Noname manuscript No.
(will be inserted by the editor)

Multi-view Clustering for Social-Based Data

Iain J. Cruickshank · Kathleen M. Carley

Received: date / Accepted: date

Abstract Insert your abstract here. Include keywords, PACS and mathematical subject classification numbers as needed.

Keywords First keyword · Second keyword · More

1 Introduction

Your text comes here. Separate text sections with

2 Background Research

3 Proposed Method

In this section I will outline two new techniques that modify the technique of Intermediate Paradigm Modularity Multi-view Clustering (IPMMC) that was proposed in the preceding chapter. The IPMMC method uses standard network modularity, which is known to have some issues with being an objective function used in clustering graphs, and considers all views as equal in

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship (DGE 1745016), Department of Defense Minerva Initiative (N00014-15-1-2797), and Office of Naval Research Multidisciplinary University Research Initiative (N00014-17-1-2675). Any opinion, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation, Department of Defense, or the Office of Naval Research.

Iain J. Cruickshank
CASOS Institute, Carnegie Mellon University
E-mail: icruicks@andrew.cmu.edu

Kathleen M. Carley
CASOS Institute, Carnegie Mellon University
E-mail: kathleen.carley@cs.cmu.edu

terms of their contributions to the multi-view clusters. As was described in the introduction to this chapter, one of the chief issues with modularity is its resolution limit. Two means of overcoming this resolution limit is to adopt a resolution parameter into the modularity function or to use one of the 'resolution free' density-based modularity functions. In the case of using a resolution parameter in the modularity function, recent work has shown what the optimal value of that resolution parameter should be for a given graph, given some assumptions about the nature of the graph [22]. This work has been even more recently extended to multi-layer graphs and been used to define what the optimal weight for each layer should be when using a modularity function with a resolution parameter [23]. Given these means of correcting for the deficiencies in the modularity function, this section will use these corrections within the same procedure as IPMMC to produce a better means of clustering multi-view, social-based data.

Common to both the proposed methods in this section and IPMMC is the framework of clustering multi-view data. In all of these techniques the general framework is to convert each view to a graph, if it is not a graph already, and then simultaneously cluster the graphs using modularity as the optimization function for the clustering. It should be noted that many of the multi-view clustering techniques employed in multi-view image clustering and multi-view clustering of medical data use the same first step of transforming all of the views into graphs [40], [39], [26]. However, none of these techniques, to the best of the author's knowledge, ever employ network modularity as the optimization function when clustering the multi-view networks. Within the realm of social-based data, only one technique in attributed graph clustering does a similar first step of converting the attributes to graphs [11]. The paper only considers a limited case of attributes in that there is only one view of categorical attributes. So, the common framework used by IPMMC and the two proposed methods here is new to social-based, multi-view data. The following figure, Figure 1, displays the overall framework.

The methodology proposed in this work for clustering multi-view. social-based data consists of two steps. The first step is to form graphs for every view of the data (A in Figure 1). This has been done by the computationally quick procedure of using a symmetric k-Nearest Neighbor graph, where k is the square root of the number of vertices, $k = \lfloor \sqrt{n} \rfloor$, for modes that are not already networks or graphs [25]. While this method can be used for creating the graphs, there are many other methods for inferring or learning graphs from data which could easily be used in this methodology [2], [25]. Transforming the data into a graph space provides two important advantages for multi-view clustering. First, it allows for the definition of a clustering goodness function that is parsimonious to all combinations of views of social-based data. Second, graphs better preserve the latent manifold structure of the data which allows for better structure, like clusters, to emerge [25]. Having formed graphs of all of the views, the next step is to use a modularity-based clustering procedure to simultaneously cluster all of the views (B in Figure 1). Based on the results of the previous chapter, modularity-based clustering — especially in an

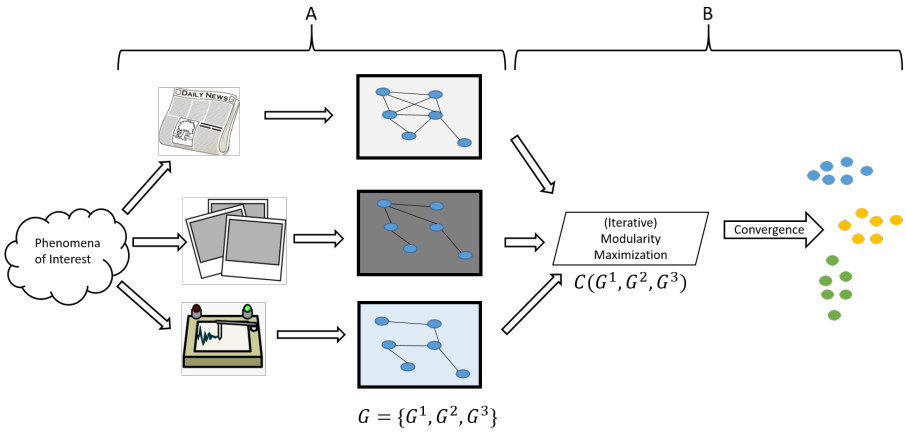

Fig. 1 Overview of the Multi-view Modularity Clustering method. the method works in two main steps. In step A, all of the views of the data are converted into graphs. In step B, these view graphs are collectively clustered using an iterative modularity maximization technique to produce clusters.

intermediate integration paradigm — gives some of the best clustering results. Having established the overall methodology, I will now describe in detail the means of performing modularity-based clustering.

3.1 Multi-view Modularity Clustering

For the first proposed technique, the weighted and resolution-corrected network modularity function will form the clustering quality function. that is to say, the quality of any clustering over the multi-view data is evaluated by:

$$Q = \sum_{v=1}^{m} w^v \sum_{ij \in E^v} [A_{ij}^v - \gamma^v \frac{deg(i)^v \times deg(j)^v}{2|E^v|}] \delta(C_i, C_j) \tag{1}$$

In effect, clustering with this function is optimizing Q, given the appropriate weight and resolution parameters. However, since the weight and resolution parameters are not always know *a priori* this function can be considered as having three, free parameters for determining the quality any given clustering: the cluster assignments, C, the view weights, w^v, and the view resolutions, γ^v. So, the clustering technique can be written as optimizing the following problem:

$$\begin{aligned} \max_{w,C,\gamma} \quad & \sum_{v=1}^{m} w^v \sum_{ij \in E^v} [A_{ij}^v - \gamma^v \frac{deg(i)^v \times deg(j)^v}{2|E^v|}] \delta(C_i, C_j) \\ s.t. \quad & C \text{ is integer} \\ & \gamma, w \in \mathbb{R} \end{aligned} \tag{2}$$

In order to solve this problem, I have adopted an alternating, iterative optimization procedure. In the first step, γ and w are fixed and C is adjusted by a modularity optimization procedure. In the second step, C is fixed and γ and w are adjusted. This alternating iterative procedure performs well in practice at finding good solutions and has certain other benefits for evaluating the quality of solutions, which will be seen later in the section on empirical testing.

