



# Clustering Massive-Categories and Complex Documents via Graph Convolutional Network

Qingchao Zhao<sup>1(✉)</sup>, Jing Yang<sup>1(✉)</sup>, Zhengkui Wang<sup>2(✉)</sup>, Yan Chu<sup>1(✉)</sup>,  
Wen Shan<sup>3(✉)</sup>, and Isfaque Al Kaderi Tuhin<sup>2(✉)</sup>

<sup>1</sup> Harbin Engineering University, Harbin, China  
{zhaoqc418,yangjing,chuyan}@hrbeu.edu.cn

<sup>2</sup> Singapore Institute of Technology, Singapore, Singapore  
{zhengkui.wang,tuhin.kaderi}@singaporetech.edu.sg

<sup>3</sup> Singapore University of Social Sciences, Singapore, Singapore  
viviensw@suss.edu.sg

**Abstract.** In recent years, a significant amount of text data are being generated on the Internet and in digital applications. Clustering the unlabeled documents becomes an essential task in many areas such as automated document management and information retrieval. A typical approach of document clustering consists of two major steps, where step one extracts proper features to model documents for clustering and step two applies the clustering methods to categorize the documents. Recent research document clustering algorithms are mostly focusing on step one to finding high-quality embedding or vector representation, after which adopting traditional clustering methods for the second step. Or infer the document representation based on the predetermined  $k$  clusters. However, the traditional clustering methods are designed with simplistic assumption of the data distribution that fails to cope with the documents with complex distribution and a small number of clusters i.e., less than 50. In addition to this, the previous need a predetermined  $k$ . In this paper, we introduce Graph Convolutional Network into the document clustering (instead of using the traditional clustering methods) and propose a supervised GCN-based document clustering algorithm, DC-GCN which is able to handle documents in noisy, huge and complex distribution by a learnable similarity estimator. Our proposed algorithm first adopts a GCN-based confidence estimator to learn the document position in a cluster via the affinity graph, and then adopts a GCN-based similarity estimator to learn the document similarity by constructing the doc-word graphs integrating the local neighbor documents and its keywords. Based on the confidence and similarity, the document clusters are finally formed. Our experimental evaluations show that DC-GCN achieves 21.88%, 17.35% and 15.58% performance improvement on  $F_p$  over the best baseline algorithms in three different datasets.

**Keywords:** Graph Convolutional Network · Document clustering · Supervised clustering · Massive-categories · Supervised learning

# 1 Introduction

Document clustering or text clustering is an important task in data mining. It aims to partition a pool of text documents into distinctive clusters based on the content similarity such that the documents in a cluster contain similar property in comparison to documents in other clusters. It has been extensively used in many applications such as topic tracking and detection, information retrieval, public opinion analysis and monitoring, and news recommend system, etc. [1].

Traditional document clustering methods are generally conducted in two steps. The first step is the vector representation of document features, which represents the high-dimensional, variable-length documents with low-dimensional fixed-length vectors. Two commonly used feature extraction approaches are proposed either based on topic models e.g. Latent dirichlet allocation (LDA) or the embedding e.g. Doc2vec, FastText [10]. Recently, there are also some clustering methods that extract document features based on the clustering task e.g. , Graph Theory [2], CNN [15], autoencoder [13,19], contractive learning [8,9]. The second step of the document clustering method aims to cluster all the documents based on the extracted features. In this stage, traditional clustering are normally adopted to cluster the documents, such as K-means, DBSCAN, Spectral, Single-Pass, etc.

The existing document clustering approaches focused more on how to get high-quality embedding in the first step. However, in the second step, they adopt the traditional clustering methods that usually result in unsatisfactory performance for the documents with complex distribution and a large number of categories, as they are designed based on simplistic assumption of data distribution with a small number of categories [2,8,9,13,15,19]. For example, K-means-based is only suitable for data distribution around the cluster center. DBSCAN is designed for clusters with uniform density. Spectral is suitable for clusters with similar cluster sizes. The Single-Pass is sensitive to the input order of clustered instances [4,16]. Moreover, most clustering methods require to know the predetermined number of clusters  $k$  in order to make accurate clusters [8,15]. Existing approaches work well when the number of clusters is small and provided, and the documents are in a particular distribution. However, there are many application scenarios which contain massive categories of documents with complex data distribution. For example, personalized recommendations for decentralized we-media content need massive categories of text which can not be just a few categories. Meanwhile, the different sizes of clusters and the richness of the contents lead to complex data distributions. The data in these domains have different characteristics compared with the datasets that have been studied before. First, the data is highly noised with a complex distribution such as a lot of non-convex clusters. Second, the number of clusters is large, which is almost impossible to pre-determine. These characteristics have made existing document clustering algorithms ineffective and inefficient unfortunately. This calls for a new approach to tackle the challenges.

