GAE-Based Document Embedding Method for Clustering

SUNGWON JUNG AND SANGMIN KA
Department of Computer Science and Engineering, Sogang University, Seoul 121-742, South Korea
Corresponding author: Sungwon Jung (jungsung@sogang.ac.kr)

This work was supported by the Institute of Information and Communications Technology Planning and Evaluation (IITP) Grant through the Korea Government [Ministry of Science and ICT (MSIT)], Development of High-Speed Analysis of Distributed Large-Scale Data for Wide Usability of Various Industries, under Grant 2021-0-00180.

ABSTRACT

Document embedding methods for clustering using deep neural networks have been proposed recently. However, the existing deep neural network-based document embedding methods for clustering have a problem of either generating document embeddings dependent on a given number of document clusters or generating document embeddings that do not take into account the characteristic of high similarity between documents belonging to the same document cluster. In this paper, we propose a new document embedding method for clustering by using a graph autoencoder. To this end, we construct an undirected and weighted sparse graph from a set of documents wherein each document is represented by a node, and all the weighted edges created in the graph have high cosine similarities between the two end nodes. We then apply the proposed graph autoencoder to the graph to compute node embedding vectors. Each node embedding vector in the graph is used as a document embedding vector. This paper presents in-depth experimental analyses of the proposed method. Experimental results on various real document data sets demonstrate that the proposed approach affords the significant performance improvement over the existing document embedding methods.

INDEX TERMS

Document embedding, text embedding, document clustering, graph autoencoder, graph CNN, autoencoder, deep neural network.

I. INTRODUCTION

In recent years, research on natural language processing and text mining techniques is being actively conducted owing to the increase in digital text documents on the Internet. In particular, there is no label information of data owing to the nature of online media where data is accumulated in real time; therefore, the demand for clustering for expansion or new insights based on unsupervised learning are increasing [1]. Clustering is an unsupervised learning-based algorithm that divides data into multiple clusters using information from given data without label information, and is in the spotlight in many fields from computer science to social network [2].

Traditional document clustering methods represent documents as very high-dimensional numeric vectors by using Bag Of Words (BOW) concept [3] or Term Frequency Inverse Document Frequency (TF-IDF) [4]. The very high-dimensional numeric vectors of documents are then clustered by the well-known partitional or hierarchical clustering algorithms [5]. However, the performance of these traditional document clustering methods was not good due to the high dimensionality of document vectors. To address this problem, Deep Neural Network (DNN)-based document embedding methods have been proposed in order to reduce extremely high-dimensional vectors to low-dimensional vectors [6], [7], [8], [9], [10].

Deep Embedded Clustering (DEC) [6] has been proposed to simultaneously learn feature representations and centroids of clusters. DEC uses a pre-trained multi-layer autoencoder to map the input data to latent feature space and simultaneously learns an optimized cluster by repeatedly redefining the centroid of the cluster through an auxiliary distribution derived from a soft assignment from the data point. However, the local structure preservation cannot be guaranteed by its
clustering loss. Consequently, the feature transformation may be misguided, leading to corruption of embedded space [11]. Furthermore, DEC computes the embedding vectors depending on a given $K$ number of clusters. If $K$ is changed, DEC has a problem of recomputing the whole embedding vectors.

Spectral Clustering with Deep Embedding (SCDE) [7] has been proposed to improve spectral clustering using the deep autoencoder. SCDE first computes good embedding of the raw document data using the deep autoencoder of DEC [6] and then applies spectral clustering on the embedding to cluster documents. It also proposes a method to estimate the number of clusters based on a softmax autoencoder. However, SCDE has one major problem. It uses the deep autoencoder of DEC to obtain the document embedding vectors for the application of the spectral clustering method. As a result, it suffers from the same problems as those based on DEC.

In spherical text embedding [9], Joint Spherical Embedding (JoSE) is a method that learns spherical text embeddings in the spherical space in an unsupervised manner. To this end, a two-step generative model is employed that jointly learns unsupervised word and paragraph embeddings by exploiting word-word and word-paragraph co-occurrence statistics. JoSE demonstrates much better clustering performance than the existing document (or paragraph) embedding methods such as averaged word embedding using Word2Vec [12], SIF [13], BERT [8], and Doc2Vec [14].

