Graph-Convolutional Networks: Named Entity Recognition and Large Language Model Embedding in Document Clustering

Imed Keraghel, Mohamed Nadif

Graph-Convolutional Networks: Named Entity Recognition and Large Language Model Embedding in Document Clustering

Imed Keraghel[1,2][0009−0008−4943−2556] and Mohamed Nadif[1][0000−0002−0007−3950]

[1] Centre Borelli UMR9010, Université Paris Cité, Paris, France
[2] Kernix Software, Paris, France.
https://www.kernix.com/

Abstract. Recent advances in machine learning, particularly Large Language Models (LLMs) such as BERT and GPT, provide rich contextual embeddings that improve text representation. However, current document clustering approaches often ignore the deeper relationships between named entities (NEs) and the potential of LLM embeddings. This paper proposes a novel approach that integrates Named Entity Recognition (NER) and LLM embeddings within a graph-based framework for document clustering. The method builds a graph with nodes representing documents and edges weighted by named entity similarity, optimized using a graph-convolutional network (GCN). This ensures a more effective grouping of semantically related documents. Experimental results indicate that our approach outperforms conventional co-occurrence-based methods in clustering, notably for documents rich in named entities.

Keywords: Large Language models, Named Entity Recognition, Graph Convolutional Networks, Node Embedding, Node Clustering.

1 Introduction

Document clustering is widely used in data analytics, especially in fields like information retrieval and natural language processing (NLP). It groups documents into categories based on shared characteristics, which is crucial for tasks like topic modeling and recommendation systems. Traditional methods depend on lexical features [29, 39], co-occurrences, or TF-IDF matrices [1, 27]. Although effective in various situations, these methods face challenges in recognizing semantic connections that extend beyond basic word frequency analysis [13].

Recent advances in machine learning, particularly with the emergence of Large Language Models (LLMs) such as BERT [35, 18] and GPT [24], have revolutionized text representation. These models offer rich contextual embeddings that capture the nuances of word meanings in context. However, many clustering methods still use traditional approaches like k-Nearest Neighbors (KNN) to construct graphs [14, 23], which depend on shallow lexical similarities and often fail to capture deeper semantic relationships between documents.

To illustrate the limitations of traditional methods, Figure 4 compares two graph structures generated from the BBC News dataset: one using a KNN-based graph and the other using our proposed NER-based graph construction method. In the KNN graph, documents are connected based on lexical similarity, leading to overlapping and indistinct clusters. In contrast, the NER-based graph leverages named entity similarities, resulting in a clearer separation of clusters that correspond to distinct semantic topics.

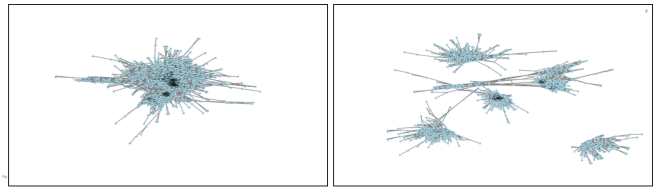

Fig. 1. Comparison of graph structures for the BBC News dataset. (Left) KNN-based graph with lexical similarity; (Right) NER-based graph capturing entity similarities.

In this paper, we propose a novel approach that leverages Named Entity Recognition (NER) alongside LLM embeddings in a graph-based framework for document clustering. Our method builds a graph where nodes represent documents and edges are weighted by the similarity of named entity contexts in each document. Named entities in similar contexts show strong semantic connections, and combining NER with LLM embeddings constructs a graph that better represents document similarities. To optimize this, we use a Graph Convolutional Network (GCN) [11, 15], which enables the joint optimization of the embeddings and the clustering objectives. This procedure guarantees that semantically related documents are grouped more effectively, addressing the shortcomings of conventional clustering techniques that do not incorporate global contextual information. Our main contributions include:

- A document clustering method that improves accuracy using NER, LLM embeddings, and graph-based representations.
- A technique to create an adjacency matrix based on named entity similarity for precise document relationship identification.
- Experiments demonstrating our approach outperforms co-occurrence-based techniques, especially for documents rich in named entities.

The remainder of this paper is organized as follows. Section 2 reviews the literature on graph representation learning, graph clustering, GCNs, LLMs, and NER. Section 3 details our method, including adjacency matrix construction and GCNs. Section 4 presents the experimental framework and results, followed by a conclusion and future perspectives in Section 5.

2 Related work

Unsupervised graph representation learning has seen remarkable progress in recent years, primarily through two key approaches: contrastive learning and autoencoders. Contrastive learning has emerged as a powerful method due to its ability to differentiate between positive and negative pairs in a self-supervised manner. Techniques such as GraphCL [42] have introduced graph augmentations that improve graph representations by maximizing the agreement between different augmented views of the same graph. Similarly, MVGRL [12] enhances the embeddings of node- and graph-level by contrasting views at both levels. On the other hand, autoencoder-based methods provide an alternative by reconstructing specific graph properties, such as the adjacency matrix, to learn embeddings. For example, Graph Autoencoders (GAEs), introduced by [37], propose an architecture designed for large-scale graph data, enabling tasks such as node classification and link prediction. Further developments, such as Variational Graph Autoencoders (VGAE) [16], extend this approach by incorporating variational autoencoders with GCNs, while Linear Variational Graph Autoencoders (LVGAE) [28] simplify this architecture using a linear transformation with a one-hop propagation matrix.