To tackle the issues of massive document clustering with complex distribution in the second step of the document clustering, we convert the clustering problem

into a structural pattern learning problem using Graph Convolutional Network (GCN). GCN has been proven effective in learning the patterns in the affinity graphs [14, 17, 18]. In this paper, we model all the documents as an affinity graph. Based on this, we propose a GCN-based clustering algorithm, DC-GCN which enables effective clustering for complex documents with massive categories. We define confidence based on the characteristic of clusters in the affinity graph. The high confidence of a document node defines the distance between the document to the cluster center where more neighbors have the same label. DC-GCN consists of two important steps. In the first step, we learn the confidence of each document node through its context structure using a GCN-based confidence estimator. In the second step, we learn the similarity of two documents via a GCN-based similarity estimator based on doc-word graphs including both documents and keywords relationship information. Based on predicted confidence and similarity, all the documents can be easily grouped into clusters.

Our key contributions of this paper are three-fold. (1) We make the first attempt to introduce Graph Convolutional Network into the document clustering problem. (2) We propose an innovative supervised GCN-based document clustering algorithm, DC-GCN which enables effective and efficient clustering for massive-category documents with complex distribution. DC-GCN does not require the data with a particular distribution and pre-determined cluster number. It is an intelligent learnable model integrating both documents and its keywords information to cluster massive-categories documents. (3) We provide experimental evaluations based on real datasets with various baseline clustering methods. The results confirm the effectiveness of DC-GCN.

The rest of the paper is organized as follows. Section 2 introduces the related works. In Sect. 3, we present our proposed solution. Section 4 and Sect. 5 provide the experimental evaluations and conclusion respectively.

## 2 Related Works

**Document Features Extraction.** Much research effort has been devoted to identify the best document feature representations and extractions as the first step of the document clustering algorithm. The simplest method is to use document word frequency to filter out irrelevant features. For example, the main idea of Term Frequency–Inverse Document Frequency (TF-IDF). The purpose of Latent Semantic Analysis (LSA) is to discover hidden semantic dimensions—namely “topic” or “Concept” from the document by singular value decomposition (SVD), then pLSA and Latent Dirichlet Allocation (LDA) were developed Later. There are also neural network-based methods. For example, Doc2vec is an unsupervised learning algorithm based on Word2vec. The algorithm predicts a vector based on the context of each word to represent different documents. The structure of the model potentially overcomes the lack of semantics of the word bag model Shortcomings like LDA. Recently, Xu, Jiaming proposed STC2 which uses CNN to fit the text and use the unsupervised learning method to get the text label. After fitting, K-means is used to cluster the hidden layer variables,

and finally the result is obtained [15]. Autoencoder-based methods combine the loss of K-means essentially. Given that there are  $k$  clusters, Dejiao Zhang proposed a MIXAE architecture. This model optimizes two parts at the same time: a set of autoencoders where each learns the distribution similar objects; a mixture assignment neural network, which inputs the concatenated latent vectors from the autoencoders set and infers the distribution over clusters [21]. By using the sample augmentations, Bassoma Diallo propose a deep embedding clustering framework based on contractive autoencoder (need a  $k$ ) with Frobenius norm [8]. These methods still needs a predetermined cluster number  $k$  and suit for less-categories data (usually less than 20). Different to these works that aims to tackle the issues in document extraction, our work is orthogonal to them by investigating how to improve the second step of clustering performance without a predetermined  $k$ .

**Clustering Methods.** After the document feature extraction, the document clustering algorithm adopts clustering method to categorize the documents. The first kind is based on agglomerative methods such as Hierarchical Agglomerative Clustering (HAC). HAC has been studied extensively in the clustering literature for document data, and FastHAC is proposed to reduce the calculation [7]. K-medoid and K-means are two classic distance-based partitioning algorithms. There are also some based on the density, such as DBcsan, MeanShift, Density Peak clustering [5, 6, 11]. Spectral clustering is a Graph-based method. Spectral clustering can also be extended to social networks or Web-based networks [12]. However, all these commonly used clustering algorithms are based on simplistic assumption on the data distribution that fails to cope with the complex distribution or required canedetermine datasets cluster number. Differently, DC-GCN is able to handle any complex distribution with unknown cluster number enabled by its learnable model.