Sentence-BERT(SBERT) [10] is a modification of the pretrained BERT network that uses Siamese and triplet network structures. It can derive semantically meaningful sentence embeddings for large-scale semantic similarity comparison, unsupervised tasks such as clustering, and information retrieval via semantic search. Previously [10], the performance of SBERT for common semantic textual similarity tasks was evaluated, wherein it outperformed other state-of-the-art sentence embedding methods. However, the document embeddings of SBERT for clustering are not effective for the document datasets where its member documents have a large number of words. SBERT (Model: all-mpnet-base-V2) affords an effective document embedding only for a document with a maximum of 384 words.

In this paper, we propose a new document embedding method for clustering using a Graph AutoEncoder (GAE). For this purpose, we first propose an effective graph construction algorithm from a set of documents. The constructed graph is sparse where a node represents a document, and a weighted edge represents a cosine similarity between documents. We then propose our GAE to compute node embedding vectors from the constructed graph. We use each node embedding vector as a document embedding vector for clustering. These document embedding vectors are independent of the number of document clusters. Consequently, they can be reused for $K$-mean clustering algorithm for any value of $K$ without recomputing the document embedding vectors with different $K$ values such as SCDE and DEC.

The main contributions of our proposed method are summarized as follows:

- We propose an efficient algorithm to construct a sparse graph from a set of documents.
- We develop an effective document embedding method optimized for clustering using a GAE.
- The proposed GAE-based document embedding method does not depend on the number $K$ of document clusters.

The remainder of this paper is organized as follows. Section II first describes how to construct an undirected and weighted sparse graph from a set of documents. It then describes the GAE model architecture which computes document(node) embedding vectors. Extensive experimental analyses of our document embedding method are presented in Section III. Finally, Section IV provides our concluding remarks.

II. METHOD: GAE-BASED DOCUMENT EMBEDDING METHOD

In this section, we describe a document embedding method for clustering with a graph autoencoder model. We first describe constructing a sparse weighted graph from a set of documents using GAE. We then describe our GAE model architecture from which document embeddings are computed as node embedding vectors.

A. GRAPH MODELING

Given a set $D = \{d_1, \ldots, d_N\}$ of documents where each $d_i$ represents a $m$-dimensional $\text{tf-idf}$ vector of a document $i$, we construct an undirected and weighted graph $G(V, E)$ to use a graph convolution neural network. We model each document $d_i$ in $D$ as a node in the graph $G(V, E)$ where all the undirected and weighted edges in $E$ are created only when their two end nodes have large cosine similarities. These cosine similarities between two nodes are used as the weights of edges in the graph. Thus, very high dimensional $\text{tf-idf}$ vectors of all the documents are transformed into an undirected and weighted graph structure.

To identify the edges having large cosine similarities between documents, we first construct the cosine similarity matrix $C$ for the document set $D = \{d_1, \ldots, d_N\}$ as follows. Each entry $C_{ij}$ in the matrix $C$ represents the cosine similarity between the documents $d_i$ and $d_j$. In $C$, all the diagonal entries $C_{ii}$ are set to zero.

$$C = \begin{bmatrix}
0 & C_{1,2} & \cdots & C_{1,N} \\
C_{2,1} & 0 & \cdots & C_{2,N} \\
\vdots & \vdots & \ddots & \vdots \\
C_{N,1} & C_{N,2} & \cdots & 0
\end{bmatrix}$$

By using the largest cosine similarity of each row in the $C$ matrix, we then normalize all the entries of the cosine similarity matrix $C$ as the following matrix $MC$. $MC_i$ in $NC$ is defined as $MC_i = \max_{1 \leq j \leq N} C_{ij}$ and
The various document datasets that only top K tf-idf is formally defined as NC. We construct the the weighted edge set NC for the edges of the node NC. We define the set TopKi for each document di from NC as follows.