Building on the foundation of graph representation learning, graph clustering aims to group nodes into clusters based on either graph structure or node-level features, or both. When focusing solely on node attributes, traditional clustering algorithms can be applied to graph data. Spectral clustering [22], for example, leverages the graph structure exclusively and optimizes the ratio and normalized cut criteria to form clusters. Deep Graph Infomax (DGI) [36], on the other hand, improves graph clustering by maximizing mutual information between graph embeddings and substructures. In addition to these approaches, more recent methods have explored joint optimization of node embeddings and clustering objectives. An example is Graph Convolutional Clustering (GCC) [11], which simultaneously learns node embeddings and clusters, resulting in enhanced performance and reduced computational costs. These methods show that incorporating clustering into the learning process can result in more meaningful representations and improved clustering outcomes.

In parallel to advancements in graph learning, recent progress in LLMs has significantly improved document representation. Models such as BERT or GPT provide deep contextual embeddings that are particularly well suited for tasks such as clustering, as they capture semantic relationships between entities and documents that traditional methods often miss [21, 13]. Sentence-BERT [25], for example, fine-tunes BERT to specialize in sentence similarity, making it an ideal candidate for clustering semantically related texts. Incorporating clustering algorithms such as k-means directly into LLM embeddings has also been explored, as seen with BERT-Kmeans [31, 13]. Furthermore, the combination of LLMs with graph-based models, such as Graph-BERT [43], provides a promising avenue for more accurate and context-aware document clustering by integrating semantic knowledge and graph structures.

3 Model and algorithm

The use of NEs in clustering, although not yet widely adopted, presents a promising approach to improve document clustering. By focusing on key entities, such as medical concepts, NE-based clustering can group documents that share relevant entities. For example, in sports news, named entities such as "Kylian Mbappé" and "Cristiano Ronaldo" can link documents about their performances, such as a match in which Mbappé leads PSG to victory or Ronaldo secures Portugal's World Cup semi-final spot. Furthermore, their involvement in political campaigns, like a UN-backed global education initiative alongside "Emmanuel Macron," can connect documents from political domain. Research such as [6, 8] shows that the integration of NEs into clustering can lead to improved performance, especially in fields where entities are central to the thematic structure of the text. We start by presenting the preliminaries and notation, followed by a description of our proposed model and algorithm (Figure 2).

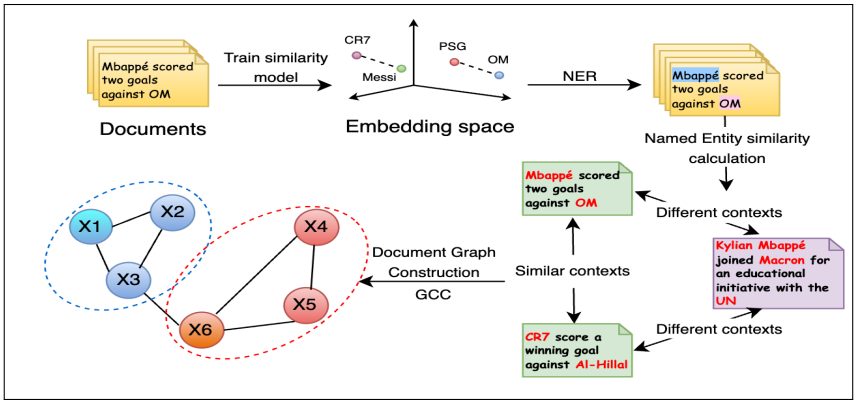

Fig. 2. Overview of the proposed model pipeline: LLM-based feature extraction, NER-based graph construction, and joint embedding and clustering.

3.1 Preliminaries and Notation

Let $\mathcal{G} = (\mathcal{V}, \mathbf{A}, \mathbf{X})$ represent an undirected graph, where: $\mathcal{V}$ is the set of vertex, consisting of nodes $\{v_1, \ldots, v_n\}$, $\mathbf{A} \in \mathbb{R}^{n \times n}$ is a symmetric adjacency matrix, where a_{ij} indicates the edge weight between nodes v_i and v_j, $\mathbf{X} \in \mathbb{R}^{n \times d}$ is the matrix of node features. In the following, we adopt the following notations: $\mathrm{Tr}(\cdot)$ denotes the trace of a matrix, k is the number of clusters, e is the embedding dimension, $\mathbb{1}_m$ is a column vector of m ones, and $\mathbf{I}_m$ is the identity matrix of dimension m. For matrix $\mathbf{X}$, $\mathbf{x}_i$ refers to the i-th row vector, $\mathbf{x}'_j$ to the j-th column vector, and x_{ij} to the element in the i-th row and j-th column.

3.2 Leveraging LLM Embeddings to Represent Node Features

In our approach, the node feature matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$, which typically represents document features, is now replaced by LLM embeddings and d becomes $d_{\ell\ell m}$. Traditional feature representations such as TF-IDF focus on surface-level word frequencies and co-occurrence patterns, which may not capture semantic nuances. Using LLM embeddings, we aim to incorporate rich contextual information that provides a more holistic view of the document semantics.

Let $f_{\ell\ell m} : \mathcal{D} \to \mathbb{R}^{d_{\ell\ell m}}$ be the LLM model that maps each document $d_i \in \mathcal{D}$ or its components (words, sentences, or entities) to a vector in a high-dimensional embedding space. The features matrix $X_{\ell\ell m}$ is constructed as:

$$\mathbf{X}_{\ell\ell m} = \begin{bmatrix} f_{\ell\ell m}(d_1) \\ f_{\ell\ell m}(d_2) \\ \vdots \\ f_{\ell\ell m}(d_n) \end{bmatrix} = [\mathbf{x}'_1, \ldots, \mathbf{x}'_{d_{\ell\ell m}}] \in \mathbb{R}^{n \times d_{\ell\ell m}} \tag{1}$$

where each row $f_{\ell\ell m}(d_i) = \mathbf{x}_i$ is the embedding of document d_i.