## 3 Problem Formulation and Our Solution

### 3.1 Framework Overview

Our proposed GCN-based clustering algorithm, DC-GCN consists of the two similar steps (feature extraction step followed by clustering step) as other document clustering algorithm. In DC-GCN, we first adopt the most popular vector representation, Doc2vec to extract the features for each document. As how to extract features is not our focus, we will omit the detail here. Note that DC-GCN is a general framework which can incorporate any feature representation.

After extracting the document features, in the next step, we construct the affinity graph by calculating the affinity (cosine similarity) between document features. The affinity graph  $G(V, E)$  consists of all the documents as the nodes in  $V$  and the edges in  $E$  that are constructed between any vertex with its  $k$  nearest neighbors. Figure 1 provides the whole framework overview. Intuitively, our algorithm tackles the clustering problem by predicting the confidence of the nodes and the similarity of the two nodes. We first adopt GCN to develop a

confidence estimator to predict the confidence of each document indicating the position of the document in the cluster. A document with high confidence is close to the cluster center. Otherwise, it is far away from the cluster center. We further propose a similarity estimator to predict the similarity of two documents, where similarity indicates the probability for two documents belonging to the same cluster. Consider that the keywords inside documents provide valuable evidence of the similarity of two documents. We further add the keywords into the learning graph to make an accurate document similarity prediction. After the obtaining the confidence and similarity, we find a path from each node to the cluster center and get final clusters easily.

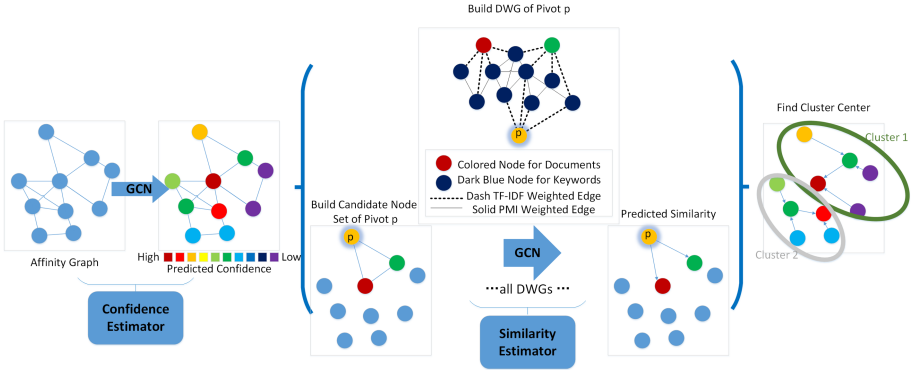


Fig. 1. Overview of DC-GCN framework. (Color figure online)

### 3.2 Confidence Estimator

According to the observation, when a document is close to the cluster center, its neighbor documents are usually more similar and in the same cluster. For a document at the margin of a cluster, there are usually other labeled documents in its neighbors. We define the confidence of a document as follows.

*Definition: Confidence.* The confidence  $C_i$  of document  $i$  is defined as the position of  $i$  in its cluster and formally calculated by:

$$c_i = \frac{1}{|N_i|} \sum_{v_j \in N_i} \begin{cases} a_{ij} & l_i = l_j \\ -a_{ij} & l_i \neq l_j \end{cases} \quad (1)$$

where  $N_i$  is the neighborhood of document  $i$ ,  $a_{ij}$  is the affinity value (cosine similarity) between two documents  $i$  and  $j$ .  $l_i$  is the ground-truth label of document  $i$ . A higher the value of  $c$  indicates that the node is close to the cluster center. To calculate the confidence of each document, we first build an affinity graph. For each document, we select its  $k$  nearest documents based on their

extracted features and build the edge among them. This affinity graph with the original feature is the input of confidence estimator GCN model.

