[Re] Autoencoding Keyword Correlation Graph for Document Clustering

Alireza Torabian
Department of Electrical Engineering and Computer Science
York University
Toronto, ON M3J 1P3
torabian@yorku.ca

Reproducibility Summary

Scope of Reproducibility

In the paper "Autoencoding Keyword Correlation Graph for Document Clustering"[1], various options are used and compared in each part of their novel approach for document clustering. Their approach consists of three main steps that are Keyword Correlation Graph Construction, Graph Autoencoding, and Clustering. The best result they have achieved is using Sentence Similarity for the edges of their graph, SBERT for its node embeddings, and Multi-Task graph autoencoder (MTGAE)[2] for graph encoding. Within the scope of the ML Reproducibility Challenge, we have tied to reproduce their best results using these properties on 20 Newsgroups and Reuters-21578 datasets.

Methodology

Their approach is implemented from scratch, and some analyses are done after each main step of their approach. We also compared their results with k-means and a deep clustering neural network on document embeddings instead of spectral clustering they have used in the paper. Another evaluation is done based on changing the number of top keywords used to create the Keyword Correlation Graph.

Results

Our results do not support the paper. The results we have reproduced are much lower than the paper's results and have lower quality than the baseline models. In keyword correlation graph construction, many documents are not assigned to this graph when only the top 50 keywords are used as specified in the paper. Even using more keywords, the documents are not well clustered, and outputs show a low-quality approach for document clustering.

What was easy

Implementing and using the models for graph encoding and clustering, which are the second and third steps of the approach in the paper, were easy since they have used well-known or published methods. Only the original published MTGAE required some changes to be usable for this paper, but there was not any challenges.

What was difficult

Constructing the keyword correlation graph was the most challenging part. First of all, many details were not mentioned in the paper while their code was not published, and even some instructions from authors were inconsistent with the paper. In addition, the computational complexity for building the graph is quadratic in the number of sentences in all documents, which is high enough to take several days to be finished.

Communication with original authors

The first author of the paper, Billy Chiu, has been answering my questions about the paper and had quick responses; however, there were some inconsistencies between his instructions and the paper.

Preprint. Under review.

1 Introduction and Paper Summary

In this paper, they aim to cluster documents using both local and global features of sentences. Their novel idea is to use a graph-based representation for clustering that captures both local and global features. A weighted graph named Keywords Correlation Graph (KCG) is created using the top keywords of documents. Each node of this graph has a feature which is the embedding of its related sentences capturing the local features, and edge weights of the graph are the correlation between nodes' related sentences to capture the global features. The encoding of the graph is used for clustering. This approach consists of three main steps:

1.1 Keyword Correlation Graph Construction

To construct the related graph for each document, first, they extract the top-50 keywords of documents using Non-Negative Matrix Factorization (NMF). NMF extracts some topics or components from the documents, and each topic has a list of related keywords with their scores showing how much related they are to that topic. Using these scores, we extract the top keywords of all documents. Each keyword is a node of the Keyword Correlation Graph (KCG). Each sentence of the document will be assigned to its most related keyword based on its TF-IDF. Therefore, each node will have a set of related sentences. These nodes have node features equal to the average of the related sentences' embeddings generated using 'bert-large-nli-stsb-mean-tokens' model from Sentence-BERT. They have compared the results using different sentence embeddings such as BERT, GloVe, and ELMo. To construct the weight of edges, three different options are used:

- **Sentence Similarity:** The similarity between two sentences equals the cosine similarity between the embedding of two sentences. The weight between two nodes is the mean pairwise similarity between their sentences' sets.
- **Sentence Position:** Important keywords usually appear at the beginning of documents. Each keyword has a score showing how close it is to the beginning of its sentences. The position weight is the mean of two related sentence sets.
- **Word Co-occurrence:** Based on the hypothesis that similar contexts have similar keywords, the weight between two nodes equals their co-occurrence words frequencies.

1.2 Graph Autoencoding

Graph Autoencoders (GAEs) are encoder-decoder models aiming to encode the graph's features such that the decoder can reconstruct the graph using them. In the paper, to capture the local information (node features) simultaneously, they have used Multi-Task GAE (MTGAE), which tries to learn the representation which can reconstruct both the graph and node features. The encoder of this graph autoencoder model takes the weights of the connected edges and the feature of one node in each step and transforms it to a space called latent space. The decoder part tries to reconstruct the original input using the information from the latent space. For both edge weights and node features, the mean squared error is used as the model's loss function. They have compared the results of this autoencoder with a variational graph autoencoder and a generic sequence-level autoencoder.

