Tensor-based Graph Modularity for Text Data Clustering

Rafika Boutalbi
IPVS - University of Stuttgart, Germany
rafika.boutalbi@ipvs.uni-stuttgart.de

Mira Ait-Saada
Centre Borelli - Université Paris Cité, France
mira.ait-saada@u-paris.fr

Anastasiia Iurshina
IPVS - University of Stuttgart, Germany
anastasiia.iurshina@ipvs.uni-stuttgart.de

Steffen Staab
IPVS - University of Stuttgart, Germany
steffen.staab@ipvs.uni-stuttgart.de

Mohamed Nadif
Centre Borelli - Université Paris Cité, France
mohamed.nadif@u-paris.fr

ABSTRACT

Graphs are used in several applications to represent similarities between instances. For text data, we can represent texts by different features such as bag-of-words, static embeddings (Word2vec, GloVe, etc.), and contextual embeddings (BERT, RoBERTa, etc.), leading to multiple similarities (or graphs) based on each representation. The proposal posits that incorporating the local invariance within every graph and the consistency across different graphs leads to a consensus clustering that improves the document clustering. This problem is complex and challenged with the sparsity and the noisy data included in each graph. To this end, we rely on the modularity metric, which effectively evaluates graph clustering in such circumstances. Therefore, we present a novel approach for text clustering based on both a sparse tensor representation and graph modularity. This leads to cluster texts (nodes) while capturing information arising from the different graphs. We iteratively maximize a Tensor-based Graph Modularity criterion. Extensive experiments on benchmark text clustering datasets are performed, showing that the proposed algorithm referred to as *Tensor Graph Modularity* –TGM– outperforms other baseline methods in terms of clustering task. The source code is available at https://github.com/TGMclustering/TGMclustering.

CCS CONCEPTS

• **Unsupervised learning**; • **Clustering** → *Tensor data*; • **NLP** → Word embedding; • **Graph theory**;

KEYWORDS

Text clustering, Tensor, Graphs, NLP, Word embedding.

ACM Reference Format:
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 (SIGIR '22), July 11–15, 2022, Madrid, Spain.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3477495.3531834

SIGIR '22, July 11–15, 2022, Madrid, Spain

ACM ISBN 978-1-4503-8732-3/22/07…$15.00
https://doi.org/10.1145/3477495.3531834

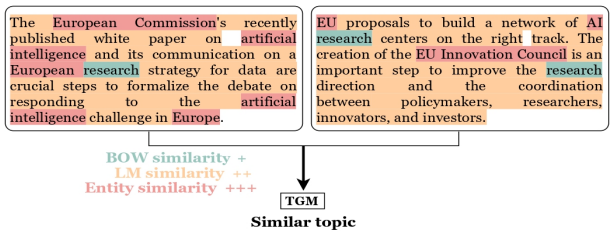

Figure 1: BOW drawbacks for similar text fragments topic. *Green* for word co-occurrence, *Red* for named entities, and *Yellow* for semantic meaning.

1 INTRODUCTION

Text clustering is an unsupervised learning task that aims at grouping a set of texts based on some extracted features (words, entities, embedding, etc.) into classes (or clusters). It relies on explicit or implicit similarity/dissimilarity measures for evaluating the similarity between texts. It is an essential task for many applications such as document retrieval, sentiment analysis, Spam e-mail detection, etc. The Bag-Of-Words (BOW) is a popular model for text representation [18, 27], where the word occurrences describe the text. Then some specific clustering algorithms, such as `Kmeans` [19] or `SphericalKmeans` [10], are applied using the BOW representation to group text into classes [1, 16]. The BOW representation could achieve excellent clustering results, especially when the clusters' topics are quite different. However, BOW does not record the text's sequence information or rich contextual information. In figure 1, the example exposes the limits of the BOW representation. The two text fragments belong to the same topic, but the occurrence of common words based on the BOW representation is very low (only one-word 'research'). To tackle these issues, several works made use of another way of representing text, which is the word embedding representations derived from a Language Model (LM) like Word2vec [20] and GloVe [23]. Unlike BOW, the dense representations provided by word embeddings have a better ability to capture the sense of words and sentences. Some authors use these representations to improve text clustering [13, 14, 29] and they show that the clustering benefits from the two representations, BOW, and word embeddings, and from the mutual information that arises from each representation. More recently, sophisticated language models like the famous Bidirectional Encoder Representations from Transformers (BERT) achieved state-of-the-art results on a wide range of NLP (Natural Language Processing) tasks, including question answering and text generation [11]. In contrast to the previous word embedding techniques, which provide one unique vector for each word of the vocabulary, the word representations delivered

by transformers depend on the context of the given word. They are therefore referred to as contextual word embeddings.