3.3 Leveraging Named Entities for Context-Aware Graph Construction

We aim to train a similarity model, such as Word2Vec [20], to capture word-level similarities. Note that any other model can be used. The objective is to identify named entities, locate entities that occur in similar contexts using the trained model and cosine similarity, and build a document graph $\bar{\mathcal{G}}$ where edges represent similarities between named entities across documents.

Let $\mathcal{D} = \{d_1, d_2, \ldots, d_n\}$ be the set of documents in the dataset. The problem is divided into the following steps:

Step 1: Training a Similarity Model First, we train a Word2Vec model $f : \mathcal{V} \to \mathbb{R}^e$, where $\mathcal{V}$ is the vocabulary of the dataset, and each word $w_i \in \mathcal{V}$ is mapped to an embedding vector $f(w_i) \in \mathbb{R}^e$. The objective of Word2Vec is to maximize the likelihood of predicting the context words for a given target word w_i using the following objective function:

$$\max_{\theta} \sum_{w_i \in \mathcal{V}} \sum_{w_j \in \text{Context}(w_i)} \log P(w_j \mid w_i; \theta) \tag{2}$$

where θ represents the parameters of the model and $P(w_j \mid w_i; \theta)$ is the probability that the word w_j appears in the context of the word w_i.

Step 2: Named Entity Recognition (NER) Let $\mathcal{E}_d = \{e_1, e_2, \ldots, e_{|E_d|}\}$ be the set of named entities extracted from a document $d \in \mathcal{D}$. We apply a NER model to each document d_i to identify named entities $\mathcal{E}_{d_i}$. Formally, we define an NER model as: $\mathcal{NER} : d \to \mathcal{E}_d$ where $\mathcal{E}_d$ is the set of named entities detected in document d.

Step 3: Entity Similarity Search For each named entity $e \in \mathcal{E}_d$, we use the trained model f to compute the cosine similarity between named entities in different documents. Given two entities e_i and e_j, their similarity is defined as:

$$\text{Sim}(e_i, e_j) = \frac{f(e_i) \cdot f(e_j)}{\|f(e_i)\|\|f(e_j)\|}. \tag{3}$$

We identify pairs of named entities (e_i, e_j) in documents that have a cosine similarity exceeding a threshold τ.

Step 4: Graph Construction (with Entity Threshold) We construct a document graph $\bar{\mathcal{G}} = (V_{\bar{\mathcal{G}}}, E_{\bar{\mathcal{G}}})$, where $V_{\bar{\mathcal{G}}}$ represents the set of documents $\{d_1, \ldots, d_n\}$ and $E_{\bar{\mathcal{G}}}$ represents the edges between documents. An edge exists between two documents d_i and d_j if they share a sufficient number of similar named entities. Specifically, an edge is formed only if the number of shared entities $\mathcal{E}_{d_i} \cap \mathcal{E}_{d_j}$ is greater than or equal to a predefined threshold τ. The weight of the edge a_{ij} is proportional to the similarity of the named entities between the two documents:

$$a_{ij} = \frac{1}{|\mathcal{E}_{d_i} \cap \mathcal{E}_{d_j}|} \sum_{e_i \in \mathcal{E}_{d_i}, e_j \in \mathcal{E}_{d_j}} \text{Sim}(e_i, e_j) \tag{4}$$

where $|\mathcal{E}_{d_i} \cap \mathcal{E}_{d_j}| \geq \tau$. This ensures that a link is established only when the documents share a sufficient number of similar entities, reducing the risk of connecting documents based on superficial relationships. The resulting graph is represented by an adjacency matrix $\mathbf{A}_{ner}$, where $\mathbf{A}_{ner}(i, j) = a_{ij}$ for documents d_i and d_j that satisfy the entity threshold τ. The final objective is to harness this graph for document clustering.

3.4 Joint Embedding and Clustering

Our objective is to simultaneously learn node embeddings and cluster assignments. Inspired by [11], we formulate the problem as follows:

$$\begin{aligned} \min_{\theta_1, \theta_2, \mathbf{G}, \mathbf{F}} \Big(& \left\| \mathrm{D}_{\theta_2} \left(\mathrm{E}_{\theta_1} \left(\mathrm{Agg}(\mathbf{A}_{ner}, \mathbf{X}_{\ell\ell m}) \right) \right) - \mathrm{Agg}(\mathbf{A}_{ner}, \mathbf{X}_{\ell\ell m}) \right\|^2 \\ & + \lambda \left\| \mathrm{E}_{\theta_1} \left(\mathrm{Agg}(\mathbf{A}_{ner}, \mathbf{X}_{\ell\ell m}) \right) - \mathbf{G}\mathbf{F} \right\|^2 \Big) \\ & \text{subject to: } \mathbf{G} \in \{0, 1\}^{n \times k}, \quad \mathbf{G}\mathbb{1}_k = \mathbb{1}_n \end{aligned} \tag{5}$$

where E_{θ_1} and D_{θ_2} represent the encoding and decoding functions. The term $\mathrm{Agg}(\mathbf{A}_{ner}, \mathbf{X}_{\ell\ell m})$ is an aggregation of the adjacency matrix $\mathbf{A}_{ner}$ and node features $\mathbf{X}_{\ell\ell m}$, $\mathbf{G} \in \{0, 1\}^{n \times k}$ is the binary cluster assignment matrix, $\mathbf{F} \in \mathbb{R}^{k \times d}$ represent the cluster centroids in the embedding space while the parameter λ controls the trade-off between reconstruction and clustering. Specifically, the second term in the objective function can be viewed as optimized by `Kmeans`. It is applied on the encoded representations forcing, thereby, the learned embeddings to be clustering-friendly. This penalizes representations that do not fit into clear clusters, following the loss of `Kmeans`.