**GCN-Based Confidence Learning Model.** In order to learn the structural patterns of nodes with similar confidence, the confidence estimator is empowered by a GCN model. The input of the GCN model is the adjacency matrix of the affinity graph and the node feature matrix, while the output is the confidence of each node. The formal representation of each layer of the model is:

$$F_{l+1} = \text{RELU}(g(\tilde{D}^{-1}(A + I), F_l)W_l) \quad (2)$$

where  $\tilde{D}_{ii} = \sum_j (A + I)_j$  is the diagonal matrix,  $A$  is the initial adjacency matrix, and  $I$  is the identity matrix.  $W_l$  is a trainable parameter matrix, and RELU is used as the nonlinear activation. We define  $g(\cdot, \cdot)$  as the concatenation of the embedding after neighborhood aggregation and the input embedding, which is formalized as  $g(\tilde{D}^{-1}(A + I), F_l) = [(F_l)^\top, (\tilde{D}^{-1}(A + I)F_l)^\top]^\top$ . The model has four layers. The last layer has to be embedded through a fully connected layer.

**GCN Model Training.** For each node in node-set  $N$ , the loss function  $L$  is the mean square error (MSE) between the predicted confidence  $c$  and the ground-truth confidence  $c'$ . The output of the model is to get the predicted confidence, which is used in the clustering forming step to add edges between documents.

$$\text{Loss} = \frac{1}{N} \sum_{i=1}^N |c_i - c'_i|^2 \quad (3)$$

### 3.3 Similarity Estimator

The aim of the similarity estimator is to learn the similarity of the documents, which is defined as the probability (a value between 0 and 1) of them with the same label. In ground truth, if two documents are with two different labels/clusters, the similarity is 0. If they are with same labels, it is 1.

To obtain the accurate similarity between documents, both the local context and the keywords information play an important role in determining the similarity. The local context of a document refers to the nearest neighbor documents that are close to each other. The keywords appeared in each document are essential information for us to determine the similarity. Therefore, in our similarity estimator, our GCN-based model is proposed to learn the similarity based on heterogeneous learning graphs, namely doc-word graphs that include both the documents and keywords. Our heterogeneous learning doc-word graphs are constructed via below three steps.

*Step 1: Node candidates of doc-word graphs.* For a given document  $p$ , we form a pivoted learning doc-word graph  $DWG_p = G(V_p, E_p)$  of it, which consists of  $p$ 's  $n$  nearest neighbor documents and the keywords of each document in  $V_p$ , and the edge set  $E_p$ . The nearest neighbor documents are those needed to measure the similarity. And, we unfold the document by capturing its keywords

to enrich its semantic meaning of the doc-word graph. These keywords provide detail measures to determine the similarity of the documents instead of using only the Doc2vec information.

*Step 2: Build the input adj matrix.* Since  $DWG_p$  is a heterogeneous graph, we design two types of edges: the edge between the document and the keyword (doc-word edge) and the edge between words (word-word edge). It is worth noting that they are different from the affinity graph in confidence estimator, the doc-word graph does not have the doc-doc edge. For doc-word edge, we assign an edge weight with TF-IDF. For the word-word edge, the point-wise mutual information (PMI) calculates the edge weight between words like TextGCN [20]. Finally, based on the  $E_p$  and the edge weight in the sub-graph. The formal description of the element  $A_{ij}$  of the adjacency matrix  $A$  is as follows:

$$A_{ij} = \begin{cases} PMI(i, j) & i, j \text{ are words, } PMI(i, j) > 0 \\ TF - IDF_{ij} & i \text{ is document, } j \text{ is word} \\ 1 & i = j \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

*Step 3: Build the feature matrix.* We design a pseudo-one-hot encoding to build the nodes' feature which is more efficient than the one-hot encoding. First, we collect all the keywords, and set the feature of the keywords into one-hot vectors like an identity matrix  $I$  whose dimension is equal to the number of keywords. The document node feature is a normalized vocabulary vector. For example, the vectors of the three keywords in corpus  $w_1, w_2, w_3$  are  $(1,0,0), (0,1,0), (0,0,1)$ . The document  $d_1$  contains the words  $w_1$  and  $w_3$ , and the vocabulary vector after the normalization of the document  $d_1$  is  $(0.5, 0, 0.5)$ .