Definition 2.1: Given the matrix NC constructed from D = {d1, ..., dN}, TopKi contains the top K largest normalized cosine similarities between the document di and all the other documents d1, d2, ..., dN. Most of NC1,1, NC1,2, ..., NC1,N of each document di are very small and similar since tf-idf vectors of documents are very sparse and high dimensional. We have experimentally found from the various document datasets that only top K largest NCi,j similarities are large enough to be considered as the edges of the document di’s close similar documents. We define the set TopKi for each document di from NC as follows.

\[
NC = \begin{bmatrix}
0 & C_{i,2} & \cdots & C_{i,N} \\
C_{2,1} & 0 & \cdots & C_{2,N} \\
\vdots & \vdots & \ddots & \vdots \\
C_{N,1} & C_{N,2} & \cdots & 0
\end{bmatrix}
\]

All the entries NCi,j of a row i in NC represent the normalized cosine similarities between the document di and all the other documents d1, d2, ..., dN. Most of NCi,1, NCi,2, ..., NCi,N of each document di are very small and similar since tf-idf vectors of documents are very sparse and high dimensional. We have experimentally found from the various document datasets that only top K largest NCi,j similarities are large enough to be considered as the edges of the document di’s close similar documents. We define the set TopKi for each document di from NC as follows.

\[
\text{Definition 2.1: Given the matrix } NC \text{ constructed from } D = \{d_1, \ldots, d_N\}, \text{ TopKi contains the top } K \text{ largest normalized cosine similarities between the document } d_i \text{ and all the other documents } d_1, d_2, \ldots, d_N. \text{ Most of } NC_{i,1}, NC_{i,2}, \ldots, NC_{i,N} \text{ of each document } d_i \text{ are very small and similar since tf-idf vectors of documents are very sparse and high dimensional. We have experimentally found from the various document datasets that only top } K \text{ largest } NC_{i,j} \text{ similarities are large enough to be considered as the edges of the document } d_i \text{'s close similar documents. We define the set TopKi for each document } d_i \text{ from } NC \text{ as follows.}
\]

\[
D_{i,j} = C_{i,j} / MC_i.
\]

\[
NC = \begin{bmatrix}
0 & C_{i,2} & \cdots & C_{i,N} \\
C_{2,1} & 0 & \cdots & C_{2,N} \\
\vdots & \vdots & \ddots & \vdots \\
C_{N,1} & C_{N,2} & \cdots & 0
\end{bmatrix}
\]

By using TopKi for each document di in D = {d1, ..., dN}, we construct the the weighted edge set E for G(V, E) with V = D. Some of K edges in TopKi may not be created for the edges of the node di. Given TopKi = {(d1, d2, NCi,j), (d2, d3, NCi,j), ..., (di, d4, NCi,j))} for each node di, we use Kth largest NCi,j as a threshold to determine if an edge with weight NCi,j between the nodes di and dj is going to be created. Let Z contain a set of triples selected from TopKi so far as the created edges of node di. Suppose that we select an edge (di, dj, NCi,j) from TopKi to check if it can be created as the next edge of the node di. We create an edge (di, dj, NCi,j) only when there exists at least one edge (di, dj, NCi,j) in Z such that NCi,j ≥ α * NCi,j. If the above condition holds, then we add (di, dj, NCi,j) to Z. Here, α is a real constant value between 0 and 1. This condition ensures that each edge (di, dj, NCi,j) in TopKi is selected as a direct edge of node di only when the node dj has the normalized cosine similarity larger than or equal to α * NCi,j with some of the neighbor nodes in the created edges of the node di. Consequently, the constructed graph is sparse.

The nodes in the graph G(V, E) constructed from this method are directly connected to only those nodes with large cosine similarities with them. The similarities among documents are captured by a graph structure where the weights of the edge represents the degree of similarity between two nodes (i.e., documents). Algorithm 1 gives the graph G(V, E) construction method from a set of documents.

### B. GAE Model Architecture

We construct the weighted adjacent matrix A_{N×N} and the degree matrix D_{N×N} from G(V, E) obtained from

Algorithm 1 Graph Construction(\(D, K, \alpha\))

**INPUT** \(D\): A set \{d1, ..., dN\} of documents

\(K\): Top K

\(\alpha\): \(0 < \alpha \leq 1\)

**OUTPUT** \(G(V, E)\): Graph constructed from \(D\)

1: \(E \leftarrow \emptyset\)

2: Compute the matrix NC from \(D\)

