An Encoder-decoder Architecture with Graph Convolutional Networks for Abstractive Summarization

QiAo Yuan*; Pin Ni*†; Junru Liu*; Xiangzhi Tong*; Hanzhe Lu*; Gangmin Li*; Steven Guan*
* Research Lab for Knowledge and Wisdom, Xi'an Jiaotong-Liverpool University, China
†School of Computer Science, The University of Auckland, New Zealand

***Abstract*—We propose a single-document abstractive summarization system that integrates token relation into a traditional RNN-based encoder-decoder architecture. We employ pointer-wise mutual information to represent the token relation and adopt Graph Convolutional Networks (GCN) to extract token representation or graph representation from the relation graph. In our experiment on Gigaword, we consider importing two kinds of structural information: token (node) representation and graph representation. Also, we implement two kinds of GCNS, a spectral-based one and a spatial-based one, to extract structural information. The result shows that the spatial based GCN-enhanced model with node representation outperforms the classical RNN-based encoder-decoder model.**

***Index Terms*—GCN, Text Summarization, Seq2Seq, Natural Language Processing**

I. Introduction

Text summarization, a classical task for natural language processing systems, aims at generating concise and coherent summaries covering salient information from source documents. Compared with extractive summarization, which selects tokens or sentences from source documents, abstractive summarization can generate novel words or phrases, requiring machine learning models to have a deeper understanding of semantics[1].

RNN based sequence-to-sequence (Seq2Seq) model is a classical paradigm for text summarization with a sequence encoder to compress and distill the input sequences to a context vector and a decoder that maps the context vector and input to the output sequence [1]. Since the length of context vector output by encoders is fixed, when addressing long sequences, information loss is inevitable. it is difficult for RNN based Seq2Seq models to grasp long-distance relation. To address this problem, Bahdanau et al. [2] imported attention mechanism into Seq2seq architectures, which assigns weights to each encoder hidden states trying to store the most important information into the context vector. Recently, many improvements of this model focus on the adaption of the attention mechanism. However, as pointed out by Jia and Liang [3], although theoretically attention-based Seq2Seq models are expected to learn arbitrary long-distance dependency, they often perform poorly when handling long texts and get distracted by simple noise.

[1]The code can be found on https://github.com/Seven-Two/GCN-Seq2Seq-for-Summarization

In this work, we attempt to maintain long-distance dependency by importing Graph Convolutional Networks (GCN). GCNs are proposed mainly to handle problems with high-structured objects such as traffic forecasting [4] and social recommendation [5], but they are also successfully applied in fields with low-structural objects such as NLP. This article is an attempt to import graph representation of sequences containing structural information into Seq2Seq models.

Briefly, we propose two methods importing two kinds of GCNs, spectral-based GCNS and spatial-based GCNS, into a basic Seq2Seq architecture. We model graphs for each input sequence, treat each token as a node, and employ pointer-wise mutual information (PMI) as weights of edges. As for the first method, the node representation output from the GCNs is concatenated with token embedding as the input of Seq2Seq models. For the other, we add a supernode connecting each node. Throughout propagation, the supernode distills and integrates information from each node and edge, thus can be regarded as graph representation. Then we concatenate it with the context vector output from the encoder and feed the concatenated vector into the decoder.

II. Related Work

Graph convolutional networks (GCNS) are designed to extend convolution from Euclidean structures to non-Euclidean structures. Generally, they learn a mapping function f, from which a node x_i is able to integrate itself and its neighbors to generate a new representation of x_i. Applications of convolution on graphs are often categorized as the spectral approaches and spatial approaches [6, 7].

Spectral-based GCNs map the data from graph domain to spectral domain with Fourier transform to get the vectorized representation of the graph, which is preferable for convolution. Bruna [8] proposed the first generation GCN, it simply treats the whole convolution filter as the training parameters, laying a heavy burden on calculation. Defferrard et al. presented a Chebyshev network [9] reducing the computation complexity by importing a Chebyshev polynomial to approximate the convolution filter as a k-order polynomial form. Kipf et al. [10] further simplified the Chebyshev network as a first-order approximation of spectral graph convolutions by limiting the layer-wise convolution operation to $K = 1$ and regulating the filter parameter.

Spectral-based GCNs are said to be transductive since the trainable parameters rely on the Laplace matrix, which changes as the graph structure changes. To extend GCNS to inductive learning, spatial-based approaches, which define graph convolution on node's spatial relation, are adopted. GraphSAGE trains a set of aggregate functions that aggregate information from a node's neighbors to generate a new representation of this node [11]. Graph Attention Network then imports attention mechanism to assign weights of a node's neighbors [12].

There are several applications of GCNs in text summarization. Yasunaga et al. [13] adopted GCN on sentence relation graph to perform extractive multi-document summarization. The GCN takes sentence representation output from RNNs as node features. Fernandes [14] treats tokens, sentences, and entities as nodes. The edges are categorized into three types: next, in, and REF, respectively denoting a relation between tokens, sentences, entities, token-sentences, and token-entities. The token and sentence representation is obtained from RNNs, then GCN is applied to this graph to obtain graph-level information. In contrast, we regard tokens as nodes and employ pointer-wise mutual information(PMI) as weights of edges.

III. Methodology

A. Graph Construction

Each token in the input sequence is treated as a node and inspired by Yao et al. [15], pointer-wise mutual information (PMI) is employed to denote the association strength between two tokens. Specifically, the weighted adjacent matrix is defined as:

$$A(i,j) = \begin{cases} PMI(i,j), & i \neq j \ \& \ PMI(i,j) > 0 \\ 1, & i = j \\ 0, & i \neq j \ \& \ PMI(i,j) < 0 \end{cases} \quad (1)$$

PMI between two tokens i, j is defined as

$$PMI(i,j) = log\frac{P(i,j)}{P(i)P(j)} \quad (2)$$

where $P(i,j)$ denotes the possibility that token i and token$j$ appear in the same window and $P(i)$ denotes the possibility that token i appears in the whole sequence. They are respectively defined as:

$$P(i,j) = \frac{W(i,j)}{W}$$

$$P(i) = \frac{W(i)}{W}$$

where $W(i,j)$ denotes the occurrence frequency of windows containing token i,j and W denotes the total number of windows. A positive PMI indicates high semantic relation between two tokens, whereas a negative value indicates the opposite. Therefore, it is reasonable