Linear Graph Embedding Model. We use a linear graph autoencoder (LGAE) approach, which has been shown to perform comparably to more complex GCN-based models for tasks such as link prediction and node clustering [16, 28]. Our encoder is defined as a simple linear transformation:

$$\mathrm{E}\left(\mathrm{Agg}(\mathbf{A}_{ner}, \mathbf{X}_{\ell\ell m}); \mathbf{W}_1\right) = \mathrm{Agg}(\mathbf{A}_{ner}, \mathbf{X}_{\ell\ell m})\mathbf{W}_1 \tag{6}$$

In contrast to LGAE, which reconstructs the adjacency matrix $\mathbf{A}_{ner}$ directly, the decoder incorporates both the adjacency matrix and the node features and is $\mathbf{Z}\mathbf{W}_2$ where $\mathbf{Z} = \mathrm{Agg}(\mathbf{A}_{ner}, \mathbf{X}_{\ell\ell m})\mathbf{W}_1$, representing the encoded graph.

Normalized Simple Graph Convolution. Inspired by the Simple Graph Convolution (SGC) [40], we define our aggregation function by $\mathrm{Agg}(\mathbf{A}_{ner}, \mathbf{X}_{\ell\ell m}) = \mathbf{T}^p\mathbf{X}_{\ell\ell m}$ where $\mathbf{T}$ the symmetric normalized adjacency matrix with added self-loops is defined by $\mathbf{T} = \mathbf{D}_T^{-1}(\mathbf{I}+\tilde{\mathbf{S}})$ where $\tilde{\mathbf{S}} = \tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}$ with $\tilde{\mathbf{A}} = \mathbf{A}_{ner}+\mathbf{I}$; $\tilde{\mathbf{D}}$ and $\mathbf{D}_T$ are the diagonal degree matrices of $\tilde{\mathbf{A}}$ and $\mathbf{I} + \tilde{\mathbf{S}}$ respectively. This formulation extends traditional graph convolution by normalizing the spectrum of the graph filter, ensuring that the filter acts as a low-pass filter in the frequency range $[0, 1]$. In the following, for convenience, denote the matrix $\mathbf{T}^p\mathbf{X}_{\ell\ell m}$ by $\mathbf{Y}^p$.

Optimization Problem. The objective function takes the following form:

$$\min_{\mathbf{G},\mathbf{F},\mathbf{W}_1,\mathbf{W}_2} \left\|\mathbf{Y}^p - \mathbf{Y}^p\mathbf{W}_1\mathbf{W}_2\right\|^2 + \lambda\left\|\mathbf{Y}^p\mathbf{W}_1 - \mathbf{G}\mathbf{F}\right\|^2 \tag{7}$$

$$\text{subject to: } \mathbf{G} \in \{0,1\}^{n\times k}, \quad \mathbf{G}\mathbb{1}_k = \mathbb{1}_n$$

The two terms in (7) establish a link between the two tasks, with the first term acting as a linear autoencoder and the second term facilitating clustering in the embedding space. The parameter λ controls the importance of the second term in terms of regularizing the embedding. However, we take $\lambda = 1$ as in [11]; this assumption can be investigated in the future even it appears effective in our experiments.

Graph Convolutional Clustering. To further enhance the interaction between embedding and clustering, we assume $\mathbf{W} = \mathbf{W}_1 = \mathbf{W}_2^\top$ and impose an orthogonality constraint $\mathbf{W}^\top\mathbf{W} = \mathbf{I}_k$, leading to the modified problem:

$$\min_{\mathbf{G},\mathbf{F},\mathbf{W}} \left\|\mathbf{Y}^p - \mathbf{Y}^p\mathbf{W}\mathbf{W}^\top\right\|^2 + \left\|\mathbf{Y}^p\mathbf{W} - \mathbf{G}\mathbf{F}\right\|^2 \tag{8}$$

$$\text{subject to: } \mathbf{G} \in \{0,1\}^{n\times k}, \quad \mathbf{G}\mathbb{1}_k = \mathbb{1}_n, \quad \mathbf{W}^\top\mathbf{W} = \mathbf{I}_k$$

This formulation encourages a more direct interaction between the embedding and clustering tasks. Following [2, 3, 17, 11], we can show that solving this problem is equivalent to solving (subject to the same constraints):

$$\min_{\mathbf{G},\mathbf{F},\mathbf{W}} \left\|\mathbf{Y}^p - \mathbf{G}\mathbf{F}\mathbf{W}^\top\right\|^2. \tag{9}$$

By decomposing the reconstruction and regularization terms, it can be shown that both formulations lead to the same solution, allowing efficient joint learning of embeddings and clusters. The solution of the classical problem (9) is accomplished by alternating updates of $\mathbf{G}$, $\mathbf{F}$ and $\mathbf{W}$; all steps are detailed in [11] where $\mathbf{W}$ is initialized using a randomized PCA on $\mathbf{Y}^p$ and $\mathbf{G}$ is initialized using `Kmeans` on $\mathbf{Y}^p\mathbf{W}$.

