i + \alpha_j + \beta) = -1$, and (b) centroid and median which satisfy $-(\alpha_i + \alpha_j + \beta) > -1$:
   (a) Regarding average link, Mcquitty and Ward, it is not difficult to prove by induction that the second term in the right-hand side of (7) always equals to -1. Consequently, for these cases, (6) with the updating rules $S(C_{(ij)}, C_k) = \alpha_i S(C_i, C_k) + \alpha_j S(C_j, C_k) + \beta S(C_i, C_j)$ and $S(C_{(ij)}, C_{(ij)}) = 1$ are globally equivalent to the general procedure.
   (b) Concerning the centroid and median methods, since $\alpha_i + \alpha_j = 1$ in (7), the coefficient assigned to $S(C_k, C_k)$ vanishes. However, $\alpha_i + \alpha_j + 2\beta \neq 1$ in (9) hence $S(C_{(ij)}, C_{(ij)}) \neq 1$. Therefore, it is important to apply the weighting system determined in (9) for the global equivalence of centroid and median to hold.

We can wrap up all particular cases discussed above through the following general procedure which defines our AHC framework. At each iteration, we solve:

$$(C_i, C_j) = \underset{(C_k, C_l)}{\arg\max}\, S(C_k, C_l) - \frac{1}{2}(S(C_k, C_k) + S(C_l, C_l)) \tag{10}$$

After having merged (C_i, C_j) into $C_{(ij)}$, the similarity matrix S is updated by applying the two following equations:

$$\begin{aligned} S(C_{(ij)}, C_k) = {} & \alpha_i S(C_i, C_k) + \alpha_j S(C_j, C_k) + \beta S(C_i, C_j) \\ & -\gamma |S(C_i, C_k) - S(C_j, C_k)| \end{aligned} \tag{11}$$

$$S(C_{(ij)}, C_{(ij)}) = \delta_i S(C_i, C_i) + \delta_j S(C_j, C_j) \tag{12}$$

Table 2 lists parameter values of each method in our framework. Note that in this table, the newly introduced parameters δ_i and δ_j sum to one except for centroid and median methods. In fact, for the other methods, we could have taken any values providing that $\delta_i + \delta_j = 1$.

3.2 Extending to kernel functions

Our approach allows us to naturally extend AHC methods to kernel functions (see for example [2]) since most of the latter mappings are defined with respect to inner products. Consequently, in our method, broader similarity measures can be easily employed and non linearly separable cases can be addressed effectively.

Thereby, let K denote a pairwise inner product matrix (or Gram matrix) of size N whose general term for two data points $x, y \in \mathcal{D}$ is $K(x, y) = \langle \phi(x), \phi(y) \rangle$

Table 2. The cosine similarity based formula: methods and parameter values.

Methods	α_i	α_j	β	γ	δ_i	δ_j
single	1/2	1/2	0	-1/2	1/2	1/2
complete	1/2	1/2	0	1/2	1/2	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	1/2	1/2
Mcquitty	1/2	1/2	0	0	1/2	1/2
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	$\frac{\lvert C_i\rvert^2}{(\lvert C_i\rvert+\lvert C_j\rvert)^2}$	$\frac{\lvert C_j\rvert^2}{(\lvert C_i\rvert+\lvert C_j\rvert)^2}$
median	1/2	1/2	-1/4	0	1/4	1/4
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	1/2	1/2

where $\phi : \mathcal{I} \to \mathcal{F}$ is a mapping from $\mathcal{I}$ to $\mathcal{F}$ and the latter notation designates a feature space of dimension $q > p$ (q is possibly infinite).

The S matrix in our approach should contain cosine measures, and more importantly, its diagonal entries should be constant. Gaussian and Laplacian kernels satisfy this condition naturally, but for other kernels, they have to be normalized. To generalize all the cases, we obtain a cosine similarity matrix by applying for all $x, y \in \mathcal{D}$: $S(x, y) = K(x, y)/\sqrt{K(x, x)K(y, y)}$.

3.3 Sparsification of the cosine similarity matrix

In general terms, S could contain negative values. In that case, let $m < 0$ be the minimal value in S and $|m|$ its absolute value. It is always possible to transform S in order to have non negative values using the following rescaling operator, $\forall x, y \in \mathcal{D}$:

$$S(x, y) \leftarrow \frac{S(x, y) + |m|}{1 + |m|} \tag{13}$$

Since this mapping is monotonically increasing, the resulting S remains an inner product matrix and, in addition, it has ones on its diagonal.