For the first step of the procedure, a standard modularity maximizing clustering procedure (i.e. Louvain, Newman-Girvan, Leiden, etc.) can be used to find the the best values for C, given the resolutions and weights remain fixed. This then leaves the problem of how to optimally set the resolutions and weights, once the cluster assignments are fixed. To do so, I utilize analytically derived equations for producing the optimal values for the weights and resolutions, given a clustering. In order to set the the resolution parameter to an optimal value for a particular graph, the following function is used:

$$\gamma = \frac{\theta_{in} - \theta_{out}}{\log\theta_{in} - \log\theta_{out}} \tag{3}$$

where γ is the resolution parameter, and θ_{in} and θ_{out} are the propensities of having edges internal to clusters or external to clusters respectively. This function was derived by relating modularity maximization to the planted partition Stochastic Block Model [22], [23]. The intuition behind this function is that when there is a greater propensity to form edges internal rather than external to clusters that the resolution should be higher which would bias the function to find more tightly-knit and possibly smaller communities. Since this formulation relies on knowing the clusters to compute the θ values, it does not on first glance seem useful for actually clustering a graph. However, the function for computing the resolution has been used in an iterative fashion with modularity-based graph clustering to optimally cluster graphs in reasonably few (i.e. less than 20) iterations [22], [23]. So, it is possible to iteratively cluster and then update the resolution parameter to both find the optimal clustering of the graph as well as its appropriate resolution parameter.

The same means of computing the optimal resolution parameter can be extended to compute the optimal view weights. Pamfil et al. used the same derivation process of finding the optimal resolution parameters from the relation of Reichardt and Bornholdt modularity to planted partition Stochastic Block Models to obtain the optimal weights for each of the layers in a multiplex graph as:

$$w_v = \frac{\log\theta^v_{in} - \log\theta^v_{out}}{< \log\theta^v_{in} - \log\theta^v_{out} >_v} \tag{4}$$

where w_v is the weight given to a view, v, and θ^v_{in} θ^v_{out} are the propensities for edges to form internal to a cluster or external for the vth view, respectively. $< . >_v$ is the average across all of the views. The intuition behind this derivation is that those views with higher propensity to have edges internal to clusters versus having edges external to clusters relative to the average

across all views will have higher weights. So, a view with a better than average propensity to have edges internal to clusters should be weighted more heavily in the modularity calculation for clustering. It should be noted that this version of choosing the weights does not suffer from the same disconnected graph problem that the spectral version of updating the weights does. Once again, as with the resolution parameter, the weight parameter can be determined in an iterative fashion [23].

Having defined the new objective function as well as means of optimizing that function, a new algorithm can then be developed to cluster multi-view data. At a high level, the algorithm runs by first assigning starting resolution and weight parameters for every view (typically one) and then clustering the graph using a modularity maximization technique like Louvain or Leiden. These clusters are then used to compute new resolution and weight parameters. This process is repeated until the resolution and weight parameters no longer change. In the event that the resolution and weight parameters do not converge (which can happen in practice [23]), the clustering with the highest modularity value is chosen as the final clustering. The following psuedocode, Algorithm 1 describes the algorithm in detail.

Algorithm 1 Multi-view Modularity Clustering (MVMC)

input:
- Adjacency for each view: A^v
- Max number of iterations: $max\_iter = 20$
- Starting resolutions: $\gamma_1^v = 1, \forall v \in m$
- Starting weights: $w_1^v = 1, \forall v \in m$
- Convergence tolerance: $tol = 0.01$

output: Cluster assignments

$clustering^* \leftarrow None$
$modularity^* \leftarrow -\infty$
for $i = 1 : max\_iter$ **do**
  $clustering_i \leftarrow cluster(A, w_i, \gamma_i)$
  $modularity_i \leftarrow RBmodularity(A, clustering_i, w_i, \gamma_i)$
  $\theta_{in}, \theta_{out} \leftarrow calculate\_thetas(A, clustering_i)$
  $\gamma_{i+1}^v \leftarrow \frac{\theta_{in}^v - \theta_{out}^v}{\log\theta_{in}^v - \log\theta_{out}^v}, \forall v \in m$
  $w_{i+1}^v \leftarrow \frac{\log\theta_{in}^v - \log\theta_{out}^v}{<\log\theta_{in}^v - \log\theta_{out}^v>_v}, \forall v \in m$
  if $abs(\gamma_{i+1} - \gamma_i) < tol$ AND $abs(weights_{i+1} - weights_i) < tol$ **then**
    $clustering^* \leftarrow clustering_i$
    $modularity^* \leftarrow modularity_i$
    BREAK
  end if
  if $iter >= max\_iter$ **then**
    $best\_iteration \leftarrow argmax(modularity)$
    $clustering^* \leftarrow clustering[best\_iteration]$
    $modularity^* \leftarrow modularity[best\_iteration]$
  end if
end for
return $clustering^*$

The algorithm begins by initializing all the resolution parameters, γ_1^v, and weight parameters, w_1^v to one (or whatever the user may specify). The algorithm then goes on to cluster the view graphs, A^v, by a modularity maximization technique (i.e. Louvain, Leiden), $cluster()$, with the current resolution and weight settings. The output of this is then used to determine the propensities for internal edge formation θ_{in}^v, and external edge formation, θ_{out}^v for each view. These values are then used to update the resolution, γ^v, and weight parameters, w^v, for each of the views. If the new weight and resolution parameters are the same as the previous ones (within tolerance), the algorithm then exists and returns the final clustering. If the algorithm fails to converge to stable resolution and weight parameters, within the maximum number of iterations allowed, then the algorithm returns whichever clustering produced the highest modularity. Note, that modularity for this algorithm is the view-weighted, Reichardt and Bornholdt modularity, which incorporates the view modularities.