**GCN-Based Similarity Learning Model.** Based on the constructed doc-word graphs, we adopt a two-layer GCN model to learn the document classification task. The formal expression is provided as follows.

$$Z = \text{softmax}(\tilde{A}\text{ReLU}(\tilde{A}XW_0)W_1) \quad (5)$$

Where  $X$  is pseudo-one-hot encoding of the documents and words,  $\tilde{A} = D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ .  $W_0$  and  $W_1$  are the trainable parameter matrix and  $\text{softmax}(x_i) = \frac{1}{Z}\exp(x_i)$ ,  $Z = \sum_i \exp(x_i)$ .

**Training of Similarity Estimator.** For a pivot node  $i$ , if a neighbor  $j$  shares the same label with  $i$ , the label is set to 1, otherwise it is 0.

$$r_{ij} = \begin{cases} 1, l_i = l_j \\ 0, l_i \neq l_j \end{cases}, v_j \in G_i \quad (6)$$

Where  $G_i$  is the sub-graph of pivot  $i$ ,  $l_i$  is the label of the node  $v_i$ . The model predicts the similarity that reflects whether two nodes have the same label. The loss function is MSE.

$$L_E(G_i) = \sum_{v_i \in G_i} |r_{ij} - r'_{ij}|^2 \quad (7)$$

### 3.4 Form Clusters via Confidence and Similarity

Recall that the document confidence reflects its position in a cluster and the similarity of two documents indicates the probability of them in one cluster. For each node, we locate all its neighbors with higher confidence and add an edge with the most similar neighbor. In this way, each node could locate another similar node which has a higher confidence with the position closer to the cluster center. By so doing, we can find a path from each document to the cluster center and form the clusters easily.

### 3.5 Complexity Analysis and Discussion

The time complexity of the whole algorithm consists of three parts: confidence estimator, similarity estimator, and clustering. In confidence estimator, building affinity graph costs  $O(n \log n)$ . The graph convolution can be efficiently implemented as the sparse-dense matrix multiplication, a complexity  $O(|\epsilon|)$ . In similarity estimator, the cost of the preparation of similarity estimator is  $O(n' m \log m)$ , where  $m$  is the number of the candidate documents and keywords which is far less than the number of keywords in the corpus, and  $n'$  is the number of pivots with high confidence we choose. With the affinity graph in confidence estimator, the last step of forming clusters is  $O(n \cdot k)$ , where  $k$  is the number of neighbor documents which is far less than  $n$ .

## 4 Experiments and Performance Analysis

**Dataset.** Most of the available document datasets are all designed for the classification task with a small number of (i.e., below 20) categories, which can not represent the application with massive categories. To evaluate the capability of DC-GCN handling the massive categories, we use the crawled Wikipedia dataset which includes more than 4000 categories. To verify scalability of method, we use three different scale datasets as shown in Table 1 and there is no overlap between the training and test datasets.

**Method Comparison and Metrics.** Recall that the most recent document clustering algorithms all aim to improve the quality of the document feature extraction following by using the classic clustering methods [2, 8, 9, 13, 15, 19]. DC-GCN is orthogonal to recent document clustering algorithm (focus on feature extraction with  $K$ ) and aims to change the second step on the clustering performance. We compare the proposed method with a series of clustering baselines, which includes the classic clustering methods widely used in recent document clustering algorithms i.e. K-means, mini-batch K-means (MK-means), DBSCAN, HAC, MeanShift, Spectral [5, 6, 11]) and the streaming document clustering algorithm Single-Pass (S-pass). Additionally, to study the performance improvement of the proposed GCN-based similarity estimator over the existing similarity calculation approach, we develop another method (CE) that only includes the proposed GCN-based confidence estimator and Euclidean distance evaluation to



**Table 1.** Wikipedia datasets introduction

	#Clusters	#Instances	Domain
Training dataset	1851	10805	Society, Geography, Universe, Politics, Concepts, Events, Government, Economy, Education, Energy, Engineering, Entities, Ethics, Business
Test dataset1	2660	15623	Technology, Health, Culture, Nature, Sports, Science, Entertainment, Religion, Philosophy, Belief, Mathematics, World
Test dataset2	2228	13094	Technology, Health, Culture, Nature, Sports, Science Entertainment
Test dataset3	1688	9921	Technology, Health, Culture, Nature, Sports

form the clusters. For fair comparison, we use unsupervised Doc2vec extracted features that do not require a predetermined  $K$  instead of previous document-clustering-based feature extraction.

We adopt three most popular evaluation metrics to evaluate the effect of clustering, namely Normalized Mutual Information (NMI), Pair-wise F-score and BCubed F-score [3]. Meanwhile, we also compare the inferring running time of different methods after the one-off training time.