Few works related to Transformer embeddings and entity embeddings are devoted to the purpose of text-clustering [5, 21]. In [31], several text representations (CBOW, BERT, ELMo, etc.) are compared by performing popular clustering algorithms such as `Kmeans`, `SpectralClustering`. The authors showed that the best-performing representation depends on the dataset, and it is impossible to choose the best representation in the unsupervised context. Therefore it is essential to develop new algorithms to leverage several text representations.

A suitable way of combining the different available representations consists in using them as part of a 3-way tensor model representation [25] or 3-way similarity tensor [7]. To achieve the latter for a given dataset, we first compute different representation matrices (using BOW, static and contextual embeddings, and entity linking). Then, we compute from each data representation a similarity matrix viewed as the adjacency matrix of a graph that connects the documents (nodes). Finally, the similarity matrices are structured as a 3-way tensor (cf. figure 2).

This paper proposes a general, practical, and parameter-free method to learn a consensus clustering from multiple similarity graphs via tensor-based graph modularity. The advantage of the proposed approach is taking into account the graphs' properties through the modularity measure. The proposed approach, referred to as *Tensor Graph Modularity* (`TGM`), optimizes a modularity criterion from multiple similarity matrices organized in a three-way data (cf. figure 2). We summarize our main contributions in this work as follows:

- We propose a tensor representation that encloses different text similarities in the same data structure.
- We develop a novel algorithm `TGM` for clustering of multiple graphs based on tensor representation and graph modularity.
- We show that improvements can be achieved by using `TGM` compared to more commonly used clustering and tensor decomposition approaches.

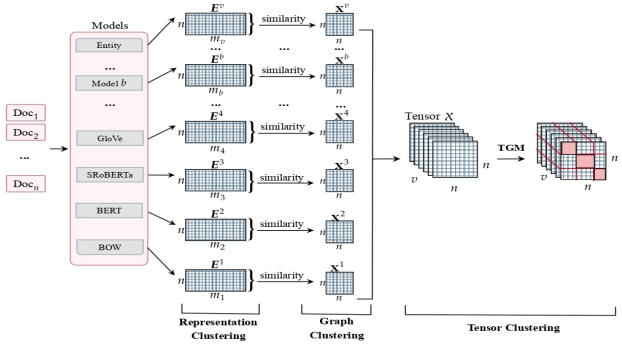

Figure 2: Goal of Tensor-based Graph Modularity (`TGM`) clustering for text data.

2 MODULARITY MEASURE

Modularity is a measure that is commonly used to evaluate the quality of graph clustering. This measure has received considerable attention in several areas since the fundamental work presented in [22] quantifies the edge density of a graph relative to the expected edge density of a random graph [26]. The modularity takes values between -0.5 and 1 [9]. An optimal graph clustering maximizing the modularity measure can be used for document clustering [2].

Given an undirected graph $G = (V, E)$ where V is the set of vertices (or nodes) and E the set of edges between nodes. Let $\mathbf{X} \in \mathbb{R}^{n \times n}$ be a binary-symmetric adjacency matrix with (i, j) as entry. The entry $x_{ij} = 1$ if there is a link between the nodes i and j, and 0 otherwise. The modularity measure $Q(\mathbf{X}, \mathbf{Z})$ for the partition matrix (or one-hot encoding of the label vector) $\mathbf{Z} \in \mathbb{R}^{n \times g}$ of the graph G represented by the adjacency matrix $\mathbf{X}$ is defined as follows:

$$Q(\mathbf{X}, \mathbf{Z}) = \frac{1}{2|E|} \sum_{i,j=1}^{n} (x_{ij} - \frac{x_{i.} x_{.j}}{2|E|}) \sum_{k=1}^{g} z_{ik} z_{jk} \tag{1}$$

where g is the number of clusters, n the number of nodes, $2|E| = \sum_{ij} x_{ij} = x_{..}$ is the number of edges, $x_{i.} = \sum_j x_{ij}$, and $x_{.j} = \sum_i x_{ij}$ the degree of the node i. The expected probability of edges $m_{ij} = \frac{x_{i.} x_{.j}}{2|E|}$ can be represented by the matrix $\mathbf{M}$. We can rewrite the modularity expression as: $Q(\mathbf{X}, \mathbf{Z}) = \frac{1}{x_{..}} \sum_{i,j=1}^{n} \sum_{k=1}^{g} (x_{ij} - m_{ij}) z_{ik} z_{jk}$.

3 TENSOR-BASED GRAPH MODULARITY

We represent the different similarity matrices using a three-way tensor. A three-way tensor or third-order tensor has three dimensions and is accessed by way of three indices. Notice that scalars are represented by lowercase letters e.g., x, and vectors are expressed by a bold lowercase letter e.g., $\mathbf{x}$. The matrices are denoted by bold capital letters e.g., $\mathbf{X}$. And finally, tensors are indicated by bold capital Euler letters e.g. $\mathcal{X}$. The element (i, j) of a matrix is expressed by x_{ij}, and x^b_{ij} represents the element (i, j, b) of a tensor.

Given a three-way tensor $\mathcal{X}$, each slice b represents a graph G_b via an adjacency matrix $\mathbf{X}^b$. The value $x^b_{i.} = \sum_j x^b_{ij}$ represents the degree of node i with regard to the graph G_b, and $x^b_{..} = \sum_{ij} x^b_{ij}$. Hereafter, we propose to tackle the tensor clustering problem by maximizing a modularity-based criterion. More specifically, we aim at maximizing the following criterion:

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

which be written as

$$Q(\mathcal{X}, \mathbf{Z}) = \sum_{b=1}^{v} \frac{1}{x^b_{..}} \sum_{i=1}^{n} \sum_{k=1}^{g} z_{ik} \sum_{j=1}^{n} (x^b_{ij} - \frac{x^b_{i.} x^b_{.j}}{x^b_{..}}) z_{jk} \tag{3}$$

Taking $x^b_{k.} = \sum_i z_{ik} x^b_{i.} = \sum_j z_{jk} x^b_{.j} = x^b_{.k}$ and $x^b_{ik} = \sum_j x^b_{ij} z_{jk}$ leads to

$$Q(\mathcal{X}, \mathbf{Z}) = \sum_{b=1}^{v} \frac{1}{x^b_{..}} \mathrm{Trace}[(\mathbf{X}^b \mathbf{Z} - \mathbf{M}^b \mathbf{Z}) \mathbf{Z}^\top] \tag{4}$$

where $\mathbf{M}^b = (m^b_{ij}) = \frac{x^b_{i.} x^b_{.j}}{2|E^b|}$. Using (3), the goal is to maximize the modularity through all graphs. The optimal hard clustering partition is given as follows. At iteration $(t + 1)$ we have

$$z^{(t+1)}_{ik} = \begin{cases} 1 & \text{if } k = \underset{1 \le k \le g}{\arg\max} \sum_{b=1}^{v} \frac{1}{x^b_{..}} \sum_{i=1}^{n} \sum_{k=1}^{g} (x^b_{ik} - \frac{x^b_{i.} x^b_{.k}}{x^b_{..}}) z^{(t)}_{ik} \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

The proposed approach deals to propose the Tensor Graph Modularity algorithm TGM presented in Algorithm 1. The convergence is achieved when the difference between iteration (t) and iteration ($t+1$) is smaller than ϵ.

