Autoencoding Keyword Correlation Graph for Document Clustering

Billy Chiu Sunil Kumar Sahu Derek Thomas Neha Sengupta Mohammady Mahdy
Inception Institute of Artificial Intelligence,
Abu Dhabi, United Arab Emirates
{hon.chiu|sunil.sahu|neha.sengupta}@inceptioniai.org,
derek.thomas@pax-ai.ae, mohammady.mahdy@inceptioniai.org

Abstract

Document clustering requires a deep understanding of the complex structure of long-text; in particular, the intra-sentential (*local*) and inter-sentential features (*global*). Existing representation learning models do not fully capture these features. To address this, we present a novel graph-based representation for document clustering that builds a *graph autoencoder* (GAE) on a *Keyword Correlation Graph*. The graph is constructed with topical keywords as nodes and multiple local and global features as edges. A GAE is employed to aggregate the two sets of features by learning a latent representation which can jointly reconstruct them. Clustering is then performed on the learned representations, using vector dimensions as features for inducing document classes. Extensive experiments on two datasets show that the features learned by our approach can achieve better clustering performance than other existing features, including term frequency-inverse document frequency and average embedding.

1 Introduction

Text classification is a core task in natural language processing (NLP) with a variety of applications, such as news topic labeling and opinion mining. Supervised methods for text classification generally perform better than unsupervised clustering methods, at the cost of heavy annotation efforts. In contrast, unsupervised clustering methods have the advantage in terms of requiring less prior knowledge and can be used to discover new classes when relevant training data is not available.

The performance of text clustering is closely related to the quality of its feature representation. While sentence-level clustering relies primarily on the *local*, intra-sentential features, document-level clustering also needs the *global*, inter-sentential features. Existing representation learning methods that model text as a bag-of-words (e.g., term frequency-inverse document frequency, TFIDF) or as sequences of variable-length units (e.g., Bidirectional Encoder Representations from Transformers, BERT) (Devlin et al., 2019) are ineffective in capturing global features across long sequences – suffering from heavy computational cost as a result of high dimensionality and complex neural network architectures, as reported by Ye et al. (2017) and Jawahar et al. (2019).

Recently, graph neural networks have been used to provide features for NLP applications, including text classification (Yao et al., 2019) and relation extraction (Sahu et al., 2019). By modeling text in a topological structure, these models can encode global information in long-range words. Despite their usefulness, graph models remain underexplored in document clustering.

In this work, we propose a novel graph-based representation for document clustering by utilizing a *graph autoencoder* (GAE) (Kipf and Welling, 2016) on a *Keyword Correlation Graph* (KCG). Our KCG represents a document as a weighted graph of topical keywords. Each graph node is a keyword, and sentences in the document are attached to the nodes they are related to. The edges between nodes indicate their correlation strength, which is determined by comparing their corresponding sets of sentences. The node and edge features in the KCG are encoded using a GAE, and the encoded features are used to infer document classes.

Our contribution is threefold. First, we propose a KCG, which can capture the complex relations among words and sentences in long text. Second, we propose a new graph-based representation for document clustering. To the best of our knowledge, this is the first attempt to use GAEs to jointly learn local and global features for document clustering.

Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 3974–3981

Last, an analysis of the individual model components indicates that our model can effectively encode both sets of features. This distinguishes us from existing sequence-level representations which generally better encode the former than the latter.

2 Related Work

In the literature, three common neural methods, Convolutional neural network (CNN), Recurrent neural network (RNN) and Transformer, have been proposed to model the sequence-level features between words. CNNs have been shown to be more effective in capturing features in short text (e.g. phrases) than in long sequences (Xu et al., 2015). In contrast, RNN is suitable for handling sequential input (Zhou et al., 2019). It aims at modelling the relations between the current word and all the previous ones in the sequence as a whole. Unlike RNN and CNN, which model a text sequence either from left to right or combined left-to-right and right-to-left, Transformer operates on the masked language model that predicts randomly-masked words in consecutive sentence pair. Nonetheless, these approaches only model the context on consecutive words/sentences, neglecting many global features that span across non-consecutive text units in multiple sentences.