Algorithm 1 `GCC*`: GCC incorporating named entities and LLM

Require: Dataset $\mathcal{D}$, $f_{\ell\ell m}$, similarity threshold τ, number of clusters k
Ensure: Clustered documents
  $\rightsquigarrow$ Extract named entities $\mathcal{E}_d$ for each $d \in \mathcal{D}$
  $\rightsquigarrow$ Train Word2Vec on $\mathcal{D}$
  $\rightsquigarrow$ Initialize list $\mathcal{L}$ to store document pairs
  for each pair of docs (d_i, d_j) **do**
    $\rightsquigarrow$ Compute similarity between $\mathcal{E}_{d_i}$ and $\mathcal{E}_{d_j}$
    if similarity $> \tau$ **then**
      $\rightsquigarrow$ Append (d_i, d_j) to $\mathcal{L}$
    end if
  end for
  for each pair (d_i, d_j) in $\mathcal{L}$ **do**
    if d_i and d_j share **at least 3 common named entities then**
      $\rightsquigarrow$ Add edge between d_i and d_j
    end if
  end for
  $\rightsquigarrow$ Construct graph $\mathbf{A}_{ner}$ with docs as nodes and similarities as edges
  $\rightsquigarrow$ Generate embeddings $\mathbf{X}_{\ell\ell m}$ for each doc using $f_{\ell\ell m}$
  $\rightsquigarrow$ Apply `GCC` [11] on $(\mathbf{A}_{ner}, \mathbf{X}_{\ell\ell m})$ and obtain optimal $\mathbf{G}$, $\mathbf{F}$ and $\mathbf{W}$
  $\rightsquigarrow$ Deduce the clusters of documents from $\mathbf{G}$.

4 Experiments

4.1 Datasets

We assess our model using different configurations: $\mathbf{X}_{co}$ (Bag-of-Words), $\mathbf{X}_{\ell\ell m}$ (LLM embeddings), $\mathbf{A}_{knn}$ (KNN-based graph), and $\mathbf{A}_{ner}$ (NER-based graph). We test our model using four datasets that vary in size and number of clusters. The characteristics of these datasets are shown in Table 1.

Table 1. Description of datasets. The balance represents the ratio between the smallest and largest class. #Tokens indicates the mean token count.

Datasets	Characteristics					
	#Documents	#Clusters	Balance	#Tokens	Domaine	Language
BBC News[3]	2,225	5	0.75	390	News articles	En
MLSUM [30]	407,835	612	2.2e-05	543	News articles	Fr
Arxiv-10 [10]	100,000	10	1	155	Scientific papers	En
PubMed	19,716	3	0.52	224	Pubmed abstracts	En

4.2 Clustering algorithms

In this section, we evaluate various clustering algorithms using different input types: the document feature matrix $\mathbf{X}_.$, the adjacency matrix $\mathbf{A}_.$, or a combination of both $(\mathbf{X}_., \mathbf{A}_.)$. These matrices capture different aspects of the data:

- `Kmeans` [19], `Deep Kmeans` [9] and `DCN` [41] : Use $\mathbf{X}_{co}$ or $\mathbf{X}_{\ell\ell m}$.
- `Spectral clustering`: Uses only $\mathbf{A}_{ner}$ or $\mathbf{A}_{knn}$ for clustering.
- `PC Kmeans`: Uses $(\mathbf{X}_{\ell\ell m}, \mathbf{A}_{ner})$ or $(\mathbf{X}_{\ell\ell m}, \mathbf{A}_{knn})$. We retain the top 500 strongest links from the named entity graph $\mathbf{A}_*$ as must-link constraints. Increasing this number reduced model performance, so we enforced the 500-link limit.
- $\texttt{GCC}_{(.,.)}$: applied to $(\mathbf{A}_{knn}, \mathbf{X}_{co})$, $(\mathbf{A}_{ner}, \mathbf{X}_{co})$ and $(\mathbf{A}_{ner}, \mathbf{X}_{\ell\ell m})$ respectively.
- $\texttt{GCC}^*$: applied to $(\mathbf{A}_{ner}, \mathbf{X}_{\ell\ell m})$.

4.3 Experimental setting

Using labeled datasets, we evaluate clustering algorithms performance with external indices: Accuracy (ACC), Normalized Mutual Information (NMI) [33], and Adjusted Rand Index (ARI) [32]. ACC measures how well each cluster matches the ground-truth class labels, while NMI, ranging from 0 to 1, evaluates the mutual dependence between the predicted clusters and true labels. Finally, the ARI considers the similarity between predicted clusters and ground truth partitions, with a range from -0.5 to 1, where higher values indicate better agreement. Intuitively, NMI quantifies how much the estimated clustering is informative about the true clustering, while the ARI measures the degree of agreement between the estimated clustering and the reference partition. Both NMI and ARI are equal to 1 if the resulting clustering partition is identical to the ground truth. Contrary to ACC, it is important to note that NMI and ARI are more reliable as external indices because they are less sensitive to disproportionate classes.

For clustering algorithms, `Kmeans` is initialized using `K-means++` [4], which selects starting centroids by sampling based on their contribution to total inertia. We limit the iterations to 300 and run 10 initial setups to enhance clustering stability. For `Deep Kmeans` and `DCN`, we use the default settings to maintain consistency across experiments.

We remove stopwords and limit features to 2000 for co-occurrence matrices. LLM embeddings were generated using OpenAI's *text-embedding-3-small* model (1536 dimensions, $0.02 per 1M tokens). Documents exceeding 8,191 tokens were excluded to fit model limits. Due to computational constraints, we randomly sample 10,000 Arxiv documents and 16,321 MLSUM documents from five categories: Sport, Health, Politics, Economy, and Climate.

For the NER, we use *camembert-ner*[4] model for MLSUM since the dataset is in French, and DeBERTa [34] for BBC News. For other datasets, we used *GPT-4o*[5] with prompts to extract named entities in JSON format, where keys are entity types and values are lists of entities. For the named entity graph, we only

[4] https://huggingface.co/Jean-Baptiste/camembert-ner
[5] https://platform.openai.com/docs/models/gpt-4o

kept links where τ was greater than 0.9 between each pair of named entities of the same type. A link between two documents was considered only if there were at least three links between their named entities. Word2Vec was trained using CBOW with `num_features = 500`, `min_word_count = 10`, context window size of `5`, and 20 epochs. Composite named entities were merged to be treated as a single token by Word2Vec.