One of the important elements in the aforementioned algorithm, Algorithm 1, is the computation of the edge propensities, θ. In order to calculate these edge propensities, I follow the guidance outlined in previous works and assume edges form by a degree-corrected model [22], [23]. Given a degree corrected

model, the expected number of edges that occur internal to clusters is given by:

$$\begin{aligned} e_{in} &= \frac{1}{2}\sum_{c}\sum_{ij\in E}\theta_{in}\frac{deg(i)deg(j)}{2|E|}\delta(C_i,C_c)\delta(C_j,C_c) \\ &= \frac{\theta_{in}}{4|E|}\sum_{c}\kappa_c^2 \end{aligned} \tag{5}$$

where c is a cluster and $\kappa_c = \sum_i deg(i)\delta(C_i, C_c)$, or the sum of the degree of the vertices within cluster c. Using the observed number of edges internal to the clusters for the expected number of edges internal to clusters, e_{in}, this equation can then be used to calculate the propensity to form edges internally as:

$$\theta_{in} = \frac{e_{in}}{\sum_c \frac{\kappa_c^2}{4|E|}} \tag{6}$$

The same update can be derived for directed graphs as:

$$\theta_{in} = \frac{e_{in}}{\sum_c \frac{\sum_{ij\in c} deg_{in}(i)deg_{out}(j)}{2|E|}} \tag{7}$$

Similar to the propensity to form edges internally, the propensity to form edges externally can be derived from the expected external edges under the degree-corrected model as:

$$\theta_{out} = \frac{|E| - e_{in}}{|E| - \frac{\sum_c \kappa_c^2}{4|E|}} \tag{8}$$

and for directed graphs as:

$$\theta_{out} = \frac{|E| - e_{in}}{|E| - \frac{\sum_c \sum_{ij\in c} deg_{in}(i)deg_{out}(j)}{2*|E|}} \tag{9}$$

With these equations, and the assumption of edges forming by a degree-corrected model, the propensities for edges to occur internal or external to a cluster can be calculated. It is important to note, however, that these equations assume the *observed* edges internal or external to the clusters are equal to the *expected* edges internal or external to clusters. In practice, these values may actually differ. For example, if every vertex ends up in its own cluster, than no edges will be internal which will lead to the propensity internal term, θ_{in} to become zero, and the resolution and weight updates to fail. A similar problem can occur if all the vertices end up in one cluster and so no external edges occur. While at first glance these examples may appear to be edge cases, these problems can also occur in more important cases. For example, if the graph consists of a series of disconnected cliques. The optimal clusters in

this situation would then be to put all of the cliques within their own, separate clusters. However, this would result in there being no external edges, and so the resolution and weight updates would fail. In order to address these shortcomings, I have chosen to have a small value substitute for the propensities if there are either no internal edges and/or no external edges. So, while the observed edges can act as a proxy for the expected edges in most cases, the expected edges will always be a nonzero number. With these corrections, the algorithm for computing the propensity values is given in pseudocode by Algorithm 2.

Algorithm 2 Calculation of Edge Propensities

input:
- Adjacency for each view: A^v
- clustering: C

output: Internal and external edge propensities ($\theta_{in}, \theta_o ut$)

for v = 1:m **do**
  $e_{in} = 0$
  $\kappa^2 = []$
  for c = 1:—C— **do**
    $e_c = |E_c^v|$
    $e_{in} += e_c$
    if A^v is directed **then**
      $\kappa^2.append(\sum_{ij \in V_c^v} deg_{in}(i) deg_{out}(j))$
    else
      $\kappa^2.append((\sum_{i \in V_c^v} deg(i))^2)$
    end if
  end for
  if A^v is directed **then**
    if $e_{in} = 0$ **then**
      $\theta_{in}^v \leftarrow \frac{1}{|E^v|}$
    else
      $\theta_{in}^v \leftarrow \frac{e_{in}}{\sum \frac{\kappa^2}{2|E^v|}}$
    end if
    if $e_{in} == |E^v|$ **then**
      $\theta_{out}^v \leftarrow \frac{1}{|E^v|}$
    else
      $\theta_{out}^v \leftarrow \frac{|E^v| - e_{in}}{|E^v| - \sum \frac{\kappa^2}{2|E^v|}}$
    end if

else
if $e_{in} = 0$ **then**
$\theta^v_{in} \leftarrow \frac{1}{|E^v|}$
else
$\theta^v_{in} \leftarrow \frac{e_{in}}{\sum \frac{\kappa^2}{4|E^v|}}$
end if
if $e_{in} == |E^v|$ **then**
$\theta^v_{out} \leftarrow \frac{1}{|E^v|}$
else
$\theta^v_{out} \leftarrow \frac{|E^v| - e_{in}}{|E^v| - \sum \frac{\kappa^2}{4|E^v|}}$
end if
end if
end for
return $\theta_{in}, \theta_{out}$

The algorithm goes through each graph to calculate the propensities for each graph separately. For each graph, the algorithm begins by calculating the number of internal edges and the degree-corrected, null-model terms (i.e. κ^2)for each of the clusters. Then, the algorithm checks as to whether the graph is directed or undirected and whether there are no internal or external edges and then calculates the final propensities for that view graph, $\theta^v_{in}, \theta^v_{out}$. Once the propensities have been calculated for all of the view graphs, these are then returned.

4 Empirical Results

In this section I will use several benchmark, social-based data sets in order to empirically evaluate the hybrid paradigm techniques proposed in the previous section. The proposed techniques will be compared to several state of the art intermediate and late integration techniques at their ability to recover the given clusters from the benchmark data sets. The general set up of the empirical testing is as follows:

- Each view of each data set is clustered by an appropriate clustering technique.
- Each non-network view of the data set is transformed into a graph. The same graph is used across all methods that require graphs.
- The collection of each data set's clusters and graphs are then clustered by each multi-view technique twenty times. This is to account for the fact that many of the optimization routines for these techniques are stochastic.
- Each of the twenty results are then evaluated against the benchmark labels for each data set and the average performance is reported.