Several methods have been proposed to represent documents as graphs. These document graphs can be induced directly from the input document, using its words, sentences, paragraphs or even the document itself as nodes (Defferrard et al., 2016), and establishing edges according to the distributional information such as, word co-occurrence frequencies (Yao et al., 2019; Peng et al., 2018), text similarities (Putra and Tokunaga, 2017) and hyperlinks between documents (Page et al., 1999). Alternatively, document graphs can be constructed indirectly with the use of NLP pipelines and knowledge bases such as WordNet (Miller, 1995) for identifying the entities in the document, as well as their syntactic and semantic relations (Sahu et al., 2019; Li et al., 2019). However, such type of approaches are limited to resource-rich languages.

3 Methodology

We describe our model architecture in Figure 1. It includes three steps. Given a document, the model first constructs a KCG with keywords as nodes and edges correspond to their local and global features. Next, it uses a GAE to encode the two feature sets

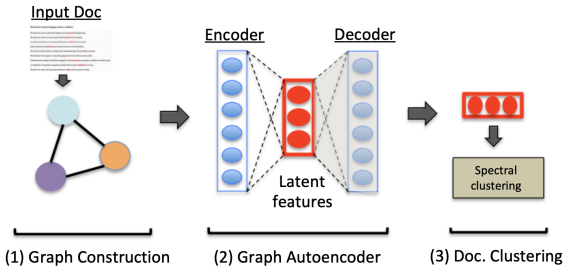

Figure 1: Proposed model architecture.

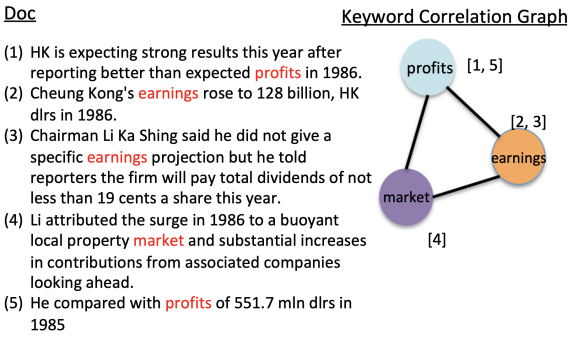

Figure 2: An example showcases a document, its keywords (red) and KCG representation. Example adapted from the Reuters dataset (Lewis et al., 2004)

by jointly reconstructing them. Finally, clustering is performed on the encoded representations, using vector dimensions as features for inducing document classes.

3.1 KCG Construction

The KCG construction involves 4 steps: Given a document, KCG first uses Non-Negative Matrix Factorization (NMF) (Févotte and Idier, 2011; Cichocki and Phan, 2009) to extract the top-50 keywords of each document as nodes.[1] Second, each sentence in the document are mapped to the node it is most related to.[2] Thus, each node will have its own *sentence sets*. An example is shown in Figure 2. Then, we generate embeddings for each sentence in the set (referred to as *sentence set embeddings* henceforth). They will be served as features of the nodes. Last, edges between nodes are established by measuring the *correlations* of their corresponding sentence sets.

[1]Earlier approaches used mature NLP pipelines (e.g., named entity recognizer) for keyword extraction (Li et al., 2019; Liu et al., 2019). Instead, we use unsupervised NMF for keyword extraction. We tested with top-10, 20, 50, 100 keywords on Latent Dirichlet allocation (LDA) (Blei et al., 2003; Hoffman et al., 2010) and NMF. We found that using NMF to extract top-50 keywords gives the best clustering result.

[2]We map sentences and keywords based on the cosine similarity between their TFIDF features

Node Feature: We represent each keyword node as the average of its sentence set embeddings. A range of word- and sentence-level embeddings, including Global Vector (GloVe) (Pennington et al., 2014), BERT, Sentence-BERT (SBERT) (Reimers and Gurevych, 2019) and Embeddings from Language Models (ELMo) (Peters et al., 2018), are tested (see Section 5.1).

Word co-occurrence edge: The distributional hypothesis suggests that similar (key)words appear in similar contexts (Firth, 1957). Thus, the co-occurrence rate between two keywords reveals helpful clues for their relatedness. For this, we connect two keywords by their co-occurrence frequencies in sentences.

Sentence similarity edge: To estimate the global correlation between two keywords, we calculate the mean pairwise (cosine) similarity between their sentence embedding sets. Two keywords will have a high edge weight if their sentence set embeddings are similar.

Sentence position edge: The position of a word in the document can be an indicator of its importance. For example, topical keywords and sentences tend to appear in the beginning of the text (Lin and Hovy, 1997). Hence, we connect two keywords by computing the average position of their sentence sets in text. If two keywords both appear early in text, they will have a high edge weight. Details are described in the Appendix.