4.4 Assessing the number of clusters k and the power p

Optimizing the propagation parameter p and the number of clusters k is essential for the performance of our method. The parameter p controls the neighborhood information captured by the GCN and affects the smoothness of node embeddings. An appropriate p aggregates enough information without over-smoothing. Similarly, choosing the right number of clusters k ensures proper grouping of documents, avoiding over- or under-segmentation.

Number of clusters k. Since our framework is unsupervised, we aim to assess whether we could accurately detect the true number of clusters. Traditional internal criteria such as silhouette score [26], Davies-Bouldin index [7], and Calinski-Harabasz index [5] did not yield satisfactory results. Most of these metrics tend to favor the minimum number of clusters specified in the grid search. To address this limitation, we applied an alternative approach by running `GCC`* with a large number of clusters - 250 for BBC News and 500 for other datasets. This over-segmentation allows us to capture local patterns, where each cluster centroid represents a dense region in the feature space. We then applied hierarchical clustering using Ward's method [38] on the centroids. This method progressively merges clusters, and by observing the dendrogram (Figure 3), we selected the appropriate number of clusters. For most datasets, the true number of clusters was easily identified. For the arXiv dataset, several partitions can be considered, with 3, 4, or 10 clusters. To remain consistent with the benchmarks and our evaluation study of `GCC`*, we opted for 10 clusters.

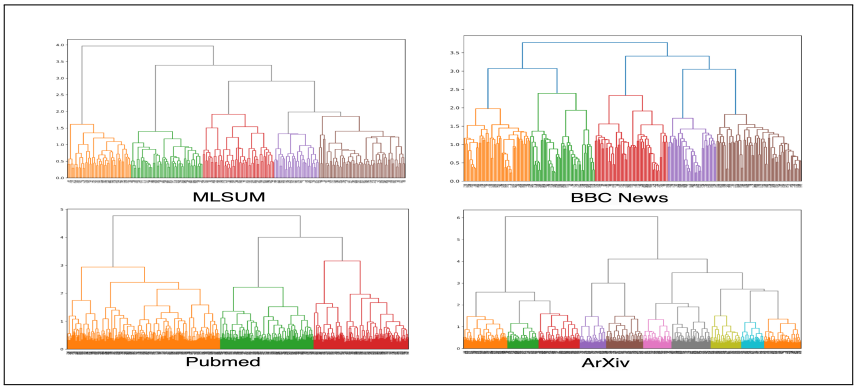

Fig. 3. Comparison of dendrograms for different datasets.

Power p. Once the number of classes has been set, our objective is to estimate p. We tested a range of values for p between 1 and 50 and selected the optimal value by minimizing the square root of the cluster loss. This ensures that the chosen p captures sufficient neighborhood information while avoiding oversmoothing of the graph signal. We observed that a value of $p = 2$ or $p = 3$ is the value retained for the 4 datasets; the difference between these two values is not significant.

4.5 Results

Quality of clustering. In Table 2 we compare the performance of different clustering algorithms on four datasets: BBC News, MLSUM, PubMed, and Arxiv-10. We observe that the algorithms using co-occurrence matrices perform significantly worse compared to those using LLM embeddings. For example, on the BBC News dataset, `Deep Kmeans` with $\mathbf{X}_{co}$ achieves the highest accuracy (66.45%), but its performance on larger datasets such as MLSUM and PubMed drops significantly (NMI $< 8\%$ and ARI $< 2\%$). This reflects the inability of co-occurrence-based representations to capture deeper semantic relationships. On the other hand, the methods incorporating LLM embeddings show a substantial improvement in clustering performance. For example, on the BBC News dataset, `Kmeans` achieves 93.01% ACC and 91.01 ARI, while `DCN` reaches 86.76% ACC. This improvement is consistent across datasets, highlighting the advantage of using rich contextual embeddings.

The proposed approach (`GCC`*) on $(\mathbf{A}_{ner}, \mathbf{X}_{\ell\ell m})$, exceeds all other methods, including KNN graph-based clustering. In the BBC News dataset, `GCC`* achieves a remarkable NMI of 95.12%, demonstrating that the incorporation of the similarities of the named entities into the graph structure significantly improves clustering performance. A similar pattern is seen in other datasets, such as MLSUM, where `GCC`* achieves an NMI of 72.42%. The results of spectral clustering using different adjacency matrices $\mathbf{A}$ show that simply using $\mathbf{A}_{knn}$ or $\mathbf{A}_{ner}$ is not sufficient to achieve good clustering performance. The main challenge lies in the structure of these adjacency matrices, which fail to capture meaningful relationships for clustering in certain scenarios. Indeed, the use of $\mathbf{A}_{knn}$ often results