1.3 Clustering

After generating the embeddings of all graph nodes, the embedding of each document is calculated based on the global average pooling of its related nodes to have fixed-length representations for each document. These representations are used in Spectral Clustering to have clusters of the documents. Since some documents do not contain any of the extracted top keywords and are not assigned to the graph, we do not have embedding for them to use in the clustering process. Therefore, all of these documents are considered as a separate cluster.

Their best results are achieved using the Sentence Similarity property for edges, SBERT for sentence embedding, and MTGAE for graph autoencoding. The paper claims that their model with these properties has outperformed the previously existing models with the AMI score of 0.530 and accuracy of 47.4% on 20 Newsgroups and the AMI score of 0.584 and accuracy of 56.3% on Reuters-21578.

2 Scope of reproducibility

Since the paper claims they have outperformed the existing models for document clustering, we have tied to reproduce their best results to verify their claims. Their best results are obtained using the Sentence Similarity property for edges, SBERT for sentence embedding, and MTGAE for graph autoencoding. They have achieved the AMI score of 0.530 and accuracy of 47.4% on 20 Newsgroups and the AMI score of 0.584 and accuracy of 56.3% on Reuters-21578. We have re-implemented their approach based on these properties and evaluated the model on 20 Newsgroups and Reuters-21578 datasets.

3 Methodology

In order to assess the architecture and performance of their approach in addition to verifying their results, we have implemented their approach from scratch and have some evaluations after each main step of their approach. After the KCG construction, we check that what proportion of documents' sentences are ignored and not assigned to the main nodes of KCG. We also evaluated the clustering using larger graphs to cover more documents in the KCG. Before the final training of graph autoencoder, we split the nodes into train and validation sets to evaluate the model and find the best number of epochs. For the clustering part, we used Spectral Clustering to reproduce the paper's results and compared it with K-Means and a deep clustering neural network aiming to improve their approach as our stretch goal.

The deep clustering model we have implemented has several steps. First, we use a 4-layer autoencoder with Relu activation functions to encode the document embeddings into a latent space. Then a k-means model is fitted on the extracted latent features of document embeddings. Afterward, a clustering layer is attached to the encoder model to cluster the latent features of document embeddings. The clustering layer is initialized with the cluster centers of the trained k-means. To improve the clustering assignments and latent features simultaneously, we define a target probability distribution and minimize its KL divergence against the model clustering result in several epochs. We also evaluated the approach using a larger number of keywords/nodes (70 keywords instead of 50 keywords) to include more sentences in the main nodes of the graph.

There are no official or unofficial published codes for this paper, but for the second step of their approach, the best graph autoencoder they have used is MTGAE, whose paper and code are published. Therefore, the published code of MTGAE is used in this project. Other parts of the paper are re-implemented. Since the KCG has a limited number of nodes, training and evaluating the MTGAE do not take much time. The most time-consuming part is constructing the KCG. For this part, we need to calculate the similarity between document sets to obtain the weight of KCG's edges. Since there are many documents in our datasets, it required high computational resources and took several days to be completed. For this reproducibility project, a hardware system with eight vCPUs and 32 GB memory is used, which is provided by Google Cloud.

3.1 Datasets

The datasets used in the paper are 20 Newsgroups and Reuters-21578. We used these two datasets as well. Both of them are open-source text classification datasets and are available on the web. 20 Newsgroups dataset has 20 classes, and Reuters-21578 has ten classes. They have done some preprocessing on these datasets based on a previous paper [3].

3.2 Hyperparameters

Most hyperparameters for graph construction and the graph autoencoder were clearly presented in *section 5 Model Training* in the paper. As the paper mentioned, we left not specified hyperparameters of MTGAE to their default values. Only to determine the best epoch to train MTGAE since it was not mentioned in the paper, we trained and evaluated the model on 150 epochs to find the best epoch number for final training. We chose early stopping with 60 epochs.