3.2 Graph Autoencoders (GAEs)

KCG captures the local and global features in documents using text embeddings and adjacency edges. After that, we compute the representation of each document by applying a GAE on the KCG. The GAE is an advanced version of the autoencoder for graph encoding, under an encoder-decoder framework. For each node in the KCG, the encoder aims to extract the *latent* features that can reconstruct the graph using the decoder. This way, the GAE learns to encode global information about (keyword) nodes that are multiple-hops away in the KCG. To capture the global features, while preserving the local ones, we use a Multi-Task GAE (**MTGAE**), whose objective is to jointly learn the latent representation that can reconstruct both the input graph and node features (Tran, 2018a,b). In Section 5.1, we will compare MTGAE performance with the GAE, the Variational GAE (**VGAE**) (Kipf and Welling, 2016), and a generic sequence-level autoencoder (**AE**) (Hinton and Salakhutdinov, 2006). The model settings are described in the Appendix.

Dataset	Size	#Classes	Avg. length	#Tokens
20ng	18,612	20	245	55,970
Reuters	7,316	10	141	27,792

Table 1: Datasets statistics

3.3 Clustering Algorithm

After we encode the KCG features for each node, we employ global average pooling over the node sequence to get a fixed-length representation of the document. We then apply the Spectral Clustering algorithm, on these representations to group documents into classes.[3] Spectral Clustering has wide applications in similar NLP tasks that involve high-dimensional feature spaces (Xu et al., 2015; Belkin and Niyogi, 2002; Xie and Xing, 2013).

4 Experiments

4.1 Datasets and Evaluation Metrics

We use two preprocessed datasets, Reuters-21578 (Reuters) (Lewis et al., 2004) and 20Newsgroups (20NG) (Lang, 1995), as provided by Ye et al. (2017) for long-text clustering. Their statistics are listed in Table 1. Following previous work (Ye et al., 2017; Xie and Xing, 2013; Xu et al., 2015), we use two sets of metrics to assess the quality of clusters: (1) Adjusted Mutual Information (AMI) (Vinh et al., 2010); and (2) Accuracy (ACC). Their descriptions are included in the Appendix.

4.2 Baseline Models

We compare our model with multiple cutting-edge text clustering and representation models, as reported by Ye et al. (2017) and Xie and Xing (2013). These include K-means on TFIDF models, Discrete Distribution Clustering on Skipgram embeddings (D2C) (Mikolov et al., 2013a; Ye et al., 2017);

[3] To emphasize the effect of the GAE on learning graphical information, we avoid using more advanced clustering methods, such as Deep clustering (Caron et al., 2018), which jointly learn feature representations and fine-tune the clustering performance during training. While this causes the performance of our model to fall notably below the state-of-the-art, we believe this minimal approach to be an effective way to focus on the quality of the document representations as they are created by our method, and we will leave the exploration of new clustering methods for future work.

Variable	MTGAE	AE
Batch size	20	20
Dim. of word emb	300	300
No. of layers	4	4
Input dropout	0.5	0.5
layer dropout	0.5	0.5
Learning rate	0.01	0.01
error proportion	1:1:1	–

Table 2: Hyper-parameters used in proposed model

NMF, LDA, Latent Semantic Indexing (LSI) (Deerwester et al., 1990), Locality Preserving Projection (LPP) (He and Niyogi, 2004; Cai et al., 2005), average of word embeddings (AvgDoc) and Paragraph Vectors (PV) (Mikolov et al., 2013b). Details on their settings can be found in Ye et al. (2017).

In addition to the aforementioned models, we also generate document embeddings using GloVe, BERT, ELMo and SBERT. Here, a document is represented as the average of the words/sentence embeddings in that document (**AvgEmb**).

5 Model Training

For embeddings, we use *GloVe-300d*, *BERT-base-uncased*, *ELMo-original* and *SBERT-bert-large-nli-stsb-mean-tokens* in our experiments. In all AEs, the ReLU activation function is employed in all layers. Parameters of all the models are optimized using the Adam optimisation algorithm with an initial learning rate of 0.01 (Kingma and Ba, 2014). We used early stopping with patience equal to 10 epochs in order to determine the best training epoch. Unless specific, other hyper-parameters are kept default as provided om their corresponding studies. The hyper-parameter values are shown in Table 2.