Table 2. Clustering performance on four datasets averaged over 20 runs

Method	Input	BBC			MLSUM			PubMed			ArXiv-10		
		ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI
`Kmeans`	$\mathbf{X}_{co}$	43.91	27.16	13.47	24.86	0.60	0.40	44.16	14.12	8.77	37.54	29.80	12.30
`DKmeans`	$\mathbf{X}_{co}$	66.45	56.11	47.73	25.63	0.61	0.42	44.83	7.86	1.89	41.23	31.19	16.67
`DCN`	$\mathbf{X}_{co}$	58.58	40.49	25.95	26.87	2.01	0.99	45.69	15.41	8.45	35.85	27.56	12.63
`Kmeans`	$\mathbf{X}_{\ell\ell m}$	93.01	87.34	91.01	75.62	66.28	62.60	63.25	27.02	23.58	69.08	60.11	51.74
`DKmeans`	$\mathbf{X}_{\ell\ell m}$	85.51	75.10	73.62	75.01	58.25	57.41	55.54	17.85	13.90	69.26	59.62	51.71
`DCN`	$\mathbf{X}_{\ell\ell m}$	86.76	72.78	71.85	70.14	52.99	49.04	58.71	25.44	18.32	66.45	56.11	47.73
`Spectral`	$\mathbf{A}_{knn,\ell\ell m}$	34.11	19.32	3.82	31.90	0.9	0.10	40.01	0.08	0.06	14.86	5.32	0.6
`Spectral`	$\mathbf{A}_{ner,\ell\ell m}$	35.01	20.01	7.87	31.33	1.23	0.20	39.93	0.20	0.0	20.01	6.31	2.12
`PCKmeans`	$\mathbf{A}_{knn}, \mathbf{X}_{\ell\ell m}$	70.85	73.00	63.89	70.01	49.00	45.80	60.01	24.24	22.87	68.12	60.90	52.14
`PCKmeans`	$\mathbf{A}_{ner}, \mathbf{X}_{\ell\ell m}$	70.88	73.06	64.01	70.71	49.87	45.88	60.21	24.89	23.06	69.01	61.90	52.34
`GCC`$_{(knn,co)}$	$\mathbf{A}_{knn}, \mathbf{X}_{co}$	88.76	76.26	75.97	25.20	0.60	0.71	63.02	25.20	24.78	60.12	47.75	39.08
`GCC`$_{(ner,co)}$	$\mathbf{A}_{ner}, \mathbf{X}_{co}$	95.43	86.18	89.27	27.69	2.91	3.80	63.00	25.23	24.80	60.01	49.57	40.01
`GCC`$_{(knn,\ell\ell m)}$	$\mathbf{A}_{knn}, \mathbf{X}_{\ell\ell m}$	95.82	87.41	90.12	76.56	67.12	64.99	65.00	28.34	**25.00**	70.33	60.70	52.01
`GCC`*	$\mathbf{A}_{ner}, \mathbf{X}_{\ell\ell m}$	**97.61**	**95.12**	**96.66**	**88.32**	**72.42**	**74.58**	**65.10**	**29.04**	**25.00**	**73.2**	**63.98**	**55.8**

in too many links between nodes, leading to a densely connected graph. This excessive connectivity diminishes the quality of the clusters as many unrelated points are grouped together. For example, on the BBC dataset, spectral clustering with $\mathbf{A}_{knn,\ell\ell m}$ achieves an ARI of only 3.82%, highlighting the negative impact of overly dense connections. However, although $\mathbf{A}_{ner}$ captures entity-level information, it tends to produce a sparse graph, especially in large and diverse documents. This sparseness leads to weakly connected clusters, as seen in the PubMed dataset, where spectral clustering with $\mathbf{A}_{ner}$ yields an NMI close to zero. The poor performance of spectral clustering emphasizes the need for more balanced graph structures that reflect both semantic and entity-level similarities.

Quality of embedding. We evaluate the quality of the embedding through the visualization of the truth classes using UMAP (default parameters) based on $\mathbf{X}_{\ell\ell m} \in \mathbb{R}^{n\times d}$ and $\mathbf{Y}^p\mathbf{W} \in \mathbb{R}^{n\times k}$ which is derived from GCC*. It is remarkable to observe the quality of class separability that we illustrate in Figure 4.

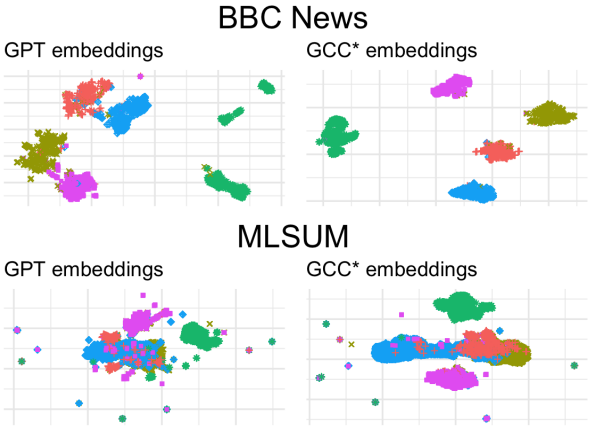

Fig. 4. UMAP projection of the cluster embeddings obtained with GPT ($\mathbf{X}_{\ell\ell m}$) compared to those obtained on $\mathbf{Y}^p\mathbf{W}$ derived from GCC*.

5 CONCLUSION

This study introduces an innovative method for document clustering that integrates NER and LLM embeddings in the attributed graphs framework. By building a document graph using entity similarities and incorporating contextual embeddings, our approach outperformed traditional clustering techniques. The application of GCN facilitates the simultaneous optimization of embeddings and clustering, leading to more effective grouping of semantically similar documents.

Future work could focus on exploring whether specific types of named entities (e.g., organizations, locations, or events) are more relevant than others for certain clustering tasks. This direction could provide more refined document groupings and reveal the underlying thematic relationships driven by particular entity types.