In the next section the data sets will be described in detail, including view cluster and view graph information. In the following section all of the techniques that are being used for comparison will be briefly described. Finally, the results of empirical testing will then be presented.

4.1 Data Set Information

For the empirical evaluation I used twelve different, social-based data sets. The data sets were chosen to encompass a wide array of scenarios that give rise to multi-view data. Since multi-view data can arise from many different types of social interactions, which can all be qualitatively different, it is important to see if there are any techniques that can work for social-based data generally or any that work for particular social scenarios. Since it is well established in clustering theory that different clustering functions will have different properties with inherent trade-offs, it is expected that certain certain types of clustering functions may work well for some data scenarios, but poorly for others [15], [30].

The data sets break into four types of scenarios that give rise to multi-view data: publication networks, social media, news stories, and multi-layer networks. Of the social media data sets, most of the data sets come from Twitter, however there is also representation of other social media sites as well. All but two of the data sets include both network and non-network views of the data. More traditional multi-view data sets of a text only data set and a multi-layer graph are also included in order to better evaluate the proposed methods across the full range of possible views arising from social-based data. The following table, Table 1, displays a summary of the different data sets.

Data set	Views	Number of Objects	Number of Clusters	Meaning of Clusters
Cora [29]	- Text (binary feature vector) - Network	2,708	7	General categories of the papers
CiteSeer [29]	- Text (binary feature vector) - Network	3,312	6	General categories of the papers
Flickr [10]	- Network - User tags	7,575	9	User Communities
BlogCatalog [10]	- Network - Blog keywords	5,196	6	User Communities
Wiki [38]	- Text (TF-IDF feature vector) - Network	2,405	19	General categories of the pages
3Sources [14]	- BBC (text term counts) - Reuters (text term counts) - The Guardian (text term counts)	169	6	News topic
AUCS [4]	- Network (lunch) - Network (Facebook) - Network (co-author) - Network (work) - Network (leisure)	61	9	Research group

Football [7] (Twitter)	- List - Text (term counts) - Network (follows) - Network (mentions) - Network (retweets)	248	20	Team membership
Olympics [7] (Twitter)	- List - Text (term counts) - Network (follows) - Network (mentions) - Network (retweets)	464	28	Sport affiliation
Politics IE [7] (Twitter)	- List - Text (term counts) - Network (follows) - Network (mentions) - Network (retweets)	348	10	Political party
Politics UK [7] (Twitter)	- List - Text (term counts) - Network (follows) - Network (mentions) - Network (retweets)	419	4	Political Party
Rugby [7] (Twitter)	- List - Text (term counts) - Network (follows) - Network (mentions) - Network (retweets)	854	8	National affiliation

Table 1: Benchmark, social-based data used in the empirical validation of the techniques. Most data sets consist of network and non-network modes of the data. An all text data set (3Sources) and an all network data set (AUCS) were also included in order to get a better empirical investigation across the types of social-based data that exists.

Since all of the multi-view methods evaluated in this section use either the clusters or the graphs, or both, from every view, the following tables summarize the view cluster qualities and the graphs form each of the views. The first table, Table 2, summarizes the clusters from each of the views across each of the data sets.

Data set	Mode	ARI	AMI	Without labels	Number of clusters	Best method to cluster	Actual number of clusters	Average ARI consensus between view clusters
Cora	network text	0.21 0.12	0.43 0.17	0.83 -0.02	107 7	Leiden LDA	7	0.05
CiteSeer	network text	0.1 0.11	0.25 0.14	0.89 0.002	466 6	Leiden LDA	6	0.05
Flickr	network attributes	0.09 0.56	0.11 0.56	0.25 -0.38	15 19	Leiden Leiden on kNN	9	0.06
BlogCatalog	network attributes	0.13 0.41	0.23 0.44	0.37 -0.04	8 6	Leiden LDA	6	0.04
Wiki	network text	0.2 0.38	0.29 0.45	0.77 -0.085	72 17	Leiden LDA	17	0.22
3Sources	BBC -text Reuters -text Guardian -text	0.16 0.01 0.12	0.34 0.23 0.24	0.02 0.05 0.1	6 6 6	Agglomerative Agglomerative Agglomerative	6	0.147

AUCS	lunch network	0.59	0.64	0.66	6	Leiden	9	0.247
	facebook network	0.27	0.21	0.34	33	Leiden		
	coauthor network	0.18	0.17	0.76	44	Leiden		
	work network	0.54	0.55	0.46	5	Leiden		
	leisure network	0.45	0.45	0.57	20	Leiden		
Football (Twitter)	list	0.4	0.56	-0.04	19	LDA	20	0.396
	text	0.02	0.08	-0.05	18	LDA		
	follows	0.49	0.74	0.45	11	Leiden		
	mentions	0.78	0.87	0.67	18	Leiden		
	retweets	0.59	0.72	0.69	29	Leiden		
Olympics (Twitter)	list	0.45	0.56	-0.06	28	LDA	28	0.497
	text	0.09	0.25	-0.13	27	LDA		
	follows	0.57	0.78	0.48	11	Leiden		
	mentions	0.86	0.91	0.8	22	Leiden		
	retweets	0.85	0.89	0.84	47	Leiden		
Politics Ire (Twitter)	list	0.16	0.16	-0.05	7	LDA	7	0.315
	text	0.15	0.23	-0.04	7	LDA		
	follows	0.9	0.85	0.37	4	Leiden		
	mentions	0.85	0.78	0.53	12	Leiden		
	retweets	0.75	0.7	0.66	49	Leiden		
Politics UK (Twitter)	list	0.81	0.79	0.05	4	LDA	5	0.402
	text	0.14	0.15	-0.03	5	LDA		
	follows	0.97	0.92	0.4	4	Leiden		
	mentions	0.67	0.65	0.29	13	Leiden		
	retweets	0.64	0.59	0.36	43	Leiden		
Rugby (Twitter)	list	0.33	0.51	-0.09	15	LDA	15	0.506
	text	0.17	0.3	0.08	14	LDA		
	follows	0.63	0.67	0.53	14	Leiden		
	mentions	0.36	0.61	0.67	19	Leiden		
	retweets	0.35	0.6	0.7	44	Leiden		

Table 2: Cluster properties and performances for each view of each of the data sets. Note, that in many cases the number and quality of clusters differs significantly between network and non-network modes of the data. Furthermore, while the network measure of clustering performance, Network Modularity, is generally indicative of the clustering performance on the network views, the Silhouette Index is not indicative of the performance of clustering the non-network views.