3.3 Experimental setup

We used a hardware system with eight vCPUs and 32 GB memory provided by Google Cloud only for the steps that require high computational resources. Since the neural networks we have do not require heavy computations, we did not need GPU resources. All the evaluations and results are available in a notebook[1]. All codes are published at *https://github.com/1997alireza/Autoencoding-Graph-for-Document-Clustering*.

[1] https://github.com/1997alireza/Autoencoding-Graph-for-Document-Clustering/blob/main/evaluation.ipynb

3.4 Computational requirements

The most computation-intensive part of this approach is Keyword Correlation Graph construction. Each node of this graph has a set of sentences, and to obtain the weight between two nodes, we need to calculate the similarity between each instance of related sets. For this part, we have used the provided system from Google Cloud. Using this system, it takes about 400 hours to build four required graphs, including small graphs and big graphs of two mentioned datasets.

4 Results

First, we tried to reproduce the results of the paper using their setup. Then we have changed the clustering method and the number of keywords to evaluate their approach using a different setup.

4.1 Default model

The default setup for document clustering using encoded keyword correlation graph we have used are:

- 50 top keywords to create the nodes of graph
- SBERT for sentence embedding
- Sentence similarity property for edge weights of keyword correlation graph
- Multi-Task graph autoencoder for encoding the keyword correlation graph
- Spectral clustering to cluster the document embeddings

In the paper, they have achieved their best results using this setup. In table 1, we show the results we have got and the results reported in the paper.

		ACC	AMI
Reuters	Paper result	0.563	0.584
	Our result	0.352	0.125
20 Newsgroups	Paper result	0.474	0.530
	Our result	0.108	0.055

Table 1: Comparing results with the paper report

4.2 Different clustering method

We used k-means instead of spectral clustering to see if this approach can produce the results with other clustering methods. This experiment is done using the default graph size, which is 50.

		ACC	AMI
Reuters	Spectral Clustering	0.352	0.125
	K-Means	0.339	0.133
20 Newsgroups	Spectral Clustering	0.108	0.055
	K-Means	0.110	0.060

Table 2: Results using different clustering methods

4.3 Larger graph

Since many documents are not assigned to the graph because they do not contain any keywords, we provided experiments to evaluate the effect of the graph size on the accuracy of document clustering. We did our previous experiments with 70 keywords (named big graph) instead of 50 keywords (named small graph) which is defined in the default setup of the paper. The results of our experiments on big graphs are provided in table 3. Comparing this table with the table 2 shows

the effect of graph size on the quality of results. Using more keywords, more documents are considered in the graph, and the clustering is more accurate in most cases. The ratio of assigned documents to the keyword correlation graph is shown in table 4.

		ACC	AMI
Reuters	Spectral Clustering	0.316	0.139
	K-Means	0.302	0.200
20 Newsgroups	Spectral Clustering	0.112	0.071
	K-Means	0.119	0.075

Table 3: Results using different clustering methods on big graphs

	Small Graph	Big Graph
Reuters	57%	78%
20 Newsgroups	71%	81%

Table 4: The ratio of assigned documents to the keyword correlation graph; small graphs and big graphs are made using 50 and 70 keywords, respectively.

5 Stretch Goals

After they generate document embeddings in their approach, they use spectral clustering to cluster the documents based on their embeddings. As they have mentioned in the paper, they did not use a more advanced clustering method to emphasize the effect of GAE on learning local and global information of documents. To improve their results, we used a neural network-based deep clustering method on the embeddings. First, the embeddings are transformed by training an autoencoder model; then, the encoded embeddings are used in a clustering layer which is initialized with the cluster centers of a train k-means model. On the Reuters dataset, the accuracy is improved a bit, but the AMI score falls to about one-third to one-half of its value with spectral clustering. For the 20 Newsgroups dataset, the results did not change a lot. This experiment shows that the clustering part is working well, and the low-quality results are not related to this part of their approach. We have compared the results of the stretch goal and the default model in table 5. These results are based on the default setup for graph construction using the top 50 keywords of documents.