Algorithm 1: TGM

Input: $\mathcal{X}$, g.
Initialization: $\mathbf{Z}^{(0)}$ randomly at $t = 0$
repeat
  (1) Compute $Q(\mathcal{X}, \mathbf{Z}^{(t)})$
  (2) Compute $\mathbf{Z}^{(t+1)}$ maximizing $Q(\mathcal{X}, \mathbf{Z}^{(t)})$ using (5)
  (3) Compute $Q(\mathcal{X}, \mathbf{Z}^{(t+1)})$
until *Convergence* $Q(\mathcal{X}, \mathbf{Z}^{(t+1)}) - Q(\mathcal{X}, \mathbf{Z}^{(t)}) < \epsilon$;
return Z, $Q(\mathcal{X}, \mathbf{Z})$

4 EXPERIMENTS

In our experiments, we used benchmark datasets for text clustering with the available true partitions. We evaluate all algorithms using metrics that compare the obtained clustering partitions and the true partitions. Three datasets are used to assess the performance of the different clustering techniques. DBLP1 is proposed in [6], classic3 by Cornell University and we also used an extract of size 8,000 of the AG-news[1] dataset. The number of samples, clusters, and features' dimensions are given for each dataset in Table 1.

Table 1: Description of datasets.

		Documents	Clusters	Features				
				BOW	Entity	GloVe	BERT	SRoBERTa
Datasets	DBLP1	1,949	3	1,585	1,210	300	1,024	768
	classic3	3,891	3	8,555	5,341			
	AG-news	8,000	4	7,873	5,538			

4.1 Experimental settings

In order to evaluate the partitions provided by each clustering, we rely on two well known metrics dedicated to clustering evaluation: Normalized Mutual Information (NMI) [28] All the clustering algorithms are run 30 times with different initializations, and their average scores are compared. In our experiments, we use three representations for each dataset, depending on the algorithm intended to be run. Here, we detail the three data representations that we exploit:

Feature-based representations. In order to use and evaluate standard clustering algorithms on the three datasets, we represent each document by a vector of fixed size, determined by the method that is used to compute the document representations. Given a dataset of n documents, we compute $v = 5$ different data matrices $\mathbf{E}^b, b = 1, \dots v$, each one of size $n \times m_b$, where m_b is the number of features of the representation and is given in Table 1. The feature-based representations are:

The obtained representations are represented by $\mathbf{E}^b, b = 1, \dots, v$, as shown in Figure 2, and are given as input for two standard clustering algorithms, namely Kmeans or SKmeans. In addition, we consider an AE learning a low-dimension representation of size 15 on which a Kmeans is applied.

[1] http://groups.di.unipi.it/~gulli/AG_corpus_of_news_articles.html

Pair-wise similarity representations. Another way of representing a dataset of n samples consists in assigning a pairwise similarity measure x^b_{ij} to each pair (i, j) of document samples as part of the $\mathbf{X}^b$ similarity matrix of size $n \times n$, as shown in figure 2. To compute the matrix $\mathbf{X}^b$, we use the corresponding representation $\mathbf{E}^b$ of the documents, that is described in the previous section. $\mathbf{e}^b_i$ and $\mathbf{e}^b_j$ are the ith and jth row of $\mathbf{E}^b$ respectively. In the case of the BOW and Entity representations, the similarity is computed by the dot product $\mathbf{e}^b_i \cdot \mathbf{e}^b_j$ which represents the number of words or entities in common in documents i and j. For the representations obtained by embedding models (GloVe, BERT, SRoBERTa), we use a binarized cosine similarity measure, computed as: $x^b_{ij} = 1$ if $\frac{\mathbf{e}^b_i \cdot \mathbf{e}^b_j}{\|\mathbf{e}^b_i\| \|\mathbf{e}^b_j\|} \geq p$ and 0 otherwise, where p is the percentile that depends on the sparsity we expect the similarity matrix to have. For the dense embedding representations, we use the percentile that leads to a sparsity of 97%. The obtained representations $\mathbf{X}^b, b = 1, \dots, v$ of size $n \times n$ are used as input for three graph clustering algorithms ITCC [12, 15], CoclustMod[2] [2, 24] and SPLBM [3, 4].