In order to evaluate how well the view clusters match the given, benchmark labels both the Adjusted Rand Index (ARI) and the Adjusted Mutual Information (AMI) were used to compare the cluster labels to the given labels [12], [36]. Generally speaking, there is a wide disparity of view cluster qualities relative to the benchmark labels, both within and between data sets. Some data sets like Cora and 3Sources have no particular view cluster well, while many of the Twitter data sets have at least one interaction network that does cluster well. The average pairwise ARI between the clusters between each of the views was also recorded. This average pairwise ARI represents a measure of consensus between the individual view clusterings. There is a wide disparity in this average ARI consensus between the data sets with the Twitter data sets having the highest values. These results indicate that there can be a wide disparity in the views in terms of their clusterability for recovering the benchmark labels. Based on the consensus principle of multi-view clustering,

it would be expected that those data sets with a greater disparity in their view clusterings should also have worse multi-view clusterings.

In order to evaluate the performance of the view clustering techniques in the absence of any ground-truth labels I have has to adopt two different measures. If the view is a graph, then I use network modularity [21] to evaluate clustering performance. If the view is other than a graph, I use the Silhouette Index to evaluate clustering performance [27]. Both the modularity and the silhouette score do not correlate well with a view's clustering performance across the data sets. Nor do they provide much insight into determining the clusterability of views within a data set. Using these measures, one cannot definitively tell the clusterability of the various views of the data sets. This would suggest that the common approach clustering data by trying various numbers of clusters and/or clustering algorithms and choosing that which delivers the best silhouette score may not be suitable for these data sets. Additionally, for many of the non-network views the silhouette score is negative, indicating overlapping clusters and poor clustering performance. For these reasons, whenever the clustering method requires the number of clusters be specified (which is most of the non-network clustering techniques) I choose to simplify the clustering procedure and just set the number of clusters to the benchmark number of clusters. As a result, there is often a disparity in number of clusters between the network and non-network modes. However, this disparity is also likely, at least in part, a function of the differences in clustering structure between views. For example, the AUCS multi-layer data set, which only has network views, has different numbers of clusters within each view as found by a network clustering procedure which automatically determines the number of clusters. Thus, there is not clustering consistency between the views of the same data set which is one source of potential difficulty in clustering these multi-view data sets.

In terms of clustering the individual views, I tried various types of commonly available clustering techniques and used those which produced the best results in terms of the benchmark labels. For the network views, the clustering technique used the Leiden modularity maximization technique [34]. For the text views either Ward Aggolmerative Clustering (using the actual number of clusters) of Latent Dirichlet Allocation and selecting the most probable cluster (again, using the actual number of clusters) produced the best clusters from the non-network modes.

The second table, Table 3, summarizes the graphs from each of the views across each of the data sets. For each of the the non-network views, a symmetric k Nearest Neighbor graph (kNN) was constructed from the features of the view. To construct the symmetric kNN, each object is connected to k of its nearest neighbors to produce a graph and then only the reciprocated edges are kept. So, $e_{uv} \in E$ iff $u \in kNN(v)$ and $v \in kNN(u)$. For weighted graphs, the lowest edge weight is used as the reciprocated edge weight. The symmetric kNN can be formed from the adjacency matrix of the original kNN by doing a element-wise minimum between the adjacency and its transpose, by $A_{sym} = min(A, A^T)$. A symmetric kNN is used as it has been found to better reveal

cluster structure in kNNs [28], [18], [19], [3]. For the value of k I used $k = \lceil \sqrt{n} \rceil$, which is generally regarded are producing effective graphs for clustering [18], [24].

Data set	Mode	Graph type	Graph density	Number of graph components	Number of graph isolates	Degree assorti-ativity	Clustering coefficient
Cora	network	directed	0.0007	78	0	-0.035	0.13
	text	undirected	0.008	1	0	0.404	0.14
CiteSeer	network	directed	0.0004	438	0	0.129	0.07
	text	undirected	0.007	4	3	0.378	0.13
Flickr	network	undirected	0.009	1	0	-0.217	0.33
	attributes	undirected	0.006	12	11	0.205	0.23
BlogCatalog	network	undirected	0.013	1	0	-0.01	0.12
	attributes	undirected	0.006	6	5	0.439	0.32
Wiki	network	directed	0.0029	45	0	0.0047	0.323
	text	undirected	0.012	98	97	0.534	0.48
3Sources	BBC -text	undirected	0.08	1	0	0.407	0.4
	Reuters -text	undirected	0.07	1	0	0.504	0.48
	Guardian -text	undirected	0.07	3	2	0.479	0.39
AUCS	lunch network	undirected	0.11	2	1	0.004	0.66
	facebook network	undirected	0.068	30	29	0.0027	0.283
	coauthor network	undirected	0.01	44	36	0.017	0.109
	work network	undirected	0.106	2	1	-0.213	0.63
	leisure network	undirected	0.048	16	14	-0.01	0.302
Football (Twitter)	list	undirected	0.046	1	0	0.51	0.69
	text	undirected	0.017	66	58	0.62	0.3
	follows	directed	0.06	2	1	-0.019	0.37
	mentions	directed	0.05	2	1	0.046	0.38
	retweets	directed	0.02	15	14	0.062	0.28
Olympics (Twitter)	list	undirected	0.03	2	0	0.49	0.56
	text	undirected	0.02	56	47	0.6	0.33
	follows	directed	0.05	2	1	-0.017	0.4
	mentions	directed	0.04	4	3	0.078	0.42
	retweets	directed	0.02	26	24	0.049	0.36
Politics IE (Twitter)	list	undirected	0.03	7	6	0.52	0.49
	text	undirected	0.02	58	56	0.5	0.36
	follows	directed	0.14	1	0	-0.082	0.48
	mentions	directed	0.05	6	5	-0.147	0.39
	retweets	directed	0.03	42	41	-0.152	0.34
Politics UK (Twitter)	list	undirected	0.03	26	21	0.48	0.47
	text	undirected	0.01	93	84	0.59	0.21
	follows	directed	0.16	2	1	-0.079	0.53
	mentions	directed	0.08	8	7	0.0207	0.33
	retweets	directed	0.04	39	38	0.067	0.26
Rugby (Twitter)	list	undirected	0.025	15	14	0.45	0.56
	text	undirected	0.01	201	186	0.69	0.29
	follows	directed	0.05	7	6	0.019	0.41
	mentions	directed	0.046	7	6	0.16	0.41
	retweets	directed	0.017	28	27	0.089	0.31

Table 3: Graph properties for all of the views across all of the data sets. As with the view clusterings in the previous table, there are often significant differences in topology between the the network view graphs and the non-network view graphs.

