A new Information theory based clustering fusion method for multi-view representations of text documents

Juan Zamora, Jérémie Sublime

A new Information theory based clustering fusion method for multi-view representations of text documents

Juan Zamora[1 [0000−0003−0003−182X]] and Jérémie Sublime[2,3 [0000−0003−0508−8550]]

[1] Instituto de Estadística, Pontificia Universidad Católica de Valparaíso, Valparaíso 2340025, Chile, juan.zamora@pucv.cl
[2] ISEP, Lisite Laboratory – DaSSIP Team, 10 rue de Vanves, 92130 Issy-Les-Moulineaux, France, jeremie.sublime@isep.fr
[3] LIPN - CNRS UMR 7030, University Paris 13, 99 Avenue J.-B. Clément, 93430 Villetaneuse, France, sublime@lipn.univ-paris13.fr

Abstract. Multi-view clustering is a complex problem that consists in extracting partitions from multiple representations of the same objects. In text mining and natural language processing, such views may come in the form of word frequencies, topic based representations and many other possible encoding forms coming from various vector space model algorithms. From there, in this paper we propose a clustering fusion algorithm that takes clustering results acquired from multiple vector space models of given documents, and merges them into a single partition. Our fusion method relies on an information theory model based on Kolmogorov complexity that was previously used for collaborative clustering applications. We apply our algorithm to different text corpuses frequently used in the literature with results that we find to be very satisfying.

Keywords: Multi-view Clustering · Information theory · Corpus Analysis.

1 Introduction

The goal of text corpus clustering is to partition a collection of text documents into several groups, such that texts inside the same groups (or clusters) are similar and share common themes or have a common style, while documents in different clusters are very distinct in nature. To achieve this goal, text documents must first be transformed using models such as the Vector Space Model (VSM) [20] in order to transform the original documents into numerical vectors that can be used by clustering algorithms such as K-Means or hierarchical clustering. One difficulty with the VSM model is the large number of existing methods to transform text documents into vector representations. Many representation models exist, some are topic oriented, others focus on word embedding, while some methods are purely statistical representations. This abundance of methods in the literature allows for multiple vector representations of the same texts, all with different strengths and weaknesses. Applying clustering algorithms to these

multiple representations can be seen as a multi-view clustering problem where the goal could be to find a consensus between the clustering partitions proposed under the various Vector Space Models [3]. Within this context, in this paper we propose a new method inspired from collaborative clustering, and which relies on the notion of Kolmogorov complexity to merge clustering partitions acquired from the clustering algorithms applied to different vector representations of text documents. Our proposed method is compared with state of the art methods applied to common text corpuses that can be found in the literature.

The remainder of this paper is organized as follows: Section 2 focuses on various related works about both text mining and the information theory model used in this paper. Section 3 presents our algorithm. Section 4 features our experimental results and some comparisons with other methods. Finally, in section 5 we draw some conclusion and give some ideas on possible extensions of this work.

2 State of the art

Cluster ensembles is an overall framework in which multiple partitions are combined in order to obtain a consensus clustering. Multi-view clustering is one of the specific problems covered in this area [5].

The problem of combining multiple data partitions into a single one has been tackled at least by two approaches, namely Clustering Ensembles [21, 9, 4, 22, 26, 24, 10, 16] and Multi-View Clustering[8, 13, 6, 27, 11, 12, 25, 18, 7, 3], also known as data fusion.

In ensemble learning and ensemble clustering, several algorithms will work on the same data set with the goal of achieving a single result that should be better that the partitions learned from the different algorithms. As one can see, in ensemble clustering, several algorithms work on the same data and therefore the same view. However, in the case of multi-view clustering like in the present work, we have several algorithms and each of them works in a different view of the same data. And since we are dealing with several views, the goal with multi-view clustering is to merge them while taking into considerations that there might be multiple truths [30].

