Hierarchical Tensor Clustering for Multiple Graphs Representation

Karima Boutalbi, Rafika Boutalbi, Hervé Verjus, Kave Salamatian

Hierarchical Tensor Clustering for Multiple Graphs Representation

Karima Boutalbi
Université Savoie Mont Blanc
Cegedim Business Services
Annecy, France
karima.boutalbi@univ-smb.fr

Rafika Boutalbi
Aix-Marseille University
Marseille, France
rafika.boutalbi@lis-lab.fr

Hervé Verjus
Université Savoie Mont Blanc
Annecy, France
herve.verjus@univ-smb.fr

Kave Salamatian
Université Savoie Mont Blanc
Annecy, France
kave.salamatian@univ-smb.fr

ABSTRACT

Graph clustering is a challenging task, especially when there is a hierarchical structure. The availability of multiple graphs (or relational graphs), in the multi-graph setting, provides additional information that can be leveraged to improve clustering results. This paper aims to develop a new hierarchical clustering algorithm for multi-graphs, the HTGM algorithm. This algorithm represents the set of graphs in the multi-graph as a 3-way tensor, and maximizes a modularity measure, extending the modularity-based graph clustering algorithm to multi-graphs and tensor structures. We evaluate the proposed algorithm over synthetic and real-world datasets and show the effectiveness of the proposed algorithm by benchmarking it to alternative clustering algorithms.

CCS CONCEPTS

• **Unsupervised learning**; • **Clustering** → *Graph clustering*; • **NLP** → Word embedding; • **Representation learning** → Tensor;

KEYWORDS

Hierarchical clustering, Tensor, Graphs, Data representation.

ACM Reference Format:
Karima Boutalbi, Rafika Boutalbi, Hervé Verjus, and Kave Salamatian. 2024. Hierarchical Tensor Clustering for Multiple Graphs Representation. In *Companion Proceedings of the ACM Web Conference 2024 (WWW '24 Companion), May 13–17, 2024, Singapore, Singapore.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3589335.3651519

1 INTRODUCTION

Graphs are a generic way of modeling relations and interactions, represented by edges with a given weight or distance, between entities represented as vertices (or nodes). They can capture complex interactions into a relatively simple framework. Nonetheless, in a larger number of settings, the relations between the set of vertices can be different considering several aspects, where each aspect is defined by a specific graph. This results in multi-graphs, where we have several graphs over the same set of vertices [19].

Clusters or communities are generally defined as a set of nodes that are densely connected internally and loosely connected to external nodes. Graph clustering is known to be a challenging problem and several classes of approaches have been proposed in the literature to implement it, *e.g.*, multi-level [9, 12], spectral [13, 14], model-based graph clustering [2, 11], and modularity-maximization methods [1, 3]. All of these approaches involve solving instances of NP-hard problems that are approximated through heuristics.

Modularity maximization techniques aim at splitting the network into groups that have large positive modularity values. The modularity of a cluster is defined as, the difference between the fraction of edges that fall within the given cluster, with the expected fraction if vertices were connected randomly. Despite the fact that modularity-maximization is NP-hard, it has become a very popular graph-clustering technique, because of its ability to auto-detect the optimal number of clusters [16], its applicability to large graphs because relatively high speed of heuristics [3], and high quality of clustering results in practice [10].

When we deal with multi-graphs, three information fusion's frameworks can be considered: the *a priori*, the *a posteriori*, and the joint fusion. In the *a priori* approach, the multiple graphs are merged into a single graph, *e.g.*, by defining edges in the resulting graph as a weighted sum of edges in individual graphs. Thereafter a classical single graph clustering approach is applied to the final graph. In the *a posteriori* approach, each individual graph in the multi-graph is clustered separately using a single graph clustering approach, and the resulting clusters are merged by a consensus mechanism. The joint clustering approach implements a global clustering over all graphs, by optimizing a global clustering optimization function. This last approach needs new clustering techniques as single graph clustering algorithms are not applicable to it. The literature reports several extensions of spectral clustering techniques to joint clustering of multi-graphs [19]. Nonetheless, despite their popularity and desirable properties, modularity-maximization approaches have not been extended to multi-graphs. This paper aims to fill this gap and proposes modularity-maximization heuristics for multi-graphs that lead to fast and robust hierarchical graph clustering.