There is a high degree of variability of view graphs both between and within data sets. Typically, the graphs constructed from the non-network views have different topologies than the network view graphs. Across most of the data sets, the non-network view graphs are often denser, have higher degree assortiativity, are symmetric, and have fewer components and isolates. However, this pattern does not hold across all of the data sets, as most of the Twitter data sets reverse this pattern. All of these topological differences are partly related to the fact that the non-network views have their graphs constructed by a kNN procedure. However, these differences are not solely related to the network and non-network view differences. For example. the text only data set (3Sources) has differences in topology between the view graphs and all of these graphs were constructed by the same method with the same parameters. Thus, while there are distinct differences between the network and non-network views graphs, these differences can only partly a function of the graph learning technique for the non-network views. So, the differences in graph topologies between views in a data set is function of the differences in views and the processes that form the different views. Thus, it is unlikely that the graphs from every view will line up topologically and this lack of alignment will contribute to the difficulty of producing one set of cluster labels for all of the graphs.

The graphs in these social-based data sets have topological features that can present difficulties for clustering that other data scenarios do not have. For example, many of the data sets will have one or more view graphs that feature isolates. These isolates are normal in social-based data scenarios; they often represent individuals that are not interacting in a particular view. For example not all users will engage in re-tweeting or re-posting behavior. This is in contrast to other data scenarios, like image processing, where there will never be isolates since all images interact in all of the views. Additionally, the graphs from social-based data can also be undirected, directed, weighted, or unweighted all within the same data set. In other data scenarios all of the view graphs are typically of only one type (i.e. unweighted and undirected). So, social-based, multi-view data scenarios present new challenges for multi-view clustering techniques as they have much more difficult graph topologies to cluster and have a large degree of heterogeneity between views of the data than do other multi-view data scenarios.

4.2 Compared Methods

In this section all of the methods that will be tested across the social-based data sets are described. Several state-of-the-art late and intermediate integration techniques that are used in the empirical tests are described in this section. The is section will also detail the user-set parameter settings for all of the proposed techniques form the methods section of this chapter.

The intermediate and late integration multi-view clustering techniques from other works I investigated are as follows:

Intermediate Integration Techniques

- **CG**: Consistent Graph [17] aims to learn both consistent and inconsistent graphs across graphs from from multiple views of the data. The method does this using graph representations of each of the views of the data and a unified optimization framework. The final consistent graph output is clustered via spectral clustering for the final clusters.
- **ETL-MSC**: Essential Tensor Learning for Multi-view Spectral Clustering [37] uses a regularized tensor decomposition method along with some other tensor operations to get a lower-dimensional representation of the data. Each mode of the data is converted into a graph and then each graph into a probability transition matrix, which are then stacked to produce a tensor. The output of the decomposition is then clustered by spectral clustering for the final clusters.
- **NFC-CCE**: Network Fusion for Composite Community Extraction [6] uses a Non-Negative Matrix Factorization for a graph of each view and across each of the views in a combined model to produce a low-dimensional cluster indicator matrix which can be clustered via k-Means. Each view of the data is converted into a graph for input into this method. I used $\alpha = 0.0001$ which was determined by hand tuning the parameter to best fit the social-based data.
- **SPSL**: Self-Paced Spectral Learning [41] iteratively clusters a weighted combination of the graphs from each of the views, and updates the weight for each view's graphs depending on how well that particular view graph does with the clusters obtained by partitioning the weighted combination of all of the view graphs.
- **ResK**: RESCAL Factorization with K-Means Clustering [35] creates a tensor from all of the graphs from each of the views and then uses RESCAL tensor factorization of this tensor to produce a cluster indicator matrix. The method then clusters the cluster indicator matrix by k-Means to produce the final clusters.
- **SFI**: Spectral Feature Integration [33] computes the random-walk normalized Laplacian for each of the view graphs. It then takes the top eigenvectors from each view's Laplacians, concatenates them, and finally uses SVD on the concatenated matrix to produce the final set of feature vectors. It then clusters those feature vectors via k-Means to get the final clusters. I each decomposition, only k vectors were taken forward from the decomposition.
- **MSIM_C**: In this technique [13] row and column similarity matrices are learned in a co-learning set up, across all of the views. The row-similarity matrices are also clustered in every iteration to produce an co-association matrix, which is then used to enhance the row-similarity matrix. After a set number of iterations are run, the final row-similarity matrix is then clustered as a graph to produce the final clusters. It should be noted that this technique does not require each of the views to be converted to a view graph, however, in empirical tests I have found universally better

performance if it is given the view graphs instead of the raw data matrices from each view. Each test ran for four iterations, which was the suggested number of iterations from the paper.