It is worth mentioning, that the distinction between ensemble clustering and multi-view clustering is not always obvious in the literature and some confusion may exist with different naming conventions depending on the field of application. In the following subsection, we make a quick review of the literature for both multi-view and ensemble clustering with a particular focus on text mining applications and methods that are close to the one presented in this work.

2.1 State of the art on combining multiple clustering partitions

There are many different applications that require to combine multiple clustering partitions: In [9], the authors make a proposal for music clustering using partitions obtained from different music feature sets. Among these sets, they

employ several word-level features. They pose the ensemble clustering problem as a binary clustering in a space induced by the multiple partitions. Additionally, they explore various optimization criteria for finding consensus partitions and propose a strategy for determining the final number of clusters. It is interesting to note that they apply this proposal for

In [7], the authors work specifically on text clustering. They propose to generate several partitions from each view by using different feature representations and then applying a clustering algorithm over each one. Then, similarity matrices are computed in three different ways, namely two based on partition memberships and another one based on feature similarity. Finally, a combined similarity matrix is obtained from those three previous ones and a standard clustering technique is applied to produce the consensus partition. In the same direction, [3] use more diverse text representations as views, more specifically LDA[1], Word2Vec[14] and TF-IDF[19] and then apply the same idea as the former work.

It is woth noting that multi-view text clustering shouldn't be confused with distributed clustering of texts [28], which mainly consists in distributing the clustering task without consideration for whether or not this is a multi-view task.

Another common application of multi-view clustering is multilingual clustering. In [18], for this specific application, the authors pose the multi-view clustering problem as a tensor decomposition as this approach was proven earlier to be theoretically efficient [11, 12].

2.2 Methods to combine multiple partitions

In [21], the authors pose that an application of Cluster Ensembles is to combine partitions obtained from partial sets of features. As we have seen earlier, this is a case of multi-view clustering. Additionally, they pose that a motivation for using a cluster ensemble is to build a more robust solution that performs well over a wide range of data sets. Since the diversity of base partitions has a positive impact on the final consensus solution, it can be introduced mainly by using different sets of features in each partition, different parameter configurations of the same algorithm (values of k for k-means) and also using different and complementary base techniques. The authors also formulate consensus clustering as a hyper-graph cutting problem and solve in three different ways.

Co-association matrices are based on relative co-occurrence of two data points in the same cluster. They are another very common tool to tackle multi-view clustering. Several works exploit them in order to produce final partitions from several combinations of different data representations. [4] explore two strategies for producing cluster ensembles: Using different views and using different clustering algorithms or parameter configurations. [26] address the problem from a similarity matrix completion problem in which missing values are associated to uncertain data pairs, this is pair of data points whose common membership in every partition is not consistent. In the same path, [16] propose to weight the

contribution of each co-association matrix based on a novel reliability measure of each partition within the ensemble.

Some other contributions employ an utility function to measure similarity between partitions and then directly maximize an objective function to obtain the consensus [22, 24, 10].

In [8], a hybrid clustering method based on weighted linear combination of distance matrices for textual and bibliometric information is proposed.

2.3 Multi-view clustering applications and Kolmogorov complexity

In the work of [17] [23], the notion of minimum description length (MDL) is introduced, with the *description length* being the minimal number of bits needed by a Turing Machine to describe an object. This measure of the minimal number of bits is also known under the name Kolmogorov complexity.

If $\mathcal{M}$ is a fixed Turing machine, the complexity of an object $\mathbf{x}$ given another object $\mathbf{y}$ using the machine $\mathcal{M}$ is defined as $K_{\mathcal{M}}(\mathbf{x}|\mathbf{y}) = \min_{p \in \mathcal{P}_{\mathcal{M}}} \{l(p) : p(\mathbf{y}) = \mathbf{x}\}$ where $\mathcal{P}_{\mathcal{M}}$ is the set of programs on $\mathcal{M}$, $p(\mathbf{y})$ designates the output of program p with argument y and l measures the length (in bits) of a program. When the argument $\mathbf{y}$ is empty, we use the notation $K_{\mathcal{M}}(\mathbf{x})$ and call this quantity the complexity of $\mathbf{x}$. The main problem with this definition is that the complexity depends on a fixed Turing machine $\mathcal{M}$. Furthermore, the universal complexity is not computable, since it is defined as a minimum over all programs of all machines.