5.1 Results

Test Performance In Table 3, we show the results[4] of our main model (**SS-SB-MT**). It is created using **S**entence **S**imilarity (edge), SBERT (node) and MTGAE (autoencoder). From Table 3, our model is notably better than the baseline models, which showcases the effectiveness of topological features on long-text datasets. The main reasons our model performs well are twofold: first, the KCG can capture both the local and global features using text embeddings and adjacency edges (*resp.*). Second, the MTGAE is able to aggregate the two sets of features by jointly reconstructing them. To better analyze the behaviour of our model, we experiment with different edges, node features and autoencoders individually. We vary one variable at a time and keep others constant. We report the results in the next section.

Model	20NG AMI	20NG ACC	Reuters AMI	Reuters ACC
TFIDF	0.417†	0.337*	0.456†	0.350*
LSI	0.398†	0.323*	0.400†	0.420*
LPP	0.515†	0.117*	0.426†	0.331*
NMF	0.453†	0.319*	0.438†	0.496*
LDA	0.288†	0.372*	0.503†	0.549*
AvgDoc	0.376†	–	0.413†	–
PV	0.275†	–	0.471†	–
D2C	0.493†	–	0.534†	–
AvgEmb				
– GloVe	0.210	0.217	0.371	0.385
– ELMo	0.460	0.402	0.510	0.526
– BERT	0.405	0.419	0.426	0.471
– SBERT	0.451	0.441	0.524	0.514
SS-SB-MT (Ours)	**0.530**	**0.474**	**0.584**	**0.563**

Table 3: Performance of SS-SB-MT in comparison to various baseline models. * denotes performance reported by Xie and Xing (2013), † denotes performance reported by Ye et al. (2017). Bold: the best score for a dataset.

Model	20NG AMI	20NG ACC	Reuters AMI	Reuters ACC
SS-SB-MT	**0.530**	**0.474**	**0.584**	**0.563**
Edge Types				
– Word co-occurrence	0.466	0.440	0.524	0.500
– Sentence position	0.501	0.451	0.550	0.491
Embeddings				
– GloVe	0.336	0.387	0.431	0.455
– ELMo	0.481	0.421	0.582	0.579
– BERT	0.421	0.433	0.540	0.521
Autoencoders				
– VGAE	0.481	0.431	0.533	0.531
– GAE	0.493	0.414	0.484	0.523
– AE	0.487	0.417	0.550	0.537

Table 4: Performance of SS-SB-MT with different edge types, text embeddings as node features and autoencoders. Default: **S**entence **S**imilarity (Edge Types), SBERT (Embeddings) and MTGAE (Autoencoders). Bold: the best score for a dataset.

Impact of Edge Types, Node Features and Autoencoders We first analyze the performance of SS-SB-MT using different edge types[5], and report them in Table 4 (upper rows). Here, we see that the sentence-level edges perform better than the word-level edge. One possible reason is that text embed-

[4]We report the scores of cutting-edge models without any additional enhancements, such as joint training with topic modeling, to avoid any effects from them in the comparison.

[5]Currently, our model only supports encoding one edge type at a time, we leave the exploration of multi-edge GAEs for future work

AE	VGAE	**Ours**	Examples
0	1	1	A question in general about displaying NTSC through a Mac. If I understand correctly, the Video Spigot can display NTSC in a small window as well as capture the data in Quicktime format. However, if I want to use a larger window, what are my options? Perhaps I misunderstood the Video Spigot Also, I am not interested in Quicktime. I would merely like to use my Mac as a television from time to time. I have a nice Sony 1430 monitor, and I would like to use it as a second TV when my wife is watching sitcoms on our regular TV. Perhaps some of the video cards for the Mac accept NTSC input? I have a IIsi, and I am willing to buy a NuBus adapter.
1	0	1	The Duo Powerbooks seem to park the heads after a few seconds of is that builtin into the drive logic or is it being programmed via software, any way to tune the idle timeout that makes the heads park I think the heads are being parked since after a few seconds of inactivity you can hear the clunk of heads parking.
0	0	1	I have a Logitech 256 grays hand scanner from a PC. I'm wondering if anyone has been successful in connecting the scanner to a Mac? It has the same connector and is a serial device on the PC. I can imagine the pins configuration would need to changed, but I'm not sure if the signal levels would be correct, and if the Mac would work with it. Of course the manuals say nothing about the interface, connector layout or anything! Any ideas?