**Clustering Effectiveness and Runtime Analysis.** For all baseline methods, we report the best results by tuning the hyper-parameter. Table 2 provides the detail results in three datasets. From the results, we have the following observations: (1) Among all the baseline algorithms, K-means performs nearly the best with the longest inferring time when  $K$  is set as the ground-truth number. However, K-means is highly depending on the predefined number of clusters  $k$ . The performance will highly decrease if the  $k$  is set as the wrong number. We also can infer that all K-means-based methods converge very slowly when the number of categories increases. (2) The sampling method of mini-batch K-means (MK-means) can speed up calculations by losing part of the accuracy. (3) The effect of spectral clustering is second to K-means in all baselines, and the computational efficiency is much higher than K-means. But, solving features Value decomposition leads to a large number of calculations and memory requirements, thereby limiting the application of spectral clustering. (4) DBSCAN is almost the most efficient among all algorithms when given the similarity matrix, but it assumes that the density of each cluster is similar. Therefore, when the

cluster distribution is complex, DBSCAN loses efficiency. (5) Although HAC does not require a pre-determined number of clusters, the iterative merging process involves a lot of computational budgets and outliers can have a great impact. (6) The overall result of MeanShift is worse than K-means and spectral clustering, but it has a slow convergence speed and only faster than K-means in all baselines. (7) The effect of single-pass is also good among the baselines, but single-pass is sensitive to the input order of documents. The outliers also have a great influence on the results. (8) Confidence Estimator is better than half of the baseline results. Through more than one thousand classes of training, the results of two thousand clusters can be predicted, which proves its effectiveness and scalable in capturing important structural characteristics of nodes. (9) DC-GCN outperforms other algorithms in all the different datasets and metrics with comparable inferring time. The final column of Table 2 provides the DC-GCN’s percentage of performance improvement over the best baseline (underlined) for that metric. This confirms the effectiveness and efficiency of the proposed approach empowered by the learning capability of confidence and similarity estimators.

**Table 2.** Performance comparison results

	Methods	K-means	MK-means	Spectral	HAC	DBSCAN	MeanShift	S-pass	CE	DC-GCN	$\Delta(\%)$
Wiki test set1	$F_P$	<u>0.2248</u>	0.1359	0.1945	0.0008	0.0009	0.0140	0.0430	0.1496	<b>0.2740</b>	21.88
	$F_B$	0.4172	0.3894	0.3589	0.3619	<u>0.4437</u>	0.3922	0.3766	0.3539	<b>0.4675</b>	5.36
	NMI	<u>0.8461</u>	0.8196	0.8325	0.4112	0.5454	0.8056	0.7715	0.8120	<b>0.8714</b>	2.99
	Clusters	2661	2633	2658	3250	4639	4533	2935	1881	4677	–
	Time	2041 s	123 s	169 s	25 s	2 s	839 s	93 s	2 s	423 s	–
Wiki test set2	$F_P$	<u>0.2432</u>	0.1660	0.2024	0.0009	0.0012	0.0188	0.0437	0.1424	<b>0.2854</b>	17.35
	$F_B$	0.4300	0.4061	0.3701	0.3836	<u>0.4600</u>	0.4028	0.3547	0.3533	<b>0.4792</b>	4.17
	NMI	<u>0.8481</u>	0.8254	0.8323	0.4430	0.5813	0.8126	0.7616	0.8057	<b>0.8713</b>	2.73
	Clusters	2228	2220	2225	2908	4108	3979	2642	1446	3813	–
	Time	991 s	99 s	237 s	29 s	2 s	724 s	81 s	2 s	173 s	–
Wiki test set3	$F_P$	<u>0.2688</u>	0.1862	0.2231	0.0009	0.0017	0.0274	0.0725	0.2260	<b>0.3107</b>	15.58
	$F_B$	0.4538	0.4310	0.3865	0.3836	<u>0.4736</u>	0.4348	0.4138	0.4209	<b>0.4941</b>	4.32
	NMI	<u>0.8504</u>	0.8283	0.8322	0.4430	0.6004	0.8155	0.7894	0.8340	<b>0.8710</b>	2.42
	Clusters	1688	1675	1686	2908	3153	3063	2039	1523	2787	–
	Time	812 s	59 s	118 s	13 s	3 s	402 s	44 s	2 s	78 s	–