In relation with this work, in [15], the authors solved the aforementioned problem by using a fixed Turing Machine before applying this notion of Kolmogorov complexity to collaborative clustering, which is a specific case of multi-view clustering where several clustering algorithms work together in a multi-view context but aim at improving each other partitions rather than merging them [2]. While collaborative clustering does not aim at a consensus, this application is still very close to what we try to achieve in this paper where we try to merge partitions of the same objects under multiple representations. For these reasons, we decided to use the same tool.

In the rest of this paper, just as the authors did in [15], we will consider that the Turing Machine $\mathcal{M}$ is fixed, and to make the equations easier we will denote by $K(\mathbf{x})$ the complexity of $\mathbf{x}$ on the chosen machine. Then, we adapt the equations used in their original paper to our multi-view context for text mining and we use Kolmogorov complexity as a tool to compute the complexity of one partition given another partition. The algorithm to do so and how we use it is described in the next section.

3 Proposed merging method

3.1 Problem definition

Let us consider a data set $\mathcal{X}$ of n data points and a measure of similarity S that allows to quantify the strength of the connection or closeness between any pair

of data points in $\mathcal{X}$. The problem of data clustering can be stated as inducing an equivalence relation[4] on $\mathcal{X}$ such that points a, b in the same equivalence class (that is a cluster) have a larger similarity value $\mathsf{S}(\mathsf{a}, \mathsf{b})$ in comparison with $\mathsf{S}(\mathsf{a}, \mathsf{c})$ or $\mathsf{S}(\mathsf{b}, \mathsf{c})$ for any other point c in a different equivalence class.

The Multi-view clustering task considers that the information regarding to each data point in $\mathcal{X}$ comes from multiple sources called views. After performing a clustering algorithm over each view several partitions are generated. Let us define this set of partitions as $\mathcal{P}$, and denote each of them with a capital letter (e.g.: A).

A partition A is a set of $|A|$ disjoint sets $\mathbf{c} \in \wp(\mathcal{X})$ (the Power set of $\mathcal{X}$) called clusters of $\mathcal{X}$. Let us define an agreement function Ω between two clusters as a mapping $\Omega : \wp(\mathcal{X}) \times \wp(\mathcal{X}) \to [0, 1]$ which attains lower values for clusters having a smaller overlap and higher values for clusters sharing more elements of $\mathcal{X}$. In this work we employ the Jaccard similarity function to measure agreement between two clusters.

For a point $\mathsf{p} \in \mathcal{X}$, its cluster in any partition $A \in \mathcal{P}$ is denoted by $\mathcal{N}_{\mathsf{p}}^{A}$ and it is defined as:

$$\mathcal{N}_{\mathsf{p}}^{A} = \{\mathsf{x} \in \mathcal{X} | \exists \mathbf{c} \in A \wedge \mathsf{p} \in \mathbf{c} \wedge \mathsf{x} \in \mathbf{c}\}$$

Given a cluster $\mathbf{c}$ and a partition B the function that maps $\mathbf{c}$ to the cluster in B with the largest overlap is called maximum agreement function and it is defined as follows:

$$\Phi_B(\mathbf{c}) = \underset{\mathbf{e} \in B}{\mathbf{argmax}}\, \Omega(\mathbf{c}, \mathbf{e}) \tag{1}$$

3.2 The Algorithm

Our goal in this paper is to combine several partitions in order to build a final consensus. To this end, in our method we perform successive pairwise fusion procedures between partitions following a bottom-up strategy until we reach a single partition. This procedure is depicted in Algorithm 1.