3: for each \(d_i \in D\) do

4: Compute TopKi from NC using the definition 2.1

5: for \(i = 1\) to \(N\) do

6: Get the \(K^{th}\) triple \((d_i, d_{i,k}, NC_{i,k})\) from TopKi

7: if \(i = 1\) or \(i = K\) then

8: \(Z \leftarrow \{(d_i, d_{i,1}, NC_{i,1})\}\)

9: end

10: for \(j = 2\) to \(K\) do

11: Let \(w\) be \(NC_{i,j}\) from \((d_i, d_{i,j}, NC_{i,j})\) in TopKi

12: for each \((d_i, d_{i,k}, NC_{i,k})\) in TopKi do

13: if \(NC_{i,k} \geq \alpha\) then

14: \(Z \leftarrow Z \cup \{(d_i, d_{i,k}, w)\}\)

15: end

16: \(E \leftarrow E \cup Z\)

17: \(V \leftarrow D\)

18: return \(G(V, E)\)

Algorithm 1. The diagonal entries \(D_{i,i}\) of the degree matrix \(D\) have the degree of node \(i\) of the graph and the rest of the entries of \(D\) are 0s. We also construct the unweighted adjacency matrix \(M_{N×N}\) from the matrix \(A\). The entry \(M_{u,v}\) of the matrix \(M\) is defined as

\[
M_{u,v} = \begin{cases} 
1 & \text{if } A_{u,v} > 0 \\
0 & \text{otherwise.}
\end{cases}
\]

We normalize the weighted adjacency matrix \(A\) for GAE using \(A_{N×N}\), \(D_{N×N}\), and an identity matrix \(I_{N×N}\). The normalized weighted adjacency matrix \(\hat{A}\) is defined as follows:

\[
\hat{A} = D^{-1/2}(A + I)D^{-1/2}
\]

We use a multi-layer GAE to generate the node embedding vectors \(Z\) of the nodes (i.e., documents) in the graph \(G\) with the following layer-wise propagation rule of the encoder:

\[
H^{(l+1)} = \text{ReLU}(\hat{A}H^{(l)}W^{(l)} + b^{(l)})
\]

One hot encoding vector of \(N\) dimension is used as a feature vector of each node in the graph. We do not use the \(tf-idf\) vector of a document as the feature vector of the corresponding node in the graph. Let \(X_{N×N}\) be a matrix consisting of the \(N\)-dimension feature vectors of the \(N\) nodes in the graph. \(W^{(l)}\) and \(b^{(l)}\) are layer-specific trainable weight and bias matrices.
The decoder of our GAE computes the pair-wise similarity between the node embedding vectors of the final layer of the encoder. Then, the decoder reconstructs the unweighted graph adjacency matrix \( \hat{M} \), which is defined as

\[
\hat{M}_{u,v} = \text{dec}(z_u, z_v) = \sigma(z_u^T z_v)
\]

where \( z_u \) is the embedding vector of node \( u \) and \( \sigma(\cdot) \) denote a sigmoid activation function. We train our proposed GAE by minimizing the negative cross-entropy loss given the above-unweighted adjacency matrix \( M \) and the reconstructed adjacency matrix \( \hat{M} \).

We define the pair of connected nodes in the graph as positive samples and the pair of unconnected nodes as negative samples to compute the negative cross-entropy loss function. Positive samples refer to all the edges (i.e., connected nodes) present in the graph \( G(V, E) \). For each training epoch, the different negative samples are generated equal to the number of positive samples (i.e., \( |E| \)) through random sampling among unconnected nodes in the graph. Let \( E_{\text{neg}} \) be a set of negative samples where \( |E| = |E_{\text{neg}}| \). The negative cross-entropy loss function of our proposed GAE is given as follows:

\[
\text{Loss} = - \frac{1}{N} \sum_{(u,v) \in E} \log(\sigma(Z_u \cdot Z_v^T))
\]

\[
- \frac{1}{N} \sum_{(u,v) \in E_{\text{neg}}} \log(1 - \sigma(Z_u \cdot Z_v^T))
\]

Our loss function enforces the connected nodes with edges to have similar node embedding vectors and unconnected nodes without edges to have dissimilar node embedding vectors.

When all the layers are trained in this manner, we use the output of the final layer of the encoder as the final node embedding vectors. We then run K-means and spherical K-Means (SK-Means) algorithms on these node embedding vectors to compute document clusters.