WWW '24 Companion, May 13–17, 2024, Singapore, Singapore

ACM ISBN 979-8-4007-0172-6/24/05
https://doi.org/10.1145/3589335.3651519

This new multi-graph clustering approach is named Hierarchical Tensor-based Multi-Graph Modularity (HTGM), and like [4] it leverages modularity-maximization over a multi-graph, *i.e.*, achieving a high value of modularity with P levels of hierarchy (see figure 1). Combining the different graphs HTGM is fusioning connectivity information from different graphs to provide a more robust and higher modularity clustering.

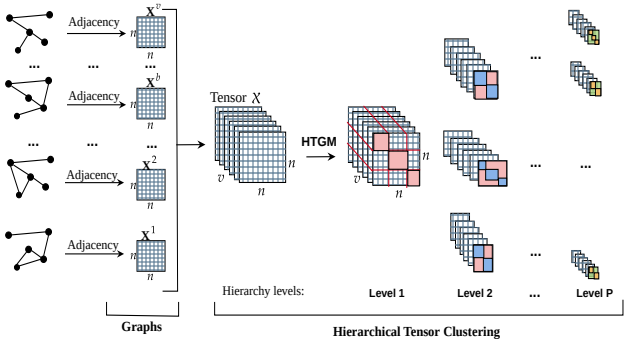

Figure 1: Goal of the proposed Hierarchical Tensor-based Graph Modularity (HTGM) approach.

In this paper, we develop a new algorithm HTGM for the Hierarchical clustering of multi-graphs based on tensor representation and graph modularity and adapt it to extract the hierarchical structure of clusters. We evaluate HTGM clustering on several real-world datasets, by comparing it with state-of-the-art techniques in the literature for hierarchical graph clustering.

2 HIERARCHICAL TENSOR-BASED GRAPH MODULARITY

As explained before we represent a multi-graph as a three-way tensor, $\mathcal{X} = [\mathbf{x}_{ij}] \in \mathbb{R}^{n \times n \times v}$, where each $n \times n$ slices is the weighted adjacency matrix of one of the sub-graphs inside the multi-graph, as shown in Figure 1. In the forthcoming, scalars are represented in lowercase letters *e.g.*, x, a vector by a bold lowercase letter *e.g.*, $\mathbf{x}$, a matrix a bold capital letters *e.g.*, $\mathbf{X}$, and a tensor is denoted by bold capital Euler letters e.g. $\mathcal{X}$. x^b_{ij} represents the entry (i, j) of the graph b, which is also the entry (i, j, b) of the 3D tensor representation $\mathcal{X}$.

Hereafter, we tackle the tensor hierarchical clustering problem by maximizing a modularity-based criterion at each level $\ell = 1...p$ of the hierarchy. The cluster k at hierarchy level l is defined through a membership index $z^\ell_{ik} = 1$ when vertex i belongs to cluster k. The modularity of the cluster k in a graph b is defined as:

$$\sum_{i,j=1}^{n} \left(x^b_{ij} - \frac{x^b_{i.} x^b_{.j}}{x^b_{..}} \right) z_{ik} z_{jk} \tag{1}$$

The value $x^b_{i.} = \sum_j x^b_{ij}$ is the degree of node i in the graph b, and $x^b_{..} = \sum_{ij} x^b_{ij}$. We wish to maximize the sum of modularity over clusters, and all graph in the multi-graph. This results into the following objective function to maximize:

$$Q(\mathcal{X}, \mathbf{Z}_\ell) = \sum_{b=1}^{v} \frac{1}{x^b_{..}} \sum_{i,j=1}^{n} \sum_{k=1}^{g_\ell} (x^b_{ij} - \frac{x^b_{i.} x^b_{.j}}{x^b_{..}}) z^\ell_{ik} z^\ell_{jk}. \tag{2}$$

$\mathbf{Z}_\ell \in \mathbb{R}^{n \times g_\ell}$ is the graph partition matrix, *i.e.*, z^ℓ_{ik} of $\mathbf{Z}_\ell = 1$, if vertex i belongs to the cluster k of level l of the hierarchy, and 0 otherwise; $b \in \{1, \ldots, v\}$ is the graph index, and the third dimension of the tensor.