References

1. Affeldt, S., Labiod, L., and Nadif, M. Regularized bi-directional co-clustering. *Statistics and Computing 31*, 3 (2021), 32.
2. Allab, K., Labiod, L., and Nadif, M. A semi-nmf-pca unified framework for data clustering. *IEEE TKDE 29*, 1 (2016), 2–16.
3. Allab, K., Labiod, L., and Nadif, M. Simultaneous spectral data embedding and clustering. *IEEE TNNLS 29*, 12 (2018), 6396–6401.
4. Arthur, D., and Vassilvitskii, S. k-means++: The advantages of careful seeding. Tech. rep., Stanford, 2006.
5. Caliński, T., and Harabasz, J. A dendrite method for cluster analysis. *Communications in Statistics-theory and Methods 3*, 1 (1974), 1–27.
6. Cao, T. H., Tang, T. M., and Chau, C. K. Text clustering with named entities: a model, experimentation and realization. In *Data Mining: Foundations and Intelligent Paradigms*. 2012, pp. 267–287.
7. Davies, D. L., and Bouldin, D. W. A cluster separation measure. *IEEE transactions on pattern analysis and machine intelligence*, 2 (1979), 224–227.
8. Derczynski, L., Maynard, D., Rizzo, G., Van Erp, M., Gorrell, G., Troncy, R., Petrak, J., and Bontcheva, K. Analysis of named entity recognition and linking for tweets. *Information Processing & Management 51*, 2 (2015), 32–49.
9. Fard, M. M., Thonet, T., and Gaussier, E. Deep k-means: Jointly clustering with k-means and learning representations. *Pattern Recognition Letters 138* (2020), 185–192.
10. Farhangi, A., Sui, N., Hua, N., Bai, H., Huang, A., and Guo, Z. Protoformer: Embedding prototypes for transformers. In *PAKDD* (2022), pp. 447–458.
11. Fettal, C., Labiod, L., and Nadif, M. Efficient graph convolution for joint node representation learning and clustering. In *WSDM* (2022), pp. 289–297.
12. Hassani, K., and Khasahmadi, A. H. Contrastive multi-view representation learning on graphs. In *ICML* (2020), pp. 4116–4126.
13. Keraghel, I., Morbieu, S., and Nadif, M. Beyond words: a comparative analysis of llm embeddings for effective clustering. In *IDA* (2024), pp. 205–216.
14. Kim, J.-H., Choi, J.-H., Park, Y.-H., Leung, C. K.-S., and Nasridinov, A. Knn-sc: novel spectral clustering algorithm using k-nearest neighbors. *IEEE Access 9* (2021), 152616–152627.
15. Kipf, T. N., and Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).
16. Kipf, T. N., and Welling, M. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308* (2016).
17. Labiod, L., and Nadif, M. Efficient regularized spectral data embedding. *Advances in Data Analysis and Classification 15*, 1 (2021), 99–119.
18. Lee, J., and Toutanova, K. Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805 3*, 8 (2018).
19. Macqueen, J. Some methods for classification and analysis of multivariate observations. In *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability/University of California Press* (1967).
20. Mikolov, T. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).
21. Muennighoff, N., Tazi, N., Magne, L., and Reimers, N. Mteb: Massive text embedding benchmark. *arXiv preprint arXiv:2210.07316* (2022).

22. Ng, A., Jordan, M., and Weiss, Y. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems 14* (2001).
23. Qin, Y., Yu, Z. L., Wang, C.-D., Gu, Z., and Li, Y. A novel clustering method based on hybrid k-nearest-neighbor graph. *Pattern recognition 74* (2018), 1–14.
24. Radford, A. Improving language understanding by generative pre-training.
25. Reimers, N. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084* (2019).
26. Rousseeuw, P. J. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics 20* (1987), 53–65.
27. Salah, A., and Nadif, M. Directional co-clustering. *Advances in Data Analysis and Classification 13* (2019), 591–620.
28. Salha, G., Hennequin, R., and Vazirgiannis, M. Simple and effective graph autoencoders with one-hop linear models. In *ECML-PKDD* (2021), pp. 319–334.
29. Schütze, H., Manning, C. D., and Raghavan, P. *Introduction to information retrieval*, vol. 39. Cambridge University Press Cambridge, 2008.
30. Scialom, T., Dray, P.-A., Lamprier, S., Piwowarski, B., and Staiano, J. Mlsum: The multilingual summarization corpus. *arXiv preprint arXiv:2004.14900* (2020).
31. Sia, S., Dalmia, A., and Mielke, S. J. Tired of topic models? clusters of pretrained word embeddings make for fast and good topics too! *arXiv preprint arXiv:2004.14914* (2020).
32. Steinley, D. Properties of the hubert-arable adjusted rand index. *Psychological methods 9*, 3 (2004), 386.
33. Strehl, A., and Ghosh, J. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *JMLR 3* (2002), 583–617.
34. Ushio, A., and Camacho-Collados, J. T-NER: An all-round python library for transformer-based named entity recognition. In *EACL* (2021), pp. 53–62.
35. Vaswani, A. Attention is all you need. *Advances in Neural Information Processing Systems* (2017).
36. Velickovic, P., Fedus, W., Hamilton, W. L., Liò, P., Bengio, Y., and Hjelm, R. D. Deep graph infomax. *ICLR (Poster) 2*, 3 (2019), 4.
37. Wang, D., Cui, P., and Zhu, W. Structural deep network embedding. In *SIGKDD* (2016), pp. 1225–1234.
38. Ward Jr, J. H. Hierarchical grouping to optimize an objective function. *Journal of the American statistical association 58*, 301 (1963), 236–244.
39. Wei, T., Lu, Y., Chang, H., Zhou, Q., and Bao, X. A semantic approach for text clustering using wordnet and lexical chains. *Expert Systems with applications 42*, 4 (2015), 2264–2275.
40. Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., and Weinberger, K. Simplifying graph convolutional networks. In *ICML* (2019), pp. 6861–6871.
41. Yang, B., Fu, X., Sidiropoulos, N. D., and Hong, M. Towards k-means-friendly spaces: Simultaneous deep learning and clustering. In *ICML* (2017), pp. 3861–3870.
42. You, Y., Chen, T., Sui, Y., Chen, T., Wang, Z., and Shen, Y. Graph contrastive learning with augmentations. *Advances in neural information processing systems 33* (2020), 5812–5823.
43. Zhang, J., Zhang, H., Xia, C., and Sun, L. Graph-bert: Only attention is needed for learning graph representations. *arXiv preprint arXiv:2001.05140* (2020).