Tensor representations. The information provided by each of the v representations of documents is intrinsically different, we posit that each brings some information that are not necessarily present in the others (cf. the example in Figure 1). In order to harness all the useful information brought by each representation, we simply reorganize the similarity matrices $\mathbf{X}^b, b = 1, \dots, v$ described in the previous section as part of a tensor $\mathcal{X}$ of size $n \times n \times v$. This tensor is used as input to tensor-based clustering techniques which are PARAFAC [17], TUCKER [30], TSPLBM[3] [6–8] and our proposed algorithm TGM.

4.2 Clustering results

Figure 3 reports the performance obtained by 12 combinations of slices where $v \in \{2, 3, 4, 5\}$ using TGM. We first observe that the configuration BOW-SRoBERTa is the best performing 2-slices configuration, achieving competitive NMI scores. Then, adding the BERT and Entity slices improves the performance for all datasets. Finally, adding the GloVe slice lead us to $v = 5$ and further improves the results for 4 datasets out of 6. This shows that TGM is capable to harness the useful knowledge provided by the different representations; it overcomes the possible noise induced by some representations.

The numerical results (NMI) [4] obtained by each clustering algorithm on the three datasets are depicted in Table 2, where TGM_{ALL} stands for the TGM algorithm applied to a tensor containing all of the five slices (BOW, BERT, SRoBERTa, GloVe and Entity). We first observe that the best scores are achieved by TGM in the majority of cases. The scores obtained by TGM are still competitive. Also, the (max-min) values on unique representation indicate that none of the five ways of representing the documents are robust enough when used individually as input to any of the clustering or graph clustering algorithms, in comparison to TGM. This also supports the idea that each of the five representations brings some valuable

[2] https://coclust.readthedocs.io/en/v0.2.1/
[3] https://tensorclus.readthedocs.io/en/latest/
[4] We provide results of TGM using other clustering metrics such as *accuracy*, *purity*, and *ARI*. The GPU version of TGM for larger datasets is also provided.

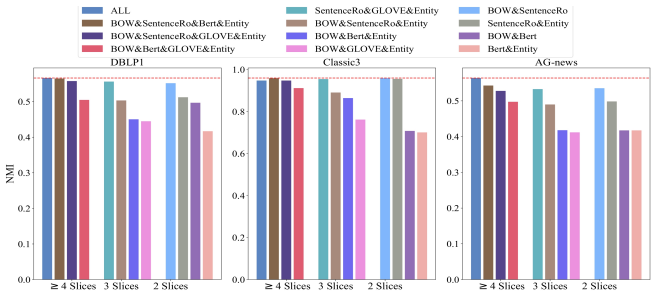

Figure 3: Comparison of TGM results for different combinations of similarity slices.

information, potentially not available in the others. Also, it is important to note that, being in an the unsupervised setting where labels are not supposed to be available, it is impossible to know in advance which slice is more likely to perform best in practice. Furthermore, among the tensor-based clustering algorithms, TGM is the one that seems to combine the different representations in the most effective way, directly followed by TSPLBM [6].

Table 2: Evaluation of documents clustering in terms of NMI. The bold blue values represent the best performances and the bold ones the second best performances.