**Candidate Document Selection in Doc-Word Graph.** In this experiment, we study the impact of the different candidate documents selection schemes in DC-GCN. We design two different schemes to select the candidate documents adding to the doc-word graph of a given pivot document  $p$ . The first scheme (KNN-DWG) adds all the  $k$  nearest neighbor documents of the pivot  $p$  to its DWG, while the second scheme (FC-DWG) only adds the nearest neighbor documents whose confidence is bigger than  $p$ . The second scheme filters out those documents in the nearest neighbor with low confidence value, which results in less nodes in the DWG and improves the calculation efficiency. Table 3 shows the comparison results of these two methods. As expected, FC-DWG is faster than KNN-DWG, as FC-DWG generates smaller size of Doc-word graphs. Interestingly, we also find that FC-DWG is able to generate comparable performance

**Table 3.** Results of different candidate selection approaches

Methods	Wiki test set1		Wiki test set2		Wiki test set3	
	KNN-DWG	FC-DWG	KNN-DWG	FC-DWG	KNN-DWG	FC-DWG
$F_P$	0.2615	<b>0.2740</b>	0.2746	<b>0.2854</b>	0.2974	<b>0.3107</b>
$F_B$	0.4579	<b>0.4675</b>	0.4708	<b>0.4792</b>	0.4845	<b>0.4941</b>
NMI	0.8710	<b>0.8714</b>	0.8706	<b>0.8713</b>	0.8700	<b>0.8710</b>
Clusters	4897	4677	3959	3813	2901	2787
Time	587 s	423 s	190 s	173 s	97 s	78 s

with KNN-DWG, which indicates that remaining those high-confidence documents in the doc-word graphs is sufficient for the learning problem.

**Comparison Between Static and Dynamic Affinity Graphs.** In DC-GCN, there are two possible approaches based on static affinity graph or dynamic affinity graph in calculating the confidence and similarity. The static affinity graph uses original affinity graph in each layer of the GCN, and finding the  $k$  nearest neighbor documents for doc-word graph generation is also based on original affinity graph and Doc2vec features. Differently, the dynamic approach rebuilds the affinity graph after each graph covolutional layer in GCN, and the finding KNN documents is also based on the updated affinity graph and updated features. This experiment aims to study the performance difference between the static approach in calculating the confidence (CE(s)) and similarity (SE(s)), and the dynamic approach (CE(d) and SE(d)). As shown in Table 4, in the confidence estimator, two kinds of methods surpass each other in different testing datasets. In the similarity estimator, using the original feature to locate the candidate documents can get a better result. Moreover, on a large-scale graph with millions of nodes, rebuilding the affinity graph by the hidden feature results in an excessively high computational budget. These observations indicate that the dynamic approach is not superior compared to the static one.

**Table 4.** Results on static and dynamic affinity graphs

Methods	Wiki test set1				Wiki test set2				Wiki test set3			
	CE(s)	CE(d)	SE(s)	SE(d)	CE(s)	CE(d)	SE(s)	SE(d)	CE(s)	CE(d)	SE(s)	SE(d)
$F_P$	0.1496	0.1855	0.2740	0.2555	0.1424	0.1907	0.2854	0.2655	0.2260	0.1925	0.3107	0.3055
$F_B$	0.3539	0.3944	0.4675	0.4544	0.3533	0.3987	0.4792	0.4644	0.4209	0.3872	0.4941	0.4844
NMI	0.8120	0.8308	0.8714	0.8608	0.8057	0.8284	0.8713	0.8608	0.8340	0.8186	0.8710	0.8608
Clusters	1881	2501	4677	4897	1446	1994	3813	3841	1523	1165	2787	2501
Time	2 s	2 s	423 s	514 s	2 s	2 s	173 s	203 s	2 s	2 s	78 s	98 s

## 5 Conclusion

This paper made the first attempt to introduce Graph Convolutional Network (GCN) into the document clustering task. We proposed a GCN-based document clustering algorithm, DC-GCN that provides effective clustering for massive-category documents with complex distribution. DC-GCN transfers the clustering task into two major learning components (learning the document confidence and similarity) by powerful GCN. It integrates both the document and its keywords into the learning framework. Our experimental results indicated that our proposed method outperforms the existing document clustering algorithms, w.r.t. the accuracy and efficiency. Meanwhile, DC-GCN does not require any pre-determined cluster numbers and copes well with the large-scale and high-noise document clustering. We expect DC-GCN can be applied in wider applications with complex data distributions.