Table 5: Examples of error analysis in 20NG dataset. All examples are drawn from *comp.sys.mac.hardware*. *1*: Text is correctly clustered; and *0*: Text is wrongly clustered. "**Ours**" is our best proposed models (i.e., SS-SB-MT).

dings (e.g., SBERT) have already encoded the local semantic relations between adjacency words and sentences. An additional word co-occurrence edge may thus be less helpful.

We then analyze the performance of SS-SB-MT using different text embeddings to generate node features. From Table 4 (middle rows), we observe that sentence-level embeddings – SBERT (i.e., SB in SS-SB-MT) consistently outperforms the other word-level embeddings (GloVe, ELMo and BERT), suggesting that it can better represent the node features in the KCG.

We additionally conduct an analysis on different autoencoders. Results are shown in Table 4 (bottom rows). While graph-level autoencoders (GAE and VGAE) generally perform better than the sequence-level one (AE), the better results come when we use MTGAE (i.e., MT in SS-SB-MT) to aggregate local and global features, indicating the important roles of both features in document clustering.

Qualitative Analysis of Autoencoders. Table 5 showcases some prediction errors from AE and VGAE. All examples describe the hardware issues specifically about *Mac* (i.e., *comp.sys.mac.hardware*). We find that VGAE performs better when the document class is determined by the entire document or a long-range semantic relation that spans over multiple sentences, rather than some local relation in consecutive keywords. Example (1) contains both the "hardware-related" phrases (e.g., *Sony monitor*), as well as the "Mac-related" ones (e.g., *Mac*), but the whole document clearly refers to Mac if one explicitly considers the related context around the first and the last sentences; thus, an architecture likes VGAE is needed to fully utilize the semantic structures over long-sequences. In contrast, AE has a competitive advantage over VGAE in modelling the local dependencies among consecutive words, as shown in example (2). Here, VGAE captures the semantic features of some key-phrases such as *drive logic* and *heads* and misclusters the example to other group that talk about general hardware issues. But AE can effectively model consecutive features and capture the information about *Duo Powerbooks*. Similar to the previous two examples, example (3) also has a mixed keywords across different sentences, but neither the local features nor the global features alone are informative enough to interpret the topic of the document: AE may capture some local key-phrases such as *scanner* and *PC*, whereas VGAE may capture the non-local relations like *scanner from a PC* and *connecting the scanner to a Mac*. A scenario of this nature highlights the need for aggregating the two feature sets, and in essence, an effective model likes our MTGAE, that can exploit the synergy between them.

6 Conclusion

In this paper, we propose a document clustering model based on features induced unsupervisedly from a GAE and KCG. Our model offers an elegant way to learn features directly from large corpora, bypassing the dependence on mature NLP pipelines. Thus, it is not limited to resource-rich languages and can be used by any applications that operate on text. Experiments show that our model achieves better performance than the sequence-level representations, and we conduct a series of analyses to further understand the reasons behind such a performance gain.

References

Mikhail Belkin and Partha Niyogi. 2002. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in neural information processing systems*, pages 585–591.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

Deng Cai, Xiaofei He, and Jiawei Han. 2005. Document clustering using locality preserving indexing. *IEEE Transactions on Knowledge and Data Engineering*, 17(12):1624–1637.

Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. 2018. Deep clustering for unsupervised learning of visual features. In *European Conference on Computer Vision*.

Andrzej Cichocki and Anh-Huy Phan. 2009. Fast local algorithms for large scale nonnegative matrix and tensor factorizations. *IEICE transactions on fundamentals of electronics, communications and computer sciences*, 92(3):708–721.

Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*, pages 3844–3852.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Cédric Févotte and Jérôme Idier. 2011. Algorithms for nonnegative matrix factorization with the β-divergence. *Neural computation*, 23(9):2421–2456.

John Rupert Firth. 1957. A synopsis of linguistic theory 1930-1955 in studies in linguistic analysis, philological society.

Xiaofei He and Partha Niyogi. 2004. Locality preserving projections. In *Advances in neural information processing systems*, pages 153–160.

Geoffrey E Hinton and Ruslan R Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507.

Matthew Hoffman, Francis R Bach, and David M Blei. 2010. Online learning for latent dirichlet allocation. In *advances in neural information processing systems*, pages 856–864.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does bert learn about the structure of language? In *57th Annual Meeting of the Association for Computational Linguistics (ACL), Florence, Italy*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *nternational Conference on Learning Representations*.