Data	Evaluation	Metrics / Algorithms	NMI BOW	NMI BERT	NMI SRoBERTa	NMI GloVe	NMI Entity	Max-Min
DBLP1	Clustering	Kmeans	0.20 ± 0.00	0.28 ± 0.00	0.25 ± 0.00	0.08 ± 0.00	0.01 ± 0.00	0.27
		SKmeans	0.25 ± 0.04	0.18 ± 0.17	0.37 ± 0.06	0.28 ± 0.03	0.13 ± 0.04	0.24
		AE	0.16 ± 0.07	0.16 ± 0.05	0.15 ± 0.02	0.32 ± 0.03	0.09 ± 0.05	0.23
	Graph (Similarity)	ITCC	0.33 ± 0.08	0.35 ± 0.07	0.34 ± 0.05	0.10 ± 0.02	0.20 ± 0.03	0.25
		CoclustMod	0.32 ± 0.06	0.36 ± 0.11	0.35 ± 0.06	0.13 ± 0.03	0.18 ± 0.04	0.23
		SPLBM	0.32 ± 0.09	0.32 ± 0.11	0.33 ± 0.06	0.06 ± 0.03	0.20 ± 0.05	0.13
	Tensor	PARAFAC	0.09 ± 0.05					
		TUCKER	0.08 ± 0.00					
		TSPLBM	**0.50 ± 0.10**					
		TGM_{ALL}	**0.53 ± 0.09**					
Classic3	Clustering	Kmeans	0.63 ± 0.00	0.85 ± 0.00	0.33 ± 0.00	0.88 ± 0.00	0.62 ± 0.00	0.55
		SKmeans	0.84 ± 0.12	0.84 ± 0.08	0.73 ± 0.12	0.86 ± 0.08	0.82 ± 0.00	0.13
		AE	0.24 ± 0.09	0.87 ± 0.01	0.20 ± 0.03	0.88 ± 0.02	0.66 ± 0.06	0.68
	Graph (Similarity)	ITCC	0.88 ± 0.00	0.45 ± 0.10	0.59 ± 0.12	0.45 ± 0.08	0.81 ± 0.00	0.43
		CoclustMod	0.84 ± 0.09	0.38 ± 0.09	0.47 ± 0.06	0.46 ± 0.03	0.81 ± 0.00	0.46
		SPLBM	0.36 ± 0.30	0.53 ± 0.08	0.65 ± 0.10	0.43 ± 0.05	0.38 ± 0.25	0.29
	Tensor	PARAFAC	0.30 ± 0.17					
		TUCKER	0.44 ± 0.00					
		TSPLBM	**0.89 ± 0.07**					
		TGM_{ALL}	**0.91 ± 0.10**					
AG-news	Clustering	K − means	0.04 ± 0.00	0.45 ± 0.00	0.29 ± 0.00	0.53 ± 0.00	0.02 ± 0.00	0.51
		SKmeans	0.12 ± 0.03	0.46 ± 0.11	0.51 ± 0.01	0.53 ± 0.00	0.07 ± 0.02	0.46
		AE	0.30 ± 0.04	0.5 ± 0.03	0.29 ± 0.03	**0.54 ± 0.01**	0.20 ± 0.04	0.34
	Graph (Similarity)	ITCC	0.13 ± 0.03	0.37 ± 0.09	0.52 ± 0.06	0.39 ± 0.04	0.07 ± 0.02	0.32
		CoclustMod	0.11 ± 0.03	0.33 ± 0.10	0.48 ± 0.07	0.43 ± 0.05	0.07 ± 0.02	0.41
		SPLBM	0.06 ± 0.02	0.34 ± 0.10	0.44 ± 0.06	0.33 ± 0.06	0.08 ± 0.03	0.38
	Tensor	PARAFAC	0.07 ± 0.03					
		TUCKER	0.05 ± 0.00					
		TSPLBM	0.51 ± 0.06					
		TGM_{ALL}	**0.55 ± 0.05**					

[1] AE, PARAFAC and TUCKER followed by Kmeans.

Figure 4 shows the evolution of the TGM's final objective function Q (normalized to a $[0, 1]$ interval) along with the NMI external measure of the partitions. First, we observe that the worst values of Q coincide with the lowest NMI scores. Then, we can notice an increasing tendency of the NMI that approximately fits the evolution of Q for most of the datasets, which is confirmed by a high correlation coefficient. This indicates that, in the absence of labeled supervision, the modularity optimized by TGM is an effective unsupervised indicator that can be used to select a run that is likely to have a good performance.

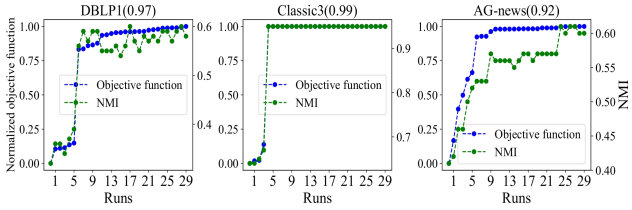

Figure 4: Evolution of the objective function of TGM_{ALL} and NMIs for the 10 top runs. (Correlation) between the objective function and the NMI values.