An iterative approach can maximize the above objective function. At iteration $(t + 1)$, we implement the following update to the membership function:

$$z^{\ell,(t+1)}_{ik} = \begin{cases} 1 & \text{if } k = \underset{1 \le k \le g_\ell}{\arg\max} \sum_{b=1}^{v} \frac{1}{x^b_{..}} \sum_{i=1}^{n} \sum_{k=1}^{g_\ell} (x^b_{ik} - \frac{x^b_{i.} x^b_{k}}{x^b_{..}}) z^{\ell,(t)}_{ik} \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

that adds the vertex i to the cluster k maximizing the global modularity. The iteration continues until the global modularity can still increase. As the iterations increase the global modularity that is bounded from above, the iteration converges toward a stable point. We leverage on the above iterative process to propose the HTGM algorithm. As the global modularity integrates all sub-graphs in the multi-graph, it implements a joint clustering.

We present in Algorithm 1 the details of the iterative method at the core of the HTGM algorithm. The algorithm is considered to have converged at level ℓ when the difference between the criterion value $Q(\mathcal{X}, \mathbf{Z}_\ell)$ at iteration t and $t + 1$ is less than ϵ.

Algorithm 1: HTGM

Input: $\mathcal{X}$: Tensor, $\mathbf{g} \in \mathbb{R}^p$: vector of cluster number for each level ℓ.
(1) $C = [\,]$ // An empty clustering vector
(2) $Hclusters = [\,]$ // Clustering vector of each level
(3) **for** $\ell \leftarrow 1$ *to* p **do**
  (3.1) $Clust_\ell = [\,]$ // Clustering vector of level ℓ
  for $c \leftarrow 1$ *to* g_ℓ **do**
    (3.2) **Initialization:** $\mathbf{Z}^{(0)}_\ell$ randomly at $t = 0$ **repeat**
      (3.2.1) Compute $Q(\mathcal{X}, \mathbf{Z}^{(t)}_\ell)$
      (3.2.2) Compute $\mathbf{Z}^{(t+1)}_\ell$ maximizing $Q(\mathcal{X}, \mathbf{Z}^{(t)}_\ell)$ using (3)
      (3.2.3) Compute $Q(\mathcal{X}, \mathbf{Z}^{(t+1)}_\ell)$
    until *Convergence* $Q(\mathcal{X}, \mathbf{Z}^{(t+1)}_\ell) - Q(\mathcal{X}, \mathbf{Z}^{(t)}_\ell) < \epsilon$;
    $\mathbf{z} \leftarrow \arg_k \mathbf{Z}^{(t+1)}_\ell$
    $Clust_\ell$.append($\mathbf{z}$)
  3.3 $Hclusters$.append($Clust_\ell$)
Return $Hclusters$, $\forall \ell = 1 \ldots p \; Q(\mathcal{X}, \mathbf{Z}_\ell)$

3 EXPERIMENTS

3.1 Evaluation settings

In order to evaluate the HTGM algorithm, we are using real-world datasets for the text clustering task. We compare HTGM with seven state-of-the-art clustering algorithms applicable to graphs and tensors, namely hierarchical NMF [17] (H − NMF), hierarchical ITCC [6] (H − ITCC), hierarchical CoclustMod [1] (H − CoclustMod), hierarchical SPLBM [2] (H − SPLBM), hierarchical PARAFAC (H − PARAFAC), hierarchical TUCKER decomposition (H − TUCKER), and hierarchical TSPLBM [5] (H − TSPLBM). We have extended the above described clustering algorithms to create hierarchical clusters. The extension gets as input the data representation at level $\ell = 0$ which contains all vertices) and the number of clusters at each hierarchy level $\ell = 1 \ldots p$. We run the clustering method on all clusters of the level ℓ to generate the clusters at level $\ell + 1$. The process is iterated again until the last level. We show the details of the proposed extension

in Algorithm 2. The evaluation will try to answer the following three questions:

Algorithm 2: `Hierarchize – Algorithm`

Input: $\mathcal{X}$: Matrix or Tensor, `Algo`: Algorithm , $\mathbf{g} \in \mathbb{R}^p$: vector of cluster number for each level ℓ.
Initialization: $HierarchicalLabels$ =[] ;
(1) Run algorithm `A`$(\mathcal{X}, nClusters = g_1)$;
(2) Generate the clustering vector C from step (1) ;
(3) $HierarchicalLabels$.append(C) (4)**for** $K \leftarrow g_2$ *to* g_ℓ **do**
  (4.1)$LabelH$ =[] (4.2) **for** $C_i \in C$ **do**
    (4.2.1) $\mathcal{X}_H = \mathcal{X}[C == C_i]$;
    (4.2.2) Run algorithm `Algo`$(\mathcal{X}_H, nClusters = K)$;
    (4.2.3) Generate the clustering vector C_H;
    (4.2.4) $LabelH$.append(C_H)
  (4.3) $HierarchicalLabels$.append($LabelH$) ;
  (4.4) $C \leftarrow LabelH$;
Return HierarchicalLabels

- **Q1-** ***What is the difference between the proposed*** `HTGM` ***and state-of-the-art tensor decomposition and clustering approaches in terms of the hierarchical clustering performance?*** We compared the `HTGM` algorithm with the developed hierarchical version (using algorithm 2) of PARAFAC [7], TUCKER decomposition and TSPLBM [5].
- **Q2-*Do* a posteriori *fusion* have a better hierarchical performance than the joint clustering methods, *e.g.,* `HTGM`?** To answer this question, we used the `ClusterEnsembles` [18][1] consensus algorithm (Algorithm 3) which computes the consensus between different single graph clustering outcomes. We have used this consensus along all feature matrix and graph clustering algorithms.
- **Q3-*What are the* `HTGM` *clustering performances compared to alternative methods at the leaf level?*** For this purpose we added a Neural Network alternative, `BERTopic` algorithm for text clustering [8] and its hierarchical version (see documentation[2]).

Algorithm 3: `Consensus – Algorithm`

Input: $\mathcal{X}$: Various data representation Matrices $1 \ldots v$, `Algo`: Clustering Algorithm , g: Number of clusters
(1) **Initialization:** $AllLabels$ =[] (2) **for** $\mathbf{x}_b$ *in* $\mathcal{X}$ **do**
  (2.1) Run the clustering algorithm `Algo`$(\mathbf{x}_b, nClusters = g)$;
  (2.2) Generate the clustering vector C_b using `Algo` ;
  (2.3) $AllLabels$.append(C_b);
(3) Run the consensus algorithm `ClusterEnsembles`($AllLabels$) that generates the consensus clustering vector $ConsenusLabel$;
Return $ConsenusLabel$

We used three metrics for evaluation, namely the Accuracy (ACC), the Purity, and the Normalized Mutual Information (NMI) which are widely used for clustering task evaluation [18]. All included clustering algorithms are run 30 times to vary the initialization, and their average scores are compared.

.

3.2 Evaluation results

In order to answer all research questions, we are using three real-world benchmark datasets coming from text clustering task where the ground-truth partitions are known: *DBpedia*[3], *Yelp* [4], and *GitHub-AI-Bio* [5] dataset. Each one of these datasets has a multi-level hierarchy. The datasets are described in Table 1. We used five text embedding methods, namely, Bow (Bag-of-word), Skipgram, XLNET[20], and Sentence-Transformers (S-BERT) [15]. The feature size of each dataset is presented in Table 1.

Table 1: Description of textual datasets.