### III. EXPERIMENTS

In this section, we present the experimental results of several document datasets to evaluate our proposed document embedding method. We compare our proposed GAE-based Document Embedding Method (GDEM) with the following baseline document embedding methods: DEC [6], SCDE [7], JoSE(Joint Spherical Embedding) [9], and SBERT(Model: all-mpnet-base-v2) [10]. The sentence and paragraph in SBERT can be mapped to the 768-dimensional density vector space and can be used for tasks such as clustering or semantic search. SBERT performs the best at 2022.09.

### A. DATASETS

We performed experiments on the following six text datasets, which are widely used for evaluating the performance of document clustering methods.

- **20Newsgroups**: It is a collection of 18,846 text documents categorized into 20 different newsgroups. All the meaningful words from the dataset are extracted to compute \( tf-idf \) vectors of size 100259 for GDEM.
- **BBC**: It consists of 2225 documents from the BBC news website corresponding to stories in five topical areas from 2004-2005. All the meaningful words from the dataset are extracted to compute \( tf-idf \) vectors of size 37644 for GDEM.
- **Reuter10k**: Reuter dataset contains a text corpus that contains 804,414 English documents categorized into 103 different topics. The Reuter-10 dataset extracted a subset of the corpus that contains 4 topics and 10,000 documents with a single topic label. All the meaningful words from the dataset are extracted to compute \( tf-idf \) vectors of size 28628 for GDEM.
- **Reuter-8**: Reuter dataset contains a text corpus that contains 804,414 English documents categorized into 103 different topics. The Reuter-8 dataset extracted a subset of the corpus that contains 8 topics and 7,674 documents with a single topic label. All the meaningful words from the dataset are extracted to compute \( tf-idf \) vectors of size 21088 for GDEM.
- **Opinosis**: Opinosis dataset contains sentences extracted from user reviews on a given topic. In total there are 51 such topics, with each topic having approximately 100 sentences. All the meaningful words from the dataset are extracted to compute \( tf-idf \) vectors of size 5263 for GDEM.
- **WebKB**: WebKB is a dataset that includes web pages from computer science departments of various universities. 4,518 web pages are categorized into 6 imbalanced categories. All the meaningful words from the dataset are extracted to compute \( tf-idf \) vectors of size 6887 for GDEM.

Table 1 summarizes the detailed characteristics of the six document datasets.

### B. EVALUATION METRIC

We use three widely used external measures as metrics: Normalized Mutual Information (NMI) [15], Adjusted Rand Index (ARI) [16], and Clustering Accuracy (ACC) [6]. MNI has a range of \([0,1]\), with one being the perfect clustering and zero the worst, and ARI has a range of \([-1,1]\), with one being the best clustering performance and minus one the worst. ACC gives the ratio between the number of clusters to the number of ground-truth categories. ACC also has a range of \([0,1]\), with one being the perfect clustering and zero the worst.

![Table 1. Characteristics of document datasets.](image-url)
C. IMPLEMENTATION

We first construct a graph $G(V, E)$ from a given set $D$ of documents by using Algorithm 1 where $\alpha = 0.2$. We then implemented our proposed GAE to have two layers of the encoder to generate the node (i.e., document) embedding vectors $Z$ of the graph $G(V, E)$ as follows.

$$GAE(X, \hat{A}) = \hat{A} \cdot ReLU(\hat{A}H^{(0)}W^{(0)} + B^{(0)})W^{(1)} + B^{(1)}$$

Two-layers learnable weight and bias matrices $W^{(0)}$, $B^{(0)}$, $W^{(1)}$ and $B^{(1)}$ are used. $H^{(0)}$ is a matrix $X$ consisting of the $N$-dimension one-hot encoding feature vectors of the $N$ nodes in the graph.

We implemented all algorithms based on Python, PyTorch, and PyTorch Geometric. We used different encoder architectures depending on the size of the dataset. We used **128-64 dimensions** for 20newsgroups and **64-32 dimensions** for the rest of the document datasets as the GAE encoder architecture. We set $K$ to 20 for 20newsgroups and set $K$ to 50 for the rest of the datasets to compute $TopK_i$ for each dataset. We trained GAE for 200 iterations using *Adam* with a learning rate of 0.01. We use both K-Means and SK-Means for clustering methods.