Late Integration Techniques

- **CSPA+**: Cluster-based Similarity Partitioning Algorithm [31] computes the co-association matrix from all of the view clusterings. I have then incorporated a global-local pruning procedure and iterative refinement of the co-association matrix, both of which have been shown to improve the performance of co-association matrix-based methods [16], [20], [32]. Additionally, I also use modularity maximization techniques to cluster the co-association matrices due to their superior empirical performance in terms of accuracy and speed [16].
- **BGPA+**: Bipartite Graph Partitioning Algorithm [1], [5] is a cluster ensembling technique that forms a bipartite graph from every object and every cluster across all modal clusterings. This bipartite graph can then be clustered to get final cluster assignments. I improve upon existing algorithms by using a bipartite graph-specific clustering algorithm rather than a generic graph clustering algorithm. In particular, I use a BiLouvain algorithm for clustering the object-by-clusters bipartite graph [1].
- **LWMC**: Locally Weighted Meta-Clustering [9] is a cluster ensembling technique that clusters a specially-weighted cluster-to-cluster graph in order to produce meta-clusters which can then be used to determine the final cluster assignments. The weighting in the cluster-to-cluster graph is done by determining the joint entropy for each cluster with respect to every other cluster, across all modes, and represents how well that particular cluster is at capturing an actual cluster in the data.
- **LWBG**: Locally Weighted Bipartite Graph clustering [9] is a cluster ensembling technique which clusters a specially weighted bipartite graph of objects-to-clusters from all of the modes. The weighting in the object-to-cluster bipartite graph is done by determining the joint entropy for each cluster with respect to every other cluster, across all modes, and represents how well that particular cluster is at capturing an actual cluster in the data.
- **GP-MGLA**: Graph Partitioning with Multi-Granularity Link Analysis [8] is a cluster ensembling technique that creates a weighted graph of all of the objects and the clusters. The weight in the graph varies based on the type of the vertices of the edge (i.e. whether its a object to cluster or cluster to cluster). This weighted graph of objects and clusters is then clustered as a standard weighted graph. I set $\alpha = 0.5$ and $\beta = 2$ as was done in the method's original paper.
- **MCLA+** Meta-Clustering Algorithm [31] is a cluster ensembling technique that computes a weighted cluster-to-cluster graph in order to produce meta-clusters which can then be used to determine the final cluster

[1] https://scikit-network.readthedocs.io/en/latest/tutorials/clustering/louvain.html#Bigraphs

assignments. In this implementation, I use Louvain [2] instead of METIS to cluster the cluster-to-cluster graph and weight the cluster-to-cluster graph by the mutual information between clusters rather than the Jaccard Index, as these modifications produced better results.
– **DREC**: Dense Representation based Ensemble Clustering [42] uses a sparse graph learning technique to learn a similarity matrix from the cluster association matrices from each view. In order to improve performance, the algorithm also uses a slimming procedure, by which all of the objects that are clustered together in all of the views are grouped together and treated as a new object. The final similarity matrix is then clustered by spectral methods. I used a grid search over the values of $[0.001, 0.01, 1, 10, 100]$ for the lambda regularization value, and selected that one which lead to the best performance in ARI and AMI.

5 Discussion

Conflict of interest

The authors declare that they have no conflict of interest.

References

1. Boongoen, T., Iam-On, N.: Cluster ensembles: A survey of approaches with recent extensions and applications. Computer Science Review **28**, 1 – 25 (2018). DOI https://doi.org/10.1016/j.cosrev.2018.01.003. URL http://www.sciencedirect.com/science/article/pii/S1574013717300692
2. Brugere, I., Gallagher, B., Berger-Wolf, T.Y.: Network structure inference, a survey: Motivations, methods, and applications. ACM Comput. Surv. **51**(2), 24:1–24:39 (2018). DOI 10.1145/3154524. URL http://doi.acm.org/10.1145/3154524
3. Carey, C.: Graph construction for manifold discovery. Ph.D. thesis, University of Massachusetts Amherst (2017). URL https://people.cs.umass.edu/ ccarey/pubs/thesis.pdf
4. Celli, F., Lascio, F.M.L.D., Magnani, M., Pacelli, B., Rossi, L.: Social Network Data and Practices: the case of Friendfeed. In: International Conference on Social Computing, Behavioral Modeling and Prediction, Lecture Notes in Computer Science. Springer Berlin Heidelberg (2010)
5. Fern, X.Z., Brodley, C.E.: Solving cluster ensemble problems by bipartite graph partitioning. In: Proceedings of the Twenty-first International Conference on Machine Learning, ICML '04, pp. 36–. ACM, New York, NY, USA (2004). DOI 10.1145/1015330.1015414. URL http://doi.acm.org/10.1145/1015330.1015414
6. Gligorijević, V., Panagakis, Y., Zafeiriou, S.: Non-negative matrix factorizations for multiplex network analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence **41**(4), 928–940 (2019). DOI 10.1109/TPAMI.2018.2821146
7. Greene, D., Cunningham, P.: Producing a Unified Graph Representation from Multiple Social Network Views. arXiv e-prints arXiv:1301.5809 (2013)
8. Huang, D., Lai, J.H., Wang, C.D.: Combining multiple clusterings via crowd agreement estimation and multi-granularity link analysis. Neurocomputing **170**, 240 – 250 (2015). DOI https://doi.org/10.1016/j.neucom.2014.05.094. URL http://www.sciencedirect.com/science/article/pii/S0925231215005640. Advances on

[2] https://scikit-network.readthedocs.io/en/latest/tutorials/clustering/louvain.html