In order to further improve model selection, we perform a consensus clustering approach [28]. For this purpose, we use the Cluster_Ensembles[5] implementation that is based on graph partitioning and aims to find a consensual partition from several input clusterings. In Table 3, we compare between the average NMI w/o (without) consensus (corresponding to the TGM_{ALL} values presented in Table 2) and the NMI obtained by the consensual partition provided by the ensemble (combining the 30 runs into a single partition). We can see that for all datasets there is an improvement in terms of NMI, particularly significant for DBLP1 and AG-news.

Table 3: Evaluation of TGM_{ALL}; ↑ % denotes the percentage of improvement in terms of NMI.

Datasets	without Consensus	with Consensus	↑ %
DBLP1	0.53	0.60	13.21
Classic3	0.91	0.95	4.40
AG−news	0.55	0.60	9.09

5 ASSESSING THE NUMBER OF CLUSTERS

Assessing the number of clusters still remains a challenge whatever the chosen approach. In our proposal, to deal with this problem we conducted a series of experiments with the number of clusters varying from 2 to 10. In Figure 5, we show the relationship between the number of clusters and the corresponding value of the modularity measure for the 3 datasets. The red lines indicate the real number of clusters for each data set. We can see that even though the maximum value of modularity does not always correspond to the real number of clusters, in many cases it is close the true value, especially if we look at the largest difference between values of modularity (e.g., DBLP1, Classic3). Therefore the moduarity can

[5] https://pypi.org/project/Cluster_Ensembles/

serve as an additional point of information when choosing the number of clusters.

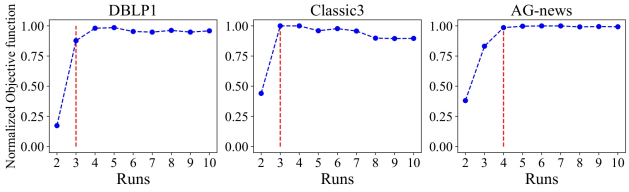

Figure 5: Estimation of the number of clusters according the objective function.

6 CONCLUSION

It is now well established that different representations are required to capture the syntactic and semantic meaning of several NLP tasks, including text clustering. In this paper we proposed the *Tensor Graph Modularity* –TGM– algorithm, which harnesses this idea by iteratively maximizing a Graph Modularity criterion. TGM relies on a tensor representation that allows us to combine several representations and take advantage of the different similarity aspects, namely syntactic and semantic. We evaluated our method through in-depth experiments with three document datasets and compare the results with three categories of clustering baselines: classical clustering, graph clustering, and tensor clustering. Then We showed how improve performance of TGM by providing a consensual partition from several input clusterings. Finally, we addressed the issue of the number of clusters which is often unknown and showed that the modularity can be harnesses for this task.

A potential future work might be to introduce a schema for different representations. This would allow to evaluate the impact of each type of representation on the clustering performance.

7 ACKNOWLEDGEMENT

Our work is funded by the German Federal Ministry for Economic Affairs and Climate Action under Grant Agreement Number 01MK20008F (Service-Meister).