		Documents	Clusters/Level			Features				
			l1	l2	l3	Bow	Entity	Skipgram	XLNET	S-BERT
Datasets	*Yelp*	5000	2	4	/	22454	8008	100	120	384
	DBpedia	11 049	3	6	12	67980	24254			
	GitHub-AI-BIO	1528	2	4	8	4994	1643			

Q1- To answer this first research question we compared, over all datasets, the `HTGM` algorithm with three hierarchical tensor-based algorithms, namely `H – PARAFAC` , `H – Tucker`, and `H – TSPLBM`. The results are shown in Figure 2. Over *Yelp* data `HTGM` attains the best performance in both levels 1 and 2, with the biggest improvement compared to other approaches in level 2. We observe that over *GitHub-AI-Bio* data, `HTGM` achieved the best result on level 1 and level 2, but achieved second place in level 3 where the best performance was achieved by the `H – PARAFAC` algorithm. On *DBpedia* data which is the biggest dataset in our experiments, `HTGM` is slightly overwhelmed by `H – PARAFAC` and H-Tucker in level 1 but achieves the best performance in level 2 and level 3. To conclude, `HTGM` achieves overall the best results, particularly at the leaf level.

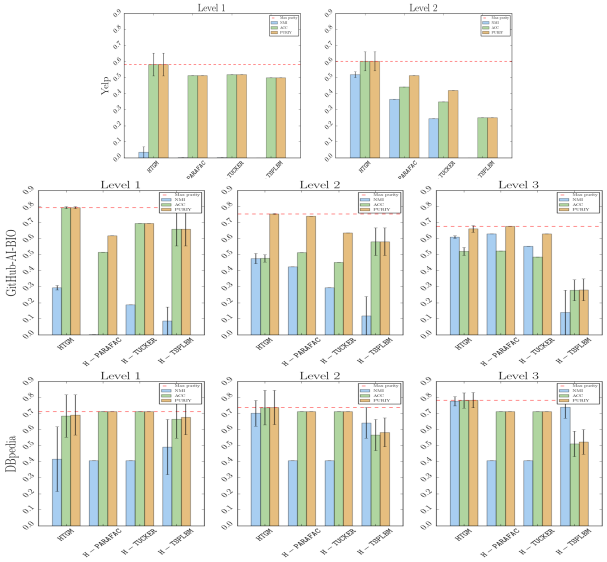

Figure 2: Comparison results of `HTGM` and tensor-based approaches.

Q2-Thereafter, we try to answer the second research question by comparing the results of *a posteriori* graph consensus-based approach using algorithm 3, with the proposed joint clustering approach `HTGM`. Table 2 presents the obtained results at leaf level of hierarchical consensus-based approaches namely `H – consensus – NMF`, `H – consensus – ITCC`, `H – consensus – CoclustMod`, `consensus – Louvain`,

[1] https://github.com/827916600/ClusterEnsembles
[2] https://maartengr.github.io/BERTopic/getting_started/hierarchicaltopics/hierarchicaltopics.html
[3] https://www.kaggle.com/code/danofer/dbpedia-hierarchical-text-classification-dl
[4] https://github.com/yumeng5/WeSHClass/tree/master/yelp
[5] https://github.com/yuzhimanhua/HiGitClass

H – consensus – SPLBM, Hierarchical – BERTopic, and the proposed HTGM. We observed that HTGM achieves consistently the best performance on NMI, purity, and accuracy (ACC). We notice, that H – consensus – ITCC achieves better results than Hierarchical – BERTopic, which leads us to deduce that combining multiple text representations improves clustering results compared to fine-tuned transformer-model for clustering task [8]. These results suggest that the contribution of multi-graph clustering working as an implicit consensus is better than the explicit consensus applied as *a posteriori* step in the clustering schema. Nonetheless, HTGM achieves the best performance in particular for retrieving the second level of hierarchy in the two synthetic data examples. Our results match with the results obtained by the TSPLBM in [5].

Table 2: Comparison of clustering results in terms of NMI using consensus. The bold blue values represent the best performances, and the bold ones are the second best performances.