Without loss of generality, when a fusion step is performed between two partitions A and B, a new partition C is created. Since the successive partition fusions are performed by following the maximum agreement criteria between clusters as stated in Eq. (1), it is possible that some data points do not fit to this rule and hence be marked as exceptions during the execution of the merge operation. The set of data points marked as exceptions before the creation of partition C is denoted by ξ_C, formally,

$$\xi_C = \{p \in \mathcal{X} | \mathcal{N}_{\mathsf{p}}^{A} \cap \Phi_B(\mathcal{N}_{\mathsf{p}}^{A}) = \emptyset \cup \mathcal{N}_{\mathsf{p}}^{B} \cap \Phi_A(\mathcal{N}_{\mathsf{p}}^{B}) = \emptyset\} \tag{2}$$

Thus, when partition C is created, each point $\mathsf{p} \in \xi_C$ receives a weight $W_C(\mathsf{p}, \mathbf{c})$ for every cluster $\mathbf{c} \in C$. This weight is made up by the relative weights

[4] For the clustering task, the relation could be stated as "has the same label as".

that both source partitions A and B contribute, namely $\omega_A(\mathsf{p}, \mathbf{c})$ and $\omega_B(\mathsf{p}, \mathbf{c})$. Without loss of generality, the contribution of each source partition is given by:

$$\omega_A(\mathsf{p}, \mathbf{c}) = \begin{cases} \Omega(\mathbf{c}, \mathcal{N}_{\mathsf{p}}^A) & \text{if } \mathsf{p} \notin \xi_A \\ \Omega(\mathbf{c}, \Phi_A(\mathbf{c})) \cdot W_A(\mathsf{p}, \Phi_A(\mathbf{c})) & \text{if } \mathsf{p} \in \xi_A \end{cases} \tag{3}$$

Thus, the final weight $W_C(\mathsf{p}, \mathbf{c})$ for each point $\mathsf{p} \in \xi_C$ in each cluster $\mathbf{c} \in C$ is given by:

$$W_C(\mathsf{p}, \mathbf{c}) = \frac{\omega_A(\mathsf{p}, \mathbf{c})}{2} + \frac{\omega_B(\mathsf{p}, \mathbf{c})}{2}$$

A more detailed description of this merging process is depicted in Algorithm 2. It is important to indicate that once a point is marked as an exception, it remains so through all the subsequent fusions. After the last fusion, each of these exception data points are assigned to one of the final clusters by picking the one whose membership weight is the highest. This exception resolution is described between lines $7-9$ in Algorithm 1 where $K(A|B)$ is the kolmogorov complexity of partition A knowing partition B [15]:

$$K(A|B) = K_B \times (\log K_A + \log K_B) + |\xi_C| \times (\log n + \log K_A) \tag{4}$$

with n the total number of points, K_A the number of clusters in partition A, K_B the number of clusters in partition B and ξ_C the set of exceptions between partitions A and B as defined in Equation (2).

Algorithm 1: Main procedure for building the consensus partition.

Input: A set $\mathcal{P}$ of m partitions over the data $\mathcal{X}$.
Output: A consensus partition.

1 $\mathcal{Q} \leftarrow []$ `/* exceptions after each merge operation */`
2 **while** $|\mathcal{P}| > 1$ **do**
3 $\quad A, B \leftarrow \underset{A^*, B^* \in \mathcal{P}}{\mathbf{argmin}}\, \mathsf{K}(A^*|B^*) + \mathsf{K}(B^*|A^*)$
4 $\quad C \leftarrow \mathbf{merge}(A, B, \mathcal{Q}, W)$
5 $\quad$ add C into $\mathcal{P}$
6 $\quad$ remove A, B from $\mathcal{P}$
`/* Solving points marked in last item from Q */`
7 $\xi_D \leftarrow$last partition's exceptions added to $\mathcal{Q}$
8 **foreach** $\mathsf{p} \in \xi_D$ **do**
9 $\quad \mathcal{N}_{\mathsf{p}}^D \leftarrow \underset{\mathbf{c} \in D}{\mathbf{argmax}}\, W_D(\mathsf{p}, \mathbf{c})$
10 **return** D

Algorithm 2: Merge procedure that fuses two partitions into a new one identifying also troublesome points as exceptions.