We visualize the effectiveness of GDEM by showing the t-SNE of the document embedding vectors after learning 10 and 200 epochs of GDEM for the six datasets in figure 1. Figure 1(a), (b), (c) and (d), (e), (f) are the results of two-dimensional mapping with t-SNE distribution after learning 10 epoch for 20Newsgroups, BBC, Reuter10k, Reuter8, Opinionis, and WebKB datasets. Similarly, figure 1(a.1), (b.1), (c.1), and (d.1), (e.1), (f.1) are the results of two-dimensional mapping with t-SNE distribution after learning 200 epochs for the six document datasets. Figure 1 demonstrates that the document embedding vectors of GDEM for each dataset are significantly better clustered after 200 epochs than after 10 epochs.

**FIGURE 1.** T-SNE results of document embedding vectors after 200 epochs.

**FIGURE 2.** $K$-means clustering comparison on ARI: with respect to the number $K$ of clusters.
TABLE 2. Document clustering evaluation with K-means and SK-means.

| Method | Eval. | 20News | BBC | WebKB | Opinion | Reuter8 | Reuter10k |
|--------|-------|--------|-----|-------|---------|--------|----------|
| DEC [6] | K-means | ACC 0.117 | 0.576 | 0.579 | 0.256 | 0.481 | 0.760 |
| | | ARI 0.051 | 0.425 | 0.350 | 0.191 | 0.352 | 0.537 |
| | | NMI 0.100 | 0.512 | 0.396 | 0.047 | 0.524 | 0.481 |
| | SK-means | ACC 0.116 | 0.539 | 0.587 | 0.183 | 0.578 | 0.758 |
| | | ARI 0.053 | 0.393 | 0.346 | 0.078 | 0.457 | 0.554 |
| | | NMI 0.009 | 0.491 | 0.387 | 0.319 | 0.544 | 0.547 |
| SCDE [7] | K-means | ACC 0.421 | 0.893 | 0.584 | 0.376 | 0.499 | 0.533 |
| | | ARI 0.247 | 0.756 | 0.303 | 0.261 | 0.341 | 0.174 |
| | | NMI 0.364 | 0.741 | 0.299 | 0.502 | 0.476 | 0.274 |
| | SK-means | ACC 0.423 | 0.892 | 0.569 | 0.373 | 0.503 | 0.533 |
| | | ARI 0.252 | 0.754 | 0.278 | 0.261 | 0.343 | 0.173 |
| | | NMI 0.365 | 0.739 | 0.278 | 0.509 | 0.468 | 0.274 |
| JoSE [9] | K-means | ACC 0.721 | 0.698 | 0.400 | 0.227 | 0.387 | 0.265 |
| | | ARI 0.581 | 0.617 | 0.251 | 0.139 | 0.475 | 0.000 |
| | | NMI 0.684 | 0.704 | 0.370 | 0.343 | 0.293 | 0.000 |
| | SK-means | ACC 0.677 | 0.944 | 0.462 | 0.257 | 0.390 | 0.260 |
| | | ARI 0.556 | 0.868 | 0.315 | 0.166 | 0.469 | 0.000 |
| | | NMI 0.664 | 0.826 | 0.371 | 0.363 | 0.206 | 0.000 |
| SBERT [10] | K-means | ACC 0.639 | 0.965 | 0.434 | 0.473 | 0.576 | 0.368 |
| | | ARI 0.515 | 0.917 | 0.079 | 0.365 | 0.470 | 0.002 |
| | | NMI 0.656 | 0.895 | 0.096 | 0.643 | 0.637 | 0.000 |
| | SK-means | ACC 0.670 | 0.958 | 0.533 | 0.476 | 0.618 | 0.367 |
| | | ARI 0.523 | 0.902 | 0.188 | 0.373 | 0.479 | 0.000 |
| | | NMI 0.657 | 0.877 | 0.169 | 0.656 | 0.652 | 0.000 |
| GDEM (ours) | K-means | ACC 0.757 | 0.962 | 0.690 | 0.400 | 0.410 | 0.817 |
| | | ARI 0.600 | 0.911 | 0.422 | 0.260 | 0.311 | 0.644 |
| | | NMI 0.710 | 0.880 | 0.419 | 0.502 | 0.536 | 0.614 |
| | SK-means | ACC 0.791 | 0.960 | 0.702 | 0.411 | 0.392 | 0.820 |
| | | ARI 0.655 | 0.907 | 0.446 | 0.287 | 0.308 | 0.623 |
| | | NMI 0.724 | 0.878 | 0.443 | 0.518 | 0.537 | 0.618 |

FIGURE 3. SK-means clustering comparison on NMI: with respect to the number K of clusters.