Diederik P Kingma and Max Welling. 2014. Auto-encoding variational bayes. *stat*, 1050:1.

Thomas N Kipf and Max Welling. 2016. Variational graph auto-encoders. *NIPS Workshop on Bayesian Deep Learning*.

Ken Lang. 1995. Newsweeder: Learning to filter netnews. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 331–339.

David D Lewis, Yiming Yang, Tony G Rose, and Fan Li. 2004. Rcv1: A new benchmark collection for text categorization research. *Journal of machine learning research*, 5(Apr):361–397.

Wei Li, Jingjing Xu, Yancheng He, Shengli Yan, Yunfang Wu, and Xu Sun. 2019. Coherent comment generation for chinese articles with a graph-to-sequence model. In *ACL*.

Chin-Yew Lin and Eduard Hovy. 1997. Identifying topics by position. In *Fifth Conference on Applied Natural Language Processing*, pages 283–290.

Bang Liu, Di Niu, Haojie Wei, Jinghong Lin, Yancheng He, Kunfeng Lai, and Yu Xu. 2019. Matching article pairs with graphical decomposition and convolutions. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6284–6294.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013a. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013b. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab.

Christos Papadimitriou and Kenneth Steiglitz. 1982. *Combinatorial Optimization: Algorithms and Complexity*, volume 32.

Hao Peng, Jianxin Li, Yu He, Yaopeng Liu, Mengjiao Bao, Lihong Wang, Yangqiu Song, and Qiang Yang. 2018. Large-scale hierarchical text classification with recursively regularized deep graph-cnn. In *Proceedings of the 2018 World Wide Web Conference*, pages 1063–1072. International World Wide Web Conferences Steering Committee.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Jan Wira Gotama Putra and Takenobu Tokunaga. 2017. Evaluating text coherence based on semantic similarity graph. In *Proceedings of TextGraphs-11: the Workshop on Graph-based Methods for Natural Language Processing*, pages 76–85.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3973–3983.

Sunil Kumar Sahu, Fenia Christopoulou, Makoto Miwa, and Sophia Ananiadou. 2019. Inter-sentence relation extraction with document-level graph convolutional neural network. In *ACL*.

Phi Vu Tran. 2018a. Learning to make predictions on graphs with autoencoders. In *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*, pages 237–245. IEEE.

Phi Vu Tran. 2018b. Multi-task graph autoencoders. *NIPS 2018 Workshop on Relational Representation Learning*.

Nguyen Xuan Vinh, Julien Epps, and James Bailey. 2010. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *Journal of Machine Learning Research*, 11(Oct):2837–2854.

Pengtao Xie and Eric P. Xing. 2013. Integrating document clustering and topic modeling. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, UAI'13, pages 694–703, Arlington, Virginia, United States. AUAI Press.

Jiaming Xu, Peng Wang, Guanhua Tian, Bo Xu, Jun Zhao, Fangyuan Wang, and Hongwei Hao. 2015. Short text clustering via convolutional neural networks. *NAACL HLT 2015*, pages 62–69.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Graph convolutional networks for text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7370–7377.

Jianbo Ye, Yanran Li, Zhaohui Wu, James Z Wang, Wenjie Li, and Jia Li. 2017. 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.

Jie Zhou, Xingyi Cheng, and Jinchao Zhang. 2019. An end-to-end neural network framework for text clustering. *arXiv preprint arXiv:1903.09424*.

A Supplemental Material

Sentence Position Edge

We connect a keyword pair by considering the average position of their sentence sets in text. Formally, for a keyword node a with a sentence set of size m, its position score P_a is computed as:

$$P_a = \frac{\sum_{t=0}^{m} \exp(-\lambda * SentPos_t)}{m} \quad (1)$$

$SentPos$ denotes the position of the sentence in which the keyword a appears (e.g. 0 implies that a is in the first sentence of the document). λ is a decay parameter pre-defined, we use $\lambda = 0.2$ throughout the study. The final position weight between a keyword pair (P_{ab}) is calculated by the average of their position scores (i.e. $P_{ab} = \frac{P_a + P_b}{2}$).