			ACC	AMI
Reuters	Small graph	Spectral Clustering	0.352	0.125
		Deep Clustering	**0.386**	**0.044**
	Big graph	Spectral Clustering	0.316	0.139
		Deep Clustering	**0.326**	**0.066**
20 Newsgroups	Small graph	Spectral Clustering	0.108	0.055
		Deep Clustering	**0.100**	**0.040**
	Big graph	Spectral Clustering	0.112	0.071
		Deep Clustering	**0.106**	**0.056**

Table 5: Results of the stretch goal on small and big graphs

6 Discussion

The results we have achieved using the best setup in the paper are lower than 62% in terms of accuracy and 21% in terms of AMI. These results do not support the claims in the paper and also show this approach cannot produce high-quality results for document clustering compared with previous successful models. In our experiments, we have evaluated the approach step by step and detected some problems in the middle step of the paper's approach.

Based on our results, changing the clustering method to a more advanced one cannot improve the results; this shows the main problem is not related to the third part when we cluster the document embeddings, and actually, the embeddings generated for documents do not have sufficient quality and cannot present useful information about the documents. The first problem we have seen in this approach is in the first step, constructing the keyword correlation graph. In the paper, they have used 50 keywords which are not sufficient to cover all documents, and many of them are not assigned to the graph. Only 57% of Reuters documents and 71% of the 20 Newsgroups documents are assigned to the graph, and for other documents, we do not have embeddings to be clustered in the final step. We have evaluated the effect of graph size on the results, which shows that graph size is not the only problem and using bigger graphs, we still cannot produce acceptable results. Therefore, we can conclude that the main problem is how the document embeddings are generated in this approach, and we cannot really preserve sufficient information about documents for clustering using the keyword correlation graph.

In our experiments, we did not have enough time to implement and experiment with the other properties they have used for the keyword correlation graph to evaluate this approach using them as well. One problem we have seen in the paper is in graph construction. The paper says that for each document, one graph is created using 50 keywords. It is not possible to have different graphs, and we cannot use several graphs for the graph autoencoder step unless the graphs are combined. Nevertheless, when we asked the author to clear this step, he corrected this part; they have created one graph for all documents.

6.1 What was easy

For keyword correlation graph autoencoding, the first network they have used is MTGAE, an open-source model whose code is available at Github, and it is a well-structured project. Only some changes were required on the output and loss function of the original MTGAE model to be fitted in this paper. But since the code of this part was available and the required hyperparameters are provided in the paper, implementing and connecting this part to the paper was easy. For the clustering part, the required details are provided, and the method they have used is a well-known method that is provided in many Python libraries.

6.2 What was difficult

For keyword correlation graph construction, many details were missing in the paper. It was not clear how they extract the top keywords using NMF, how they assign sentences to the most related keyword, and what we should do with documents that are not assigned to any keywords. The paper has said they have extracted 50 keywords for each document; however, it is not possible to have a different graph for each document. Based on the instructions from the author, 50 keywords are extracted from all documents together. These ambiguities made the implementation of this part harder than it seems. Also, since there are many documents with many sentences in the used datasets, and the weight between two keywords is calculated based on the similarity between the embeddings of their related sentences, it is computation-intensive to build this graph and took many CPU hours to be completed.

6.3 Communication with original authors

Since there were many ambiguities and a lack of details in the paper, especially in the KCG construction part, it was not possible to implement their approach. We communicated with the main author of the paper, Billy Chiu, by email to ask for the details of their implementation. There were some inconsistencies between the author's instructions and the paper. We followed the instructions from the author in such cases. The contradictions we found are:

- The paper said that the top 50 keywords of each document are extracted to create the KCG, which is not possible. Because in such a case, the top keywords of each document can be different, and it is not reasonable to use them as the same features for all documents. However, as the author said, we extracted the top 50 keywords of all documents.
- The loss function they have used to train the graph autoencoder model in the paper is the cross-entropy function, but the author said they have used mean squared error.

References

[1] Billy Chiu, Sunil Kumar Sahu, Derek Thomas, Neha Sengupta, and Mohammady Mahdy. Autoencoding keyword correlation graph for document clustering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3974–3981, Online, July 2020. Association for Computational Linguistics.

[2] Phi Vu Tran. Learning to make predictions on graphs with autoencoders. In *5th IEEE International Conference on Data Science and Advanced Analytics*, 2018.

[3] Jianbo Ye, Yanran Li, Zhaohui Wu, James Z. Wang, Wenjie Li, and Jia Li. Determining gains acquired from word embedding quantitatively using discrete distribution clustering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1847–1856, Vancouver, Canada, July 2017. Association for Computational Linguistics.