Biological Rhythmic Pattern Generation: Experiments, Algorithms and Applications Selected Papers from the 2013 International Conference on Intelligence Science and Big Data Engineering (IScIDE 2013) Computational Energy Management in Smart Grids
9. Huang, D., Wang, C., Lai, J.: Locally weighted ensemble clustering. IEEE Transactions on Cybernetics **48**(5), 1460–1473 (2018). DOI 10.1109/TCYB.2017.2702343
10. Huang, X., Li, J., Hu, X.: Label informed attributed network embedding. In: Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, WSDM '17, p. 731–739. Association for Computing Machinery, New York, NY, USA (2017). DOI 10.1145/3018661.3018667. URL https://doi.org/10.1145/3018661.3018667
11. Huang, Y., Wangg, H.: Consensus and multiplex approach for community detection in attributed networks. In: 2016 IEEE Global Conference on Signal and Information Processing (GlobalSIP), pp. 425–429 (2016). DOI 10.1109/GlobalSIP.2016.7905877
12. Hubert, L., Arabie, P.: Comparing partitions. Journal of Classification **2**(1), 193–218 (1985). DOI 10.1007/BF01908075. URL https://doi.org/10.1007/BF01908075
13. Hussain, S.F., Bashir, S.: Co-clustering of multi-view datasets. Knowledge and Information Systems **47**, 545–570 (2016). DOI 10.1007/s10115-015-0861-4. URL https://doi.org/10.1007/s10115-015-0861-4
14. Insight: 3 sources dataset (2009). URL http://mlg.ucd.ie/datasets/3sources.html
15. Kleinberg, J.: An impossibility theorem for clustering. In: Proceedings of the 15th International Conference on Neural Information Processing Systems, NIPS'02, p. 463–470. MIT Press, Cambridge, MA, USA (2002)
16. Lancichinetti, A., Fortunato, S.: Limits of modularity maximization in community detection. Physical Review E **84**, 066122 (2011). DOI 10.1103/PhysRevE.84.066122
17. Liang, Y., Huang, D., Wang, C.D.: Consistency meets inconsistency: A unified graph learning framework for multi-view clustering. In: 2019 IEEE International Conference on Data Mining (ICDM) (2019)
18. Maier, M., Hein, M., von Luxburg, U.: Optimal construction of k-nearest neighbor graphs for identifying noisy clusters. arXiv e-prints arXiv:0912.3408 (2009)
19. Maier, M., von Luxburg, U., Hein, M.: How the result of graph clustering methods depends on the construction of the graph. arXiv e-prints arXiv:1102.2075 (2011)
20. Mandaglio, D., Amelio, A., Tagarelli, A.: Consensus community detection in multilayer networks using parameter-free graph pruning. CoRR **abs/1804.06653** (2018). URL http://arxiv.org/abs/1804.06653
21. Newman, M.E.J.: Networks: An Introduction. Oxford University Press (2010)
22. Newman, M.E.J.: Community detection in networks: Modularity optimization and maximum likelihood are equivalent. arXiv e-prints arXiv:1606.02319 (2016)
23. Pamfil, A.R., Howison, S.D., Lambiotte, R., Porter, M.A.: Relating modularity maximization and stochastic block models in multilayer networks. CoRR **abs/1804.01964** (2018). URL http://arxiv.org/abs/1804.01964
24. Premachandran, V., Kakarala, R.: Consensus of k-nns for robust neighborhood selection on graph-based manifolds. In: 2013 IEEE Conference on Computer Vision and Pattern Recognition, pp. 1594–1601 (2013). DOI 10.1109/CVPR.2013.209
25. Qiao, L., Zhang, L., Chen, S., Shen, D.: Data-driven graph construction and graph learning: A review. Neurocomputing **312**, 336 – 351 (2018). DOI https://doi.org/10.1016/j.neucom.2018.05.084. URL http://www.sciencedirect.com/science/article/pii/S0925231218306696
26. Rappoport, N., Shamir, R.: Multi-omic and multi-view clustering algorithms: review and cancer benchmark. Nucleic Acids Research **46**(20), 10546–10562 (2018). DOI 10.1093/nar/gky889. URL https://doi.org/10.1093/nar/gky889
27. Rousseeuw, P.J.: Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. Journal of Computational and Applied Mathematics **20**, 53 – 65 (1987). DOI https://doi.org/10.1016/0377-0427(87)90125-7. URL http://www.sciencedirect.com/science/article/pii/0377042787901257
28. Ruan, J.: A fully automated method for discovering community structures in high dimensional data. In: 2009 Ninth IEEE International Conference on Data Mining, pp. 968–973 (2009). DOI 10.1109/ICDM.2009.141
29. Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., Eliassi-Rad, T.: Collective classification in network data. AI Magazine **29**(3), 93 (2008). DOI 10.1609/aimag.v29i3.2157. URL https://www.aaai.org/ojs/index.php/aimagazine/article/view/2157

30. anbd Shai Ben-David, S.S.S. (ed.): Understanding Machine Learning: From Theory to Algorithms. Cambridge University Press, New York, NY, USA (2012)
31. Strehl, A., Ghosh, J.: Cluster ensembles — a knowledge reuse framework for combining multiple partitions. J. Mach. Learn. Res. **3**, 583–617 (2003). DOI 10.1162/153244303321897735. URL https://doi.org/10.1162/153244303321897735
32. Tandon, A., Albeshri, A., Thayananthan, V., Alhalabi, W., Fortunato, S.: Fast consensus clustering in complex networks. Physical Review E **99**(4), 042301 (2019). DOI 10.1103/PhysRevE.99.042301
33. Tang, L., Liu, H.: Community Detection and Mining in Social Media. Morgan & Claypool Publishers (2010)
34. Traag, V.A., Waltman, L., van Eck, N.J.: From louvain to leiden: guaranteeing well-connected communities. Nature: Scientific Reports **9** (2019). DOI https://doi.org/10.1038/s41598-019-41695-z. URL https://www.nature.com/articles/s41598-019-41695-z
35. Verma, A., Bharadwaj, K.K.: Identifying community structure in a multi-relational network employing non-negative tensor factorization and ga k-means clustering. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery **7**(1) (2017). DOI 10.1002/widm.1196. URL https://doi.org/10.1002/widm.1196
36. Vinh, N.X., Epps, J., Bailey, J.: Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. J. Mach. Learn. Res. **11**, 2837–2854 (2010). URL http://dl.acm.org/citation.cfm?id=1756006.1953024
37. Wu, J., Lin, Z., Zha, H.: Essential tensor learning for multi-view spectral clustering. CoRR **abs/1807.03602** (2018). URL http://arxiv.org/abs/1807.03602
38. Yang, C., Liu, Z., Zhao, D., Sun, M., Chang, E.Y.: Network representation learning with rich text information. In: Proceedings of the 24th International Conference on Artificial Intelligence, IJCAI'15, p. 2111–2117. AAAI Press (2015)
39. Yang, Y., Wang, H.: Multi-view clustering: A survey. Big Data Mining and Analytics **1**(2), 83–107 (2018)
40. Ye, F., Chen, Z., Qian, H., Li, R., Chen, C., Zheng, Z.: New approaches in multi-view clustering. Recent Applications in Data Clustering (2018). DOI 10.5772/intechopen.75598. URL https://www.intechopen.com/books/recent-applications-in-data-clustering/new-approaches-in-multi-view-clustering
41. Yu, H., Lian, Y., Zong, L., Tian, L.: Self-paced learning based multi-view spectral clustering. In: 2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI), pp. 6–10 (2017). DOI 10.1109/ICTAI.2017.00013
42. Zhou, J., Zheng, H., Pan, L.: Ensemble clustering based on dense representation. Neurocomputing **357**, 66 – 76 (2019). DOI https://doi.org/10.1016/j.neucom.2019.04.078. URL http://www.sciencedirect.com/science/article/pii/S0925231219306794