Graph Autoencoder (GAE), Multi-Task GAE (MTGAE) and Variational GAE (VGAE)

Given a graph $G = (A, X)$ where A is the weighted adjacency matrix, denoting the correlation between the keyword nodes and X is the node feature matrix, the encoder aims at learning the *latent* representation (Z) that can effectively reconstruct A by the decoder. For a 2-layer encoder, its output is given by $Z^2 = Z = q\big(Z|Z, A\big)$,

$$\begin{aligned} Z^1 &= f_{relu}\big(Z^0, XA|W^0\big) \\ Z^2 &= f_{linear}\big(Z^1, XA|W^1\big) \end{aligned} \quad (2)$$

where $f\big(Z, XA|W\big)$ is a spectral convolutional operation for feature extraction, and Z is the set of latent features extracted for the nodes. The decoder reconstructs an adjacency matrix $\hat{A}$ by computing an inner product between these latent features, its output is given by $\hat{A} = p\big(A|Z\big)$,

$$\hat{A} = \sigma\big(ZZ^T\big) \quad (3)$$

During training, GAE learns to minimize the *reconstruction loss* ($\mathcal{L}_{\mathcal{R}}$), as measured by the cross entropy between its input A and its reconstructed $\hat{A}$,

$$\mathcal{L}_{\mathcal{R}} = E_{q(Z|X,A)}\big[\log p(A|Z)\big] \tag{4}$$

At inference time, we use the latent representation Z for document clustering and disregard the reconstructed part $\hat{A}$.

To encode more content information from the graph, one can reconstruct both the input adjacency matrix (A) and feature matrix (X). Regarding this, Tran (2018a,b) proposed the Multi-Task GAE (MTGAE). Here, the MT-reconstruction loss is defined as:

$$\mathcal{L}_{\mathcal{R}} = \mathcal{L}_{(a_i,\hat{a_i})} + \mathcal{L}_{(x_i,\hat{x_i})} \tag{5}$$

where $\mathcal{L}_{(a_i,\hat{a_i})}$ and $\mathcal{L}_{(x_i,\hat{x_i})}$ both are is the standard cross-entropy loss with sigmoid function $\sigma(\cdot)$, as,

$$\begin{aligned}\mathcal{L}_{(a_i,\hat{a_i})} &= -a_i\, log(\sigma(\hat{a_i})) - (1-a_i)\, log(1-\sigma(\hat{a_i}))\\ \mathcal{L}_{(x_i,\hat{x_i})} &= -x_i\, log(\sigma(\hat{x_i})) - (1-x_i)\, log(1-\sigma(\hat{x_i}))\end{aligned} \tag{6}$$

Variational Graph Autoencoder (VGAE) is an extension of the GAE architecture proposed by Kipf and Welling (2016). VGAE extends GAE by introducing an inference encoder, which is defined as:

$$\begin{aligned}q(Z|X,A) &= \prod_{i=1}^{n} q(z_i|X,A)\\ q(z_i|X,A) &= \mathcal{N}(x_i|\mu_i, diag(\sigma^2))\end{aligned} \tag{7}$$

$\mu = Z^2$ is a matrix of mean vectors z_i, $\sigma = f_l inear(Z^1, A|W^1)$ is the covariance matrix. During training, the VGAE optimizes the variational lower bound as:

$$\mathcal{L}_{\mathcal{R}} = \mathcal{L}_{\mathcal{R}} + KL(q(Z|X,A)||p(Z)) \tag{8}$$

$KL(q(\cdot)||p(\cdot))$ denotes the Kullback-Leibler divergance and $p(Z) = \prod_i \mathcal{N}(z_i|0,I)$ denotes the Gaussian prior for the latent data distribution. We perform the reparameterization trick (Kingma and Welling, 2014) to train the variational model.

Adjusted Mutual Information (AMI), and Accuracy (ACC)

Here, we describe the details of AMI and ACC. AMI is formally defined as:

$$AMI(U,C) = \frac{MI(U,C) - E\{MI(U,C)\}}{avg\{H(U),H(C)\} - E\{MI(U,C)\}} \tag{9}$$

U and C are the ground truth and predict classes (*resp.*). MI and H stand for Mutual Information and Entropy (*resp.*). $E\{MI\}$ stands for the expected mutual information.

ACC is formally defined as:

$$ACC = \frac{\sum_{i=1}^{N} \delta(y_i = map(c_i))}{N} \tag{10}$$

where $\delta(\cdot)$ is an indicator function, c_i is the predicted label for x_i, $map(\cdot)$ transforms the predicted label c_i to its group label by the Hungarian algorithm (Papadimitriou and Steiglitz, 1982), and y_i is the ground truth of x_i.