Input: Partitions $A, B \in \mathcal{P}$ s.t. $|A| > |B|$
, list with previous merge exceptions $\mathcal{Q}$ and weight function for previously created partitions W
Output: New partition C and a set of marked points along with their scores $\forall \mathbf{c} \in C$.

$\mathcal{M} \leftarrow []$
foreach $\mathbf{a} \in A$ **do**
  add $\Phi_B(\mathbf{a})$ into $\mathcal{M}[\mathbf{a}]$
foreach $\mathbf{b} \in B$ **do**
  `/* b can be associated to more than one cluster in A */`
  add $\mathbf{b}$ into $\mathcal{M}[\Phi_A(\mathbf{b})]$
$C \leftarrow \emptyset$ `/* The new partition to be returned */`
foreach $\mathbf{a} \in A$ **do**
  $\mathbf{c} \leftarrow \emptyset$
  foreach $\mathbf{b} \in \mathcal{M}[\mathbf{a}]$ **do**
    $\mathbf{c} \leftarrow \mathbf{c} \cup (\mathbf{a} \cap \mathbf{b})$
    $\mathbf{a}' \leftarrow \mathbf{a}$
    $\mathbf{a} \leftarrow \mathbf{a} - \mathbf{b}$ `/* updates cluster a */`
    $\mathbf{b} \leftarrow \mathbf{b} - \mathbf{a}'$ `/* updates cluster b */`
  add $\mathbf{c}$ into C
`/* generating the list of marked points by the current fusion */`
$\xi_C \leftarrow \emptyset$ **foreach** $\mathbf{a} \in A$ **do**
  if $|\mathbf{a}| > 0$ **then**
    add each $\mathtt{p} \in \mathbf{a}$ into ξ_C
  foreach $\mathbf{b} \in \mathcal{M}[\mathbf{a}]$ *s.t.* $|\mathbf{b}| > 0$ **do**
    add each $\mathtt{p} \in \mathbf{b}$ into ξ_C
add ξ_C into $\mathcal{Q}$
foreach $\mathtt{p} \in \xi_C$ *and* $\mathbf{c} \in C$ **do**
  $W_C(\mathtt{p}, \mathbf{c}) = \frac{\omega_A(\mathtt{p}, \mathbf{c})}{2} + \frac{\omega_B(\mathtt{p}, \mathbf{c})}{2}$
return C

4 Experimental results

4.1 Experimental settings

Since external class labels are available for each data set, let us consider the true clustering T and the final partition obtained by the clustering algorithm as C. Then, two measures are employed to assess the quality of a clustering solution, namely Entropy and Purity. Entropy is defined in two parts: the former allows to measure the Entropy for a single partition and it is characterized for any partition $\mathbf{c} \in C$ in equation (5). The latter is just a weighted average of the entropy computed for all the partitions in the final solution and it is defined in

equation (6). Purity is defined in a similar way, that is first the Purity of a single partition is defined in equation (7) and then, the overall Purity of the partition is denoted as equation (8).

$$\mathsf{E}(\mathbf{c}) = -\frac{1}{\log|T|}\sum_{\mathbf{t}\in T}\frac{|\mathbf{c}\cap\mathbf{t}|}{|\mathbf{c}|}\log\frac{|\mathbf{c}\cap\mathbf{t}|}{|\mathbf{c}|} \tag{5}$$

$$\mathsf{Entropy}(C) = \sum_{\mathbf{c}\in C}\frac{|\mathbf{c}|}{n}\mathsf{E}(\mathbf{c}) \tag{6}$$

$$\mathsf{P}(\mathbf{c}) = \frac{1}{|c|}\max_{\mathbf{t}\in T}|\mathbf{c}\cap\mathbf{t}| \tag{7}$$

$$\mathsf{Purity}(C) = \sum_{\mathbf{c}\in C}\frac{|\mathbf{c}|}{n}\mathsf{P}(\mathbf{c}) \tag{8}$$

Entropy measures the degree in which the true classes are dispersed within each cluster. A good solution is the one that does not break the true clusters into too many parts. Purity is targeted to measure the extent to which each cluster contains documents from mostly a single true class. Thus, a good solution should present homogeneous clusters in terms of the true classes of the contained documents.