D. EXPERIMENT RESULT
1) CLUSTERING PERFORMANCE COMPARISON WITH A GROUND TRUTH NUMBER OF CLUSTERS
Table 2 illustrates the evaluation of K-Means and SK-Means clustering methods for the six datasets based on each document embedding method. We use the ground truth number of clusters K given in the table 1 for K-means and SK-means clustering methods. The best performances of 20Newsgroups, BBC, WebKB, Reuter-8 and Reuter10k among the five document embedding methods are listed.
in bolded numbers in the table. Our proposed document embedding method (GDEM) outperforms the baseline methods by large margins on 20Newsgroups, BBC, WebKB, and Reuter10k datasets. For Opinosis and Reuter-8 datasets, GDEM performs less than SBERT but much better than the rest of the document embedding methods.

From the tables 1 and 2, we find that SBERT gives good embeddings for the document datasets where their member documents are short documents. A short document consists of a small number of words. The characteristics of Opinosis and Reuter8 datasets from the table 1 show that they have short documents whose average number of words in a document
is small. The average numbers of words in a document for Opinosis and Reuter-8 are 17 and 102, respectively. Our GDEM gives good and effective embeddings for the document datasets with medium- and large-sized documents, shown in the tables 1 and 2.

2) CLUSTERING PERFORMANCE COMPARISON WITH DIFFERENT NUMBER OF CLUSTERS

In this experiment, we vary the number of clusters \( K \) to evaluate the effectiveness of our embedding method (GDEM) compared with DEC, SCDE, JoSE, and SBERT in terms of \( K \)-means and \( K \)-means clustering algorithms. Note that we do not recompute the document embeddings for SCDE and DEC methods as \( K \) is increased. The document embeddings of SCDE and DEC are computed only once with the fixed value of \( K \) for each dataset where \( K \) is set to the ground truth number of clusters given in table 1.

As \( K \) increases, figures 2, 3, 4, and 5 show that \( K \)-means and \( K \)-means clustering methods using GDEM gives the much better ARI and NMI than using DEC, SCDE, JoSE, and SBERT for 20Newsgroups, BBC, Reuter10k, and WebKB. SBERT performs better than the GDEM, DEC, SCDE, and JoSE for Opinosis and Reuter-8 datasets having short documents. Interestingly, the document embedding methods that depend on \( K \), such as SCDE and DEC, perform worse than GDEM, JoSE, and SBERT as \( K \) changes. Our GDEM also demonstrates a very good clustering performance for the document datasets with medium- and large-sized documents as \( K \) changes.

IV. CONCLUSION

We propose an effective document embedding method, GDEM, for clustering using a GAE. We first proposed an effective graph construction algorithm to use the GAE. Our graph construction algorithm creates a sparse undirected graph from a set of documents, which effectively captures the high similarities between documents by its weighted edges. The node embedding vectors computed by GDEM from this graph are used as the document embedding vectors. Unlike most existing DNN-based document embedding methods, the document embedding vectors obtained by GDEM are generated independently of the number of document clusters. We performed extensive experimental analyses of GDEM over various real document datasets compared with the existing DNN-based document embedding methods. We demonstrated that GDEM affords excellent and effective document embeddings for the datasets with medium- and large-sized documents.

REFERENCES

[1] Q. Lu, J. G. Conrad, K. Al-Kofahi, and W. Keenan, “Legal document clustering with built-in topic segmentation,” in Proc. 20th ACM Int. Conf. Inf. Knowl. Manage., New York, NY, USA, 2011, pp. 383–392.

[2] C. C. Aggarwal, “Mining text data,” in Data Mining, Berlin, Germany: Springer, 2015, pp. 429–455.