Assuming that S is non negative, we propose to apply a simple thresholding operator which depends on a parameter $\tau \in [0, 1]$: any similarity value below τ[1] is considered irrelevant and it is replaced with 0[2], $\forall x, y \in \mathcal{D}$:

$$S(x, y) \leftarrow S(x, y) I_{(S(x,y)\geq\tau)} \tag{14}$$

where $I_{(S(x,y)\geq\tau)} = 1$ if $S(x, y) \geq \tau$ and $I_{(S(x,y)\geq\tau)} = 0$ otherwise.

The resulting S matrix is sparser than the original one and thus requires less memory.

Next, we propose to restrict the search for pairs of clusters to merge in (10) to the following subset: $\mathbb{S} = \{(C_k, C_l) : S(C_k, C_l) > 0\}$. This allows the running

[1] Note that if $\tau = 0$ then S is not sparsified.

[2] It is interesting to mention that such a thresholding operator cannot be applied to a dissimilarity matrix D, because the larger values are the less relevant ones in that case and replacing them with 0 is not sound.

time to be diminished as well, since the bottleneck procedure in the general AHC algorithm is precisely the search for the optimal proximity value, which has $O(N^2)$ time complexity. Accordingly, we propose to replace (10) with:

$$(C_i, C_j) = \underset{(C_k, C_l) \in \mathbb{S}}{\arg\max}\, S(C_k, C_l) - \frac{1}{2}(S(C_k, C_k) + S(C_l, C_l)) \tag{15}$$

As we shall see in the next section, not only this approach dramatically reduces the processing time but it also allows obtaining better clustering results.

4 Experiments

The goals of our experiments are to demonstrate that: (1) our framework based on equations (10), (11) and (12) is equivalent to the usual AHC procedure (1) based on the LW formula (2) under the assumptions exposed previously; (2) sparsifying the cosine similarity matrix with (14) and applying our AHC given by (15), (11) and (12) considerably decreases memory use and running time while having the capacity to provide better clustering results.

To this end, we experimented on text clustering tasks. Indeed, hierarchical clustering is particularly interesting in this case, since it allows expressing the relationships between different topics in a collection and at different granularity levels. Moreover, cosine similarities are classic proximity functions used for documents. In addition, hierarchical document organization based on the conventional AHC procedure faces the problem of scalability since text collections are usually very large. Our experiments seek to demonstrate new perspectives to overcome these limits.

It is important to note that our purpose is not to compare the different AHC methods between each other, but rather to exemplify the properties of our framework compared to the usual AHC procedure using the LW formula. As a consequence, the results obtained by the latter conventional approach are our baselines.

4.1 Datasets, preprocessing and evaluation measures

We used three well-known corpora employed in text clustering benchmarks: Reuters-21578[3] (Reuters), Smart [3] and 20Newsgroups[4] (20ng) [6]. Their descriptive statistics are given in Table 3.

We used the bag-of-words approach where each document is represented by a vector in the space spanned by a set of terms. As for preprocessing we applied a rough feature selection by removing terms that appear in less than 0.2% and more than 95% documents of the collection. No stemming, lemmatization nor stop word removal were applied. Then, the tfidf weighting strategy was performed.

[3] Distribution 1.0, the ApteMod version.

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

The adjusted Rand index (ARI) and (the absolute value of) the cophenetic correlation (CC) between dendrograms are used to compare the clustering outputs. CC is employed to evaluate how far our dendrogram is from the one produced by the conventional AHC procedure. In this case, higher is better and a maximum value one means that the dendrograms are equivalent and thus represent the same hierarchy. ARI is an external assessment criterion that evaluates the quality of the clustering output in regard to a given ground-truth. It requires to flatten the dendrogram with the correct number of clusters, then the obtained partition and the ground-truth are compared to each other. Greater ARI values imply better clustering outputs. The maximum value one is observed when the ground-truth is perfectly recovered.

Table 3. Descriptions of datasets.