Since the quality of the overall solution depends on the initial source k-Means clusterings, which in turn have a random nature, we follow the scheme presented in [29] to eliminate some of this sensitivity in the performance assessment. This is, we use several values for k and for each specific value, the overall clustering procedure is repeated 10 times and the best performance solution is kept. Additionally, since partition quality improves as the number of clusters increases, relative performances are reported for each clustering solution. To compute the relative entropy, we divide the entropy attained by a particular solution by the smallest entropy for that particular data set and value of k. In case of relative purity and in order to allow the same interpretation of the relative entropy, we divide the best Purity attained for that particular data set and value of k by the entropy value obtained by the clustering solution under evaluation. Since these two ratios represent the extent to which a specific algorithm performed worse than the best algorithm, for each dataset better solutions are closer to 1.0 and they are worse as the become greater then 1.0. Finally, as a performance summary for each solution the average relative performance across all data sets are reported for each clustering solution.

4.2 Results and interpretations

The result tables 2,1, 4 and 3 show the relative performances attained by the proposal, each source clustering and another ensemble method recently proposed in [3].

k	lda	skipgram	tfidf	fraj	proposal
5	1.737	1.509	**1.031**	2.271	**1.075**
10	1.565	1.447	**1.019**	2.045	**1.074**
15	1.727	1.453	**1.021**	2.239	**1.057**

Table 1. Average relative entropy

k	dataset	lda	skipgram	tfidf	fraj	proposal
5	WebKB	**1.000**	1.111	1.008	1.289	**1.012**
5	20Newsgroup	1.306	1.256	**1.000**	1.813	**1.116**
5	BBCSport	3.641	1.948	**1.000**	4.397	**1.144**
5	Reuters-R8	**1.000**	1.722	1.116	1.586	**1.028**
10	WebKB	**1.000**	1.216	1.074	1.256	**1.021**
10	20Newsgroup	1.147	1.310	**1.000**	1.653	**1.023**
10	BBCSport	3.105	1.628	**1.000**	3.541	**1.191**
10	Reuters-R8	1.010	1.632	**1.000**	1.727	**1.059**
15	WebKB	**1.000**	1.246	1.082	1.282	**1.012**
15	20Newsgroup	1.210	1.375	**1.000**	1.784	**1.205**
15	BBCSport	3.689	1.642	**1.000**	4.185	**1.000**
15	Reuters-R8	1.009	1.548	**1.000**	1.707	**1.010**

Table 2. Relative entropy

As we can see from Table 1 and Table 2, the results on the relative entropy show that our proposed method achieves significantly better results than the method of Fraj et al. [3] on the same data sets.

Going more into details, from Table 2 we can see that overall the TFIDF first and the LDA view second have the best results in term en entropy and are used as baseline for the relative entropy. We can see that for many data set our proposed method not only is close from the best entropy result, but that it achieves better results on average than the 3 original lda, skipgram and tfidf views, and always better results than the method from Fraj et al.

Since each view may hold its own truth, it is only logical that we rarely achieve fusion results that are better than all original view. This is a common problem in multi-view clustering [30] and should be considered as normal. Regardless, it is worth mentioning that our proposed method still achieves the best results in the case of the BBCSport data set with 15 clusters in terms of relative entropy.

k	lda	skipgram	tfidf	fraj	proposal
5	1.129	1.083	**1.001**	1.446	**1.020**
10	1.112	1.094	**1.006**	1.283	**1.010**
15	1.119	1.096	**1.000**	1.263	**1.020**