REFERENCES

[1] Aggarwal, C. C., and Zhai, C. A survey of text clustering algorithms. In *Mining text data*. Springer, 2012, pp. 77–128.
[2] Ailem, M., Role, F., and Nadif, M. Co-clustering document-term matrices by direct maximization of graph modularity. In *CIKM* (2015), pp. 1807–1810.
[3] Ailem, M., Role, F., and Nadif, M. Model-based co-clustering for the effective handling of sparse data. *Pattern Recognition 72* (2017), 108–122.
[4] Ailem, M., Role, F., and Nadif, M. Sparse poisson latent block model for document clustering. *IEEE Transactions on Knowledge and Data Engineering 29*, 7 (2017), 1563–1576.
[5] Ait-Saada, M., Role, F., and Nadif, M. *How to Leverage a Multi-Layered Transformer Language Model for Text Clustering: An Ensemble Approach*. Association for Computing Machinery, 2021, p. 2837–2841.
[6] Boutalbi, R., Labiod, L., and Nadif, M. Sparse tensor co-clustering as a tool for document categorization. In *ACM SIGIR* (2019), pp. 1157–1160.
[7] Boutalbi, R., Labiod, L., and Nadif, M. Implicit consensus clustering from multiple graphs. *Data Mining and Knowledge Discovery 35*, 6 (2021), 2313–2340.
[8] Boutalbi, R., Labiod, L., and Nadif, M. Tensorclus: A python library for tensor (co)-clustering. *Neurocomputing 468* (2022), 464–468.
[9] Brandes, U., Delling, D., Gaertler, M., Gorke, R., Hoefer, M., Nikoloski, Z., and Wagner, D. On modularity clustering. *IEEE Transactions on Knowledge and Data Engineering 20*, 2 (2008), 172–188.
[10] Buchta, C., Kober, M., Feinerer, I., and Hornik, K. Spherical k-means clustering. *Journal of statistical software 50*, 10 (2012), 1–22.
[11] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In *ACL* (2019), pp. 4171–4186.
[12] Dhillon, I. S., Mallela, S., and Modha, D. S. Information-theoretic co-clustering. In *Proceedings of the Ninth ACM SIGKDD* (2003), pp. 89–98.
[13] Ganguly, D., and Ghosh, K. Contextual word embedding: a case study in clustering tweets about emergency situations. In *Companion Proceedings of the The Web Conference 2018* (2018), pp. 73–74.
[14] Ge, L., and Moh, T.-S. Improving text classification with word embedding. In *2017 IEEE International Conference on Big Data (Big Data)* (2017), pp. 1796–1805.
[15] Govaert, G., and Nadif, M. Mutual information, phi-squared and model-based co-clustering for contingency tables. *Advances in Data Analysis and Classification 12*, 3 (Sep 2018), 455–488.
[16] Guan, R., Zhang, H., Liang, Y., Giunchiglia, F., Huang, L., and Feng, X. Deep feature-based text clustering and its explanation. *IEEE Transactions on Knowledge and Data Engineering* (2020).
[17] Harshman, R. A., and Lundy, M. E. Parafac : parallel factor analysis. *Computational statistics and data analysis 18* (1994), 39–72.
[18] Khan, K., Baharudin, B. B., Khan, A., et al. Mining opinion from text documents: A survey. In *2009 3rd IEEE International Conference on Digital Ecosystems and Technologies* (2009), IEEE, pp. 217–222.
[19] MacQueen, J., et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability* (1967), vol. 1, Oakland, CA, USA, pp. 281–297.
[20] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems* (2013), pp. 3111–3119.
[21] Nadif, M., and Role, F. Unsupervised and self-supervised deep learning approaches for biomedical text mining. *Briefings in Bioinformatics 22*, 2 (2021), 1592–1603.
[22] Newman, M. E., and Girvan, M. Finding and evaluating community structure in networks. *Physical review E 69*, 2 (2004), 026113.
[23] Pennington, J., Socher, R., and Manning, C. GloVe: Global vectors for word representation. In *EMNLP* (2014), pp. 1532–1543.
[24] Role, F., Morbieu, S., and Nadif, M. Coclust: A python package for co-clustering. *Journal of Statistical Software 88*, 7 (2019), 1–29.
[25] Romeo, S., Tagarelli, A., and Ienco, D. Semantic-based multilingual document clustering via tensor modeling. In *EMNLP: Empirical Methods in Natural Language Processing* (2014), pp. 600–609.
[26] Shao, Y., and Zavala, V. M. Modularity measures: Concepts, computation, and applications to manufacturing systems. *AIChE Journal 66*, 6 (2020), e16965.
[27] Sharma, S., and Gupta, V. Recent developments in text clustering techniques. *International Journal of Computer Applications 37*, 6 (2012), 14–19.
[28] Strehl, A., and Ghosh, J. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *Journal of machine learning research 3* (2002), 583–617.
[29] Tache, A. M., Gaman, M., and Ionescu, R. T. Clustering word embeddings with self-organizing maps. application on laroseda–a large romanian sentiment data set. *arXiv preprint arXiv:2101.04197* (2021).
[30] Tucker, L. R. Some mathematical notes on three-mode factor analysis. *Psychometrika 31*, 3 (1966), 279–311.
[31] Walkowiak, T., and Gniewkowski, M. Evaluation of vector embedding models in clustering of text documents. In *RANLP* (2019), pp. 1304–1311.