Dataset	Nb of classes	Nb of documents	Nb of features
Reuters	10	2446	2547
Smart	3	3893	3025
20ng	15	4483	4455

4.2 Experiments settings and results

Given a term-document matrix, two types of matrices are generated: the cosine similarity matrix S and the corresponding distance matrix D as defined by (5). Note that since the term-document matrix consists of non negative values then S takes values in $[0, 1]$ therefore no rescaling operator is needed.

Given a clustering method, the S matrix is taken as the input to our framework, while the related dense D matrix is input to the conventional AHC algorithm. Consequently, two dendrograms are returned and we compute the CC in order to assess the similarity between the two outputs. Two cases are of interest: (1) when $\tau = 0$ which means no sparsification and the dense S is used; and (2) when $\tau > 0$ and being increased which leads to sparser and sparser S matrices.

In addition to 0, we chose other threshold values τ as the 10th, 25th, 50th, 75th and 90th percentiles of distribution of values in S. Let k denote the rank of a percentile so that $k \in \{0, 10, 25, 50, 75, 90\}$ with the convention that the 0th percentile is 0. Accordingly, when k grows the kth percentile τ is greater and greater and the S matrix becomes sparser and sparser.

We experimented with two types of kernel: linear and Gaussian. The linear kernel is simply the inner dot product in $\mathcal{I}$ between normalized vectors as defined in (3). The Gaussian kernel between two points $x, y \in \mathcal{D}$ is given by $K(x, y) = \exp(-\gamma\|x - y\|^2)$. It corresponds to a cosine measure in $\mathcal{F}$. In our experiment we set γ to $1/p$ by default[5].

[5] Note that this default setting is used in popular SVM packages. Furthermore, in this case γ is very low and the Gaussian kernel provides values close to one and close to each other between pairs of points.

In Figure 1, we show the results obtained for all seven methods on Reuters, Smart and 20ng datasets respectively. We report the curves of several measurements (y-axis) when S is progressively sparsified as the percentile rank (x-axis) increases. In addition to CC and ARI graphs (dotted lines with circle and triangle symbols respectively), the percentage of the memory cost of a sparse S with respect to the dense S, and the proportion of the running time when using a sparse S as compared to the dense S, are plotted as well (solid lines with plus symbols and dashed lines with cross symbols respectively). Therefore, the memory and processing time costs related to the full S (corresponding to the 0th percentile where $\tau = 0$) serve as baselines (with y-axis value of 100%). In these cases, the lower the percentages the bigger the gains.

Equivalence between our method and the LW formula. In Figure 1, for all datasets and both kernels, the CC values are all equal to one when $\tau = 0$ (0th percentile shown at the origin). This empirically demonstrates that our approach is equivalent to the AHC algorithm using the LW formula as claimed previously.

Next, as the percentile rank increases, the CC values generally decrease illustrating the fact that the dendrograms move away from the LW formula based results. However, when using the linear kernel, the CC values generally remain high even when the majority of the similarity values are removed. Concerning the Gaussian kernel, the CC values drop rapidly after having thresholded 10% of the lowest similarities but they start increasing again after this fall.

The single link method however, presents a peculiar behavior: for all collections and both kernels, it always recover the result given by the usual AHC procedure despite the fact that 90% of the S matrix is sparsified. In other words, our framework is able to obtain the same dendrogram provided by the original LW formula but with 90% of memory usage and running time saved.

Impact of the sparsification of S on scalability. Let $M \leq N^2$ be the number of non zero cells in S. The storage cost of our approach is $O(M)$. The time complexity[6] is $O(NM)$ which indicates a linearly relationship with respect to the storage complexity.

In Figure 1, the solid lines with plus symbols give the percentage of size of the sparse S with respect to the dense S. As expected, this quantity linearly decreases as the percentile rank k grows.

Next, the dashed lines with cross signs show the proportion of the processing time observed with a sparse S with respect to the running time noted with the dense S. We observe linear curves as well which depicts the linear relationship between the memory and time complexities as mentioned above.

The sparsification of the S matrix enables decreasing the storage complexity and the running time. Besides, it also has an impact on the clustering quality. Previously, we have noticed that CC values were decreasing as S were sparser

[6] Similarly to the general AHC algorithm based on a dissimilarity matrix for which $M = N(N-1)$.