Table 3. Average relative purity

k	dataset	lda	skipgram	tfidf	fraj	proposal
5	WebKB	1.006	1.037	**1.000**	1.344	**1.011**
5	20Newsgroup	1.087	1.024	**1.000**	1.513	**1.045**
5	BBCSport	1.422	1.139	**1.000**	1.822	**1.017**
5	Reuters-R8	**1.000**	1.131	1.005	1.106	**1.005**
10	WebKB	1.020	1.110	**1.020**	1.200	**1.000**
10	20Newsgroup	1.049	1.121	**1.000**	1.226	**1.026**
10	BBCSport	1.353	1.066	**1.000**	1.586	**1.013**
10	Reuters-R8	1.027	1.077	**1.005**	1.119	**1.000**
15	WebKB	1.013	1.132	**1.000**	1.209	**1.027**
15	20Newsgroup	1.053	1.098	**1.000**	1.279	**1.054**
15	BBCSport	1.368	1.045	**1.000**	1.436	**1.000**
15	Reuters-R8	1.042	1.109	**1.000**	1.127	**1.001**

Table 4. Relative purity

From Table 3 and 4, we can see that the results in term of purity are the same than the one we had with entropy, thus enabling us to affirm that our proposed method proved superior than the one of Fraj et al. on all data sets regardless of the number of clusters.

Like for entropy, we can see that we rarely achieve the best results among views, but that we still do better than the average of the 3 original views, and from Table 3 we can see that our algorithm remains very competitive even when compare to the best view.

The best performances of our proposed algorithm for relative purity are for the BBCSport data set with 15 clusters, Reuters-R8 with 15 clusters and WebbKB with 10 clusters. For all 3 cases, we not only get better results than other methods in the literature, but we also do better than the best views in term of relative purity.

5 Conclusion and future works

We have presented a new clustering fusion method applied to the case of multi-view text corpus clustering. Our method was applied to 4 data sets that are very common in the literature (20Newsgroup, Reutors-R8, WebKB and BBCSport) and has proved to be competitive with state of the art methods. Unlike previously proposed methods, our algorithm relies on the notion of Kolmogorov complexity and information compression thus giving it a solid theoretical background on how to best fusion the clustering partitions.

In our future works, we plan on coupling our proposed method with existing collaborative method so that we could have a collaborative step first, and a merging step then. We hope that doing so may help to detect incompatible or noisy views, but could also ease the merging process by creating closer partition with collaborative clustering before hand. Other possible extensions of this work include applications on merging multi-view clustering partitions in fields other than text mining and natural language processing.