**Acknowledgement.** This research was supported by Singapore MOE TIF grant (MOE2017-TIF-1-G018), the National Natural Science Foundation of China (61672179, 61370083) and China Postdoctoral Science Foundation (2019M651262).

## References

1. Aggarwal, C.C., Zhai, C.: A survey of text clustering algorithms. In: *Mining Text Data*, pp. 77–128. Springer, Boston (2012). [https://doi.org/10.1007/978-1-4614-3223-4\\_4](https://doi.org/10.1007/978-1-4614-3223-4_4)
2. Ali, I., Melton, A.: Semantic-based text document clustering using cognitive semantic learning and graph theory. In: *ICSC*, pp. 243–247. IEEE (2018)
3. Amigó, E., Gonzalo, J., Artiles, J., Verdejo, F.: A comparison of extrinsic clustering evaluation metrics based on formal constraints. *Inf. Retrieval* **12**(4), 461–486 (2009)
4. Berkhin, P.: A survey of clustering data mining techniques. In: *Grouping Multi-dimensional Data*, pp. 25–71. Springer, Berlin (2006). [https://doi.org/10.1007/3-540-28349-8\\_2](https://doi.org/10.1007/3-540-28349-8_2)
5. Bohm, C., Railing, K., Kriegel, H.P., Kroger, P.: Density connected clustering with local subspace preferences. In: *ICDM 2004*, pp. 27–34. IEEE (2004)
6. Cheng, Y.: Mean shift, mode seeking, and clustering. *IEEE Trans. Pattern Anal. Mach. Intell.* **17**(8), 790–799 (1995)
7. Dash, M., Liu, H., Scheuermann, P., Tan, K.L.: Fast hierarchical clustering and its validation. *Data Knowl. Eng.* **44**(1), 109–138 (2003)
8. Diallo, B., Hu, J., Li, T., et al.: Deep embedding clustering based on contractive autoencoder. *Neurocomputing* **433**, 96–107 (2021)
9. Hu, W., Miyato, T., Tokui, S., et al.: Learning discrete representations via information maximizing self-augmented training. In: *ICML*, pp. 1558–1567 (2017)
10. Joulin, A., Grave, E., Bojanowski, P., et al.: Fasttext. zip: compressing text classification models. [arXiv:1612.03651](https://arxiv.org/abs/1612.03651) (2016)
11. Rodriguez, A., Laio, A.: Clustering by fast search and find of density peaks. *Science* **344**(6191), 1492–1496 (2014)
12. Von Luxburg, U.: A tutorial on spectral clustering. *Stat. Comput.* **17**(4), 395–416 (2007)

13. Wang, X., Peng, D., Hu, P., et al.: Adversarial correlated autoencoder for unsupervised multi-view representation learning. *KBS* **168**, 109–120 (2019)
14. Wang, Z., Zheng, L., Li, Y., Wang, S.: Linkage based face clustering via graph convolution network. In: *CVPR*, pp. 1117–1125 (2019)
15. Xu, J., Xu, B., Wang, P., et al.: Self-taught convolutional neural networks for short text clustering. *Neural Netw.* **88**, 22–31 (2017)
16. Xu, R., Wunsch, D.: Survey of clustering algorithms. *IEEE Trans. Neural Netw.* **16**(3), 645–678 (2005)
17. Yang, L., Chen, D., Zhan, X., et al.: Learning to cluster faces via confidence and connectivity estimation. In: *CVPR*, pp. 13369–13378 (2020)
18. Yang, L., Zhan, X., Chen, D., et al.: Learning to cluster faces on an affinity graph. In: *CVPR*, pp. 2298–2306 (2019)
19. Yang, L., Cheung, N.M., et al.: Deep clustering by gaussian mixture variational autoencoders with graph embedding. In: *ICCV*, pp. 6440–6449 (2019)
20. Yao, L., Mao, C., Luo, Y.: Graph convolutional networks for text classification. *AAAI* **33**, 7370–7377 (2019)
21. Zhang, D., Sun, Y., Eriksson, B., Balzano, L.: Deep unsupervised clustering using mixture of autoencoders. arXiv preprint [arXiv:1712.07788](https://arxiv.org/abs/1712.07788) (2017)