Data	Algorithms	NMI(Leaf level)	ACC(Leaf level)	Purity(Leaf level)
Github-AI-BIO	H – Consensus – NMF	0.23 ± 0.0	**0.37** ± 0.0	**0.38** ± 0.0
	H – Consensus – ITCC	**0.26** ± 0.06	0.34 ± 0.02	0.35 ± 0.02
	H – Consensus – CoclustMod	**0.26** ± 0.01	0.32 ± 0.01	0.35 ± 0.01
	Consensus – Louvain	0.19 ± 0.03	0.3 ± 0.02	0.33 ± 0.02
	H – Consensus – SPLBM	0.25 ± 0.02	0.35 ± 0.01	**0.38** ± 0.01
	Hierarchical – BERTopic	0.25 ± 0.03	0.31 ± 0.01	0.35 ± 0.01
	HTGM	**0.61** ± 0.01	**0.52** ± 0.02	**0.66** ± 0.02
Yelp	H – Consensus – NMF	0.01 ± 0.0	0.3 ± 0.0	0.3 ± 0.0
	H – Consensus – ITCC	**0.04** ± 0.02	**0.33** ± 0.02	**0.33** ± 0.02
	H – Consensus – CoclustMod	0.03 ± 0.02	**0.33** ± 0.02	**0.33** ± 0.02
	Consensus – Louvain	**0.04** ± 0.02	**0.33** ± 0.03	**0.33** ± 0.03
	H – Consensus – SPLBM	**0.04** ± 0.02	**0.33** ± 0.02	**0.33** ± 0.02
	Hierarchical – BERTopic	0.02 ± 0.0	0.3 ± 0.01	0.31 ± 0.01
	HTGM	**0.52** ± 0.02	**0.6** ± 0.06	**0.6** ± 0.06
DBpedia	H – Consensus – NMF	0.7 ± 0.0	**0.48** ± 0.0	**0.48** ± 0.0
	H – Consensus – ITCC	**0.73** ± 0.04	**0.48** ± 0.03	**0.48** ± 0.03
	H – Consensus – CoclustMod	0.51 ± 0.05	0.39 ± 0.03	0.39 ± 0.03
	Consensus – Louvain	0.48 ± 0.06	0.38 ± 0.04	0.38 ± 0.04
	H – Consensus – SPLBM	0.46 ± 0.05	0.37 ± 0.03	0.37 ± 0.03
	Hierarchical – BERTopic	0.44 ± 0.03	0.28 ± 0.03	0.33 ± 0.02
	HTGM	**0.78** ± 0.03	**0.78** ± 0.05	**0.78** ± 0.05

Figure 3 shows the improvement results of the proposed HTGM compared to Hierarchical – Bertopic at leaf level. In the left figure, we compute the gain of performance using the three measures on all datasets. In the right figure, we vary the size of the DBpedia dataset by generating multiple samples of different sizes, and we compute the percentage of improvement considering the NMI. Thus, the positive values give an advantage to HTGM and show that it outperforms the fine-tuned Hierarchical – Bertopic for clustering. Also, even if the dataset size increases, the HTGM seems to be robust and stable regarding the data scalability.

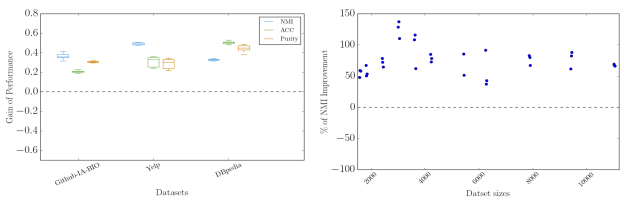

Figure 3: Left: Gain of performance between HTGM and Hierarchical – Bertopic at leaf level for all datasets. Right: Percentage of NMI improvement for HTGM on DBpedia varying the dataset size.