References

1. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. Journal of machine Learning research **3**(Jan), 993–1022 (2003)
2. Cornuéjols, A., Wemmert, C., Gançarski, P., Bennani, Y.: Collaborative clustering: Why, when, what and how. Information Fusion **39**, 81–95 (2018)
3. Fraj, M., HajKacem, M.A.B., Essoussi, N.: Ensemble method for multi-view text clustering. In: Computational Collective Intelligence - 11th International Conference, ICCCI 2019, Hendaye, France, September 4-6, 2019, Proceedings, Part I. pp. 219–231 (2019). https://doi.org/10.1007/978-3-030-28377-3_18
4. Fred, A.L., Jain, A.K.: Combining multiple clusterings using evidence accumulation. IEEE transactions on pattern analysis and machine intelligence **27**(6), 835–850 (2005)
5. Ghosh, J., Acharya, A.: Cluster ensembles. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery **1**(4), 305–315 (2011)
6. Greene, D., Cunningham, P.: A matrix factorization approach for integrating multiple data views. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases. pp. 423–438. Springer (2009)
7. Hussain, S.F., Mushtaq, M., Halim, Z.: Multi-view document clustering via ensemble method. Journal of Intelligent Information Systems **43**(1), 81–99 (2014)
8. Janssens, F., Glänzel, W., De Moor, B.: Dynamic hybrid clustering of bioinformatics by incorporating text mining and citation analysis. In: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 360–369. ACM (2007)
9. Li, T., Ogihara, M., Ma, S.: On combining multiple clusterings. In: Proceedings of the thirteenth ACM international conference on Information and knowledge management. pp. 294–303. ACM (2004)
10. Liu, H., Zhao, R., Fang, H., Cheng, F., Fu, Y., Liu, Y.Y.: Entropy-based consensus clustering for patient stratification. Bioinformatics **33**(17), 2691–2698 (2017)
11. Liu, X., Glänzel, W., De Moor, B.: Hybrid clustering of multi-view data via tucker-2 model and its application. Scientometrics **88**(3), 819–839 (2011)
12. Liu, X., Ji, S., Glänzel, W., De Moor, B.: Multiview partitioning via tensor methods. IEEE Transactions on Knowledge and Data Engineering **25**(5), 1056–1069 (2012)
13. Liu, X., Yu, S., Moreau, Y., De Moor, B., Glänzel, W., Janssens, F.: Hybrid clustering of text mining and bibliometrics applied to journal sets. In: Proceedings of the 2009 SIAM International Conference on Data Mining. pp. 49–60. SIAM (2009)
14. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems. pp. 3111–3119 (2013)
15. Murena, P., Sublime, J., Matei, B., Cornuéjols, A.: An information theory based approach to multisource clustering. In: IJCAI. pp. 2581–2587. ijcai.org (2018)
16. Rashidi, F., Nejatian, S., Parvin, H., Rezaie, V.: Diversity based cluster weighting in cluster ensemble: an information theory approach. Artificial Intelligence Review pp. 1–28 (2019)
17. Rissanen, J.: Modeling by shortest data description. Automatica **14**(5), 465 – 471 (1978)
18. Romeo, S., Tagarelli, A., Ienco, D.: Semantic-based multilingual document clustering via tensor modeling (2014)

19. Salton, G., Buckley, C.: Term-weighting approaches in automatic text retrieval. Information processing & management **24**(5), 513–523 (1988)
20. Salton, G., Wong, A., Yang, C.S.: A vector space model for automatic indexing. Communications of the ACM **18**(11), 613–620 (1975), the paper where vector space model for IR was introduced
21. Strehl, A., Ghosh, J.: Cluster ensembles—a knowledge reuse framework for combining multiple partitions. Journal of machine learning research **3**(Dec), 583–617 (2002)
22. Topchy, A., Jain, A.K., Punch, W.: Clustering ensembles: Models of consensus and weak partitions. IEEE transactions on pattern analysis and machine intelligence **27**(12), 1866–1881 (2005)
23. Wallace, C.S., Boulton, D.M.: An information measure for classification. The Computer Journal **11**(2), 185–194 (1968). https://doi.org/10.1093/comjnl/11.2.185
24. Wu, J., Liu, H., Xiong, H., Cao, J., Chen, J.: K-means-based consensus clustering: A unified view. IEEE transactions on knowledge and data engineering **27**(1), 155–169 (2014)
25. Xie, X., Sun, S.: Multi-view clustering ensembles. In: International Conference on Machine Learning and Cybernetics, ICMLC 2013, Tianjin, China, July 14-17, 2013. pp. 51–56 (2013). https://doi.org/10.1109/ICMLC.2013.6890443
26. Yi, J., Yang, T., Jin, R., Jain, A.K., Mahdavi, M.: Robust ensemble clustering by matrix completion. In: 2012 IEEE 12th international conference on data mining. pp. 1176–1181. IEEE (2012)
27. Yu, S., Moor, B., Moreau, Y.: Clustering by heterogeneous data fusion: framework and applications. In: NIPS workshop (2009)
28. Zamora, J., Allende-Cid, H., Mendoza, M.: Distributed clustering of text collections. IEEE Access **7**, 155671–155685 (2019)
29. Zhao, Y., Karypis, G.: Criterion functions for document clustering: Experiments and analysis. , Department of Computer Science, University of Minnesota, Technical Report TR 01-40 (2001)
30. Zimek, A., Vreeken, J.: The blind men and the elephant: on meeting the problem of multiple truths in data from clustering and pattern mining perspectives. Machine Learning **98**(1-2), 121–155 (2015)