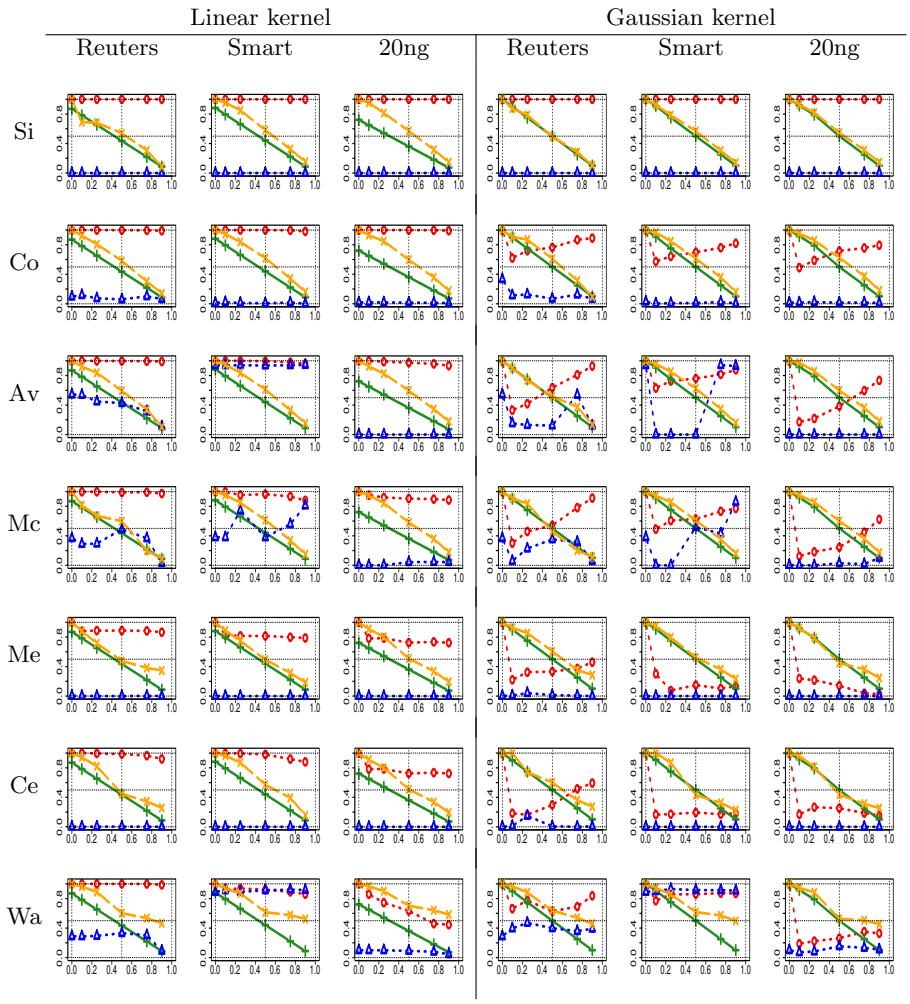

Fig. 1. Experiments results for linear and Gaussian kernels in left and right blocks. Rows correspond to AHC methods (using their abbreviations) and columns to collections. In each graph: each point corresponds to one of the measurements listed afterwards with respect to an S matrix; the x-axis correspond to percentile ranks (divided by 100) which define the threshold values τ (not shown); solid lines with plus signs represent the relative memory use, dashed lines with cross signs show the relative running time, dotted lines with circle symbols indicates the absolute value of cophenetic coefficient (CC), dotted lines with triangle symbols give the ARI values.

and sparser. In the sequel, we examine some cases in which our framework wins on both sides: scalability and quality.

Impact of sparsification of S on clustering quality. We focus on the quality of clustering outputs by analyzing the ARI values. We observe that average link, Mcquitty and the Ward techniques worked out better in general. Surprisingly, many of the best results are obtained with a very sparse S matrix and not with the full one. In Table 4 we report the best outcomes where such a phenomenon is illustrated. Mem% and Time% indicate the percentage of saved memory and processing time respectively, when using the corresponding sparse S as compared to the dense S.

Table 4. Best ARI results for each collection when $\tau = 0$ (baseline) and when $\tau > 0$ (sparsified S) and relative gains in memory and time.