4 CONCLUSION

In this paper, we presented joint clustering techniques working on multi-graphs extending the modularity maximization approach to multi-graph clustering. We validated and evaluated the proposed method HTGM over several synthetic and real-world datasets and observed that the HTGM method consistently achieves better performance than alternative methods. The HTGM method is therefore a promising approach for multi-graph clustering. For future work, we plan to tackle the problem of cluster number selection for each level of hierarchical clustering. Also, we are working on extending the HTGM to other applications such as graphs extracted from the web. Finally, assigning weights for graphs, that express their contribution to maximizing the objective function is a relevant challenge for multi-graph clustering task.

REFERENCES

[1] Melissa Ailem, François Role, and Mohamed Nadif. 2015. Co-clustering document-term matrices by direct maximization of graph modularity. In *CIKM*. 1807–1810.

[2] Melissa Ailem, François Role, and Mohamed Nadif. 2017. Sparse poisson latent block model for document clustering. *IEEE Transactions on Knowledge and Data Engineering* 29, 7 (2017), 1563–1576.

[3] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. *Journal of statistical mechanics* 2008, 10 (2008).

[4] Rafika Boutalbi, Mira Ait-Saada, Anastasiia Iurshina, Steffen Staab, and Mohamed Nadif. 2022. Tensor-based Graph Modularity for Text Data Clustering. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. Association for Computing Machinery, 2227–2231.

[5] Rafika Boutalbi, Lazhar Labiod, and Mohamed Nadif. 2021. Implicit consensus clustering from multiple graphs. *Data Mining and Knowledge Discovery* 35, 6 (2021), 2313–2340.

[6] Inderjit S. Dhillon, Subramanyam Mallela, and Dharmendra S. Modha. 2003. Information-theoretic Co-clustering. In *Proceedings of the Ninth ACM SIGKDD*. 89–98.

[7] Nan Du, Bin Wu, Xin Pei, Bai Wang, and Liutong Xu. 2007. Community Detection in Large-Scale Social Networks. In *Proceedings of the WebKDD Workshop*. ACM, 16–25.

[8] Maarten Grootendorst. 2022. BERTopic: Neural topic modeling with a class-based TF-IDF procedure. *arXiv preprint arXiv:2203.05794* (2022).

[9] Bruce Hendrickson, Robert W Leland, et al. 1995. A Multi-Level Algorithm For Partitioning Graphs. *SC* 95, 28 (1995), 1–14.

[10] Darko Hric, Richard K Darst, and Santo Fortunato. 2014. Community detection in networks: Structural communities versus ground truth. *Physical Review E* 90, 6 (2014).

[11] B. Karrer and M. EJ Newman. 2011. Stochastic blockmodels and community structure in networks. *Physical review E* 83, 1 (2011).

[12] George Karypis and Vipin Kumar. 1998. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on scientific Computing* 20, 1 (1998), 359–392.

[13] Andrew Knyazev. 2018. On spectral partitioning of signed graphs. In *Proceedings of the Seventh SIAM Workshop on Combinatorial Scientific Computing*. 11–22.

[14] Michael W Mahoney, Lorenzo Orecchia, and Nisheeth K Vishnoi. 2012. A local spectral method for graphs: With applications to improving graph partitions and exploring data graphs locally. *The Journal of Machine Learning Research* 13, 1 (2012), 2339–2365.

[15] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *EMNLP-IJCNLP*.

[16] Jianhua Ruan. 2009. A Fully Automated Method for Discovering Community Structures in High Dimensional Data. In *Ninth IEEE International Conference on Data Mining*. 968–973.

[17] Suvrit Sra and Inderjit Dhillon. 2005. Generalized nonnegative matrix approximations with Bregman divergences. *Advances in neural information processing systems* 18 (2005).

[18] Alexander Strehl and Joydeep Ghosh. 2002. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *Journal of machine learning research* 3 (2002), 583–617.

[19] Wei Tang, Zhengdong Lu, and Inderjit S Dhillon. 2009. Clustering with multiple graphs. In *Ninth IEEE International Conference on Data Mining*. IEEE, 1016–1021.

[20] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. *XLNet: generalized autoregressive pretraining for language understanding*. Curran Associates Inc., Chapter 1.