	Method	kernel	τ	Mem%	Time%	CC	ARI
Reuters	Average	Gaussian	0	0	0	1	**0.543**
	Average	Gaussian	0.99	-75	-62	0.81	0.539
Smart	Average	Linear	0	0	0	1	0.939
	Average	Linear	0.078	-90	-85	0.96	**0.944**
20ng	Ward	Gaussian	0	0	0	1	0.100
	Ward	Gaussian	0.99	-50	-47	0.26	**0.154**

For Reuters, the best ARI value is provided by average link with a full S given by the Gaussian kernel. However, a comparable performance is obtained with the same method and kernel but with a sparse S that saves 75% of memory and 62% of processing time.

Concerning Smart, average link gave the best ARI value as well, but with a linear kernel. Compared to the LW based AHC algorithm, our framework obtained higher ARI and with 90% of memory and 85% of running time less.

Regarding 20ng, it is the Ward technique with Gaussian kernel that worked out the best. Our method allows increasing the baseline ARI value up to 54% and meanwhile consuming around half of memory and running time.

5 Discussion and future work

We have introduced an equivalent formulation of the LW formula based on cosine similarities instead of squared Euclidean distances. Our AHC procedure that relies on this formulation and a sparsified cosine similarity matrix, not only has better scalability properties but is also able to give better clustering results.

We believe that two reasons account for this phenomenon. Firstly, sparsifying the S matrix reduces the noise by removing the lowest similarity values, therefore leading to better clustering performances. Secondly, when two clusters (C_i, C_j) are merged together, their respective neighborhoods (clusters having a non null

similarity value with C_i and C_j respectively) are fused as well, so that $C_{(ij)}$ has a larger neighborhood than both C_i and C_j. Furthermore, the updating rule (11) allows reinforcing the similarity value of $C_{(ij)}$ with C_k if the latter cluster belongs to both initial neighborhoods. In fact, our approach can be viewed as a sort of "transitive closure" starting with reliable seeds (the pairs with highest similarity values) and propagating similarities through "trusted" neighborhoods.

However, the main drawback of our method is that, either sparsifying S does not improve the ARI value at all (see complete link applied to Reuters with Gaussian kernel in Figure 1 for instance), or the improvements are not regular and setting the threshold value τ becomes difficult. More theoretical investigations should be undertaken in these respects to have a better understanding of the properties of our framework.

Another line of research that we intend to pursue is to implement our approach in the manner of distributed computing to take better advantage of its scalability.

Acknowledgment This work was supported by the french national project Request PIA/FSN.

References

1. Bruynooghe, M.: Classification ascendante hiérarchique des grands ensembles de données : un algorithme rapide fondé sur la construction des voisinages réductibles. Cahiers de l'analyse des données 3(1), 7–33 (1978)
2. Cristianini, N., Shawe-Taylor, J.: An introduction to support vector machines and other kernel-based learning methods. Cambridge university press (2000)
3. Dhillon, I.S.: Co-clustering documents and words using Bipartite Co-clustering documents and words using Bipartite Spectral Graph Partitioning. Proc of 7th ACM SIGKDD Conf pp. 269–274 (2001)
4. Guha, S., Rastogi, R., Shim, K.: Cure: an efficient clustering algorithm for large databases. In: ACM SIGMOD Record. vol. 27, pp. 73–84. ACM (1998)
5. Lance, G.N., Williams, W.T.: A general theory of classificatory sorting strategies ii. clustering systems. The computer journal 10(3), 271–277 (1967)
6. Lang, K.: NewsWeeder : learning to filter netnews. In: Proceedings of the Twelfth International Conference on Machine Learning. pp. 331–339 (1995)
7. Murtagh, F.: A survey of recent advances in hierarchical clustering algorithms. The Computer Journal 26(4), 354–359 (1983)
8. Murtagh, F., Contreras, P.: Algorithms for hierarchical clustering: an overview. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 2(1), 86–97 (2012)
9. Nguyen, T.D., Schmidt, B., Kwoh, C.K.: Sparsehc: a memory-efficient online hierarchical clustering algorithm. Procedia Computer Science 29, 8–19 (2014)
10. Xu, R., Wunsch, D., et al.: Survey of clustering algorithms. Neural Networks, IEEE Transactions on 16(3), 645–678 (2005)
11. Zhang, T., Ramakrishnan, R., Livny, M.: Birch: an efficient data clustering method for very large databases. In: ACM Sigmod Record. vol. 25, pp. 103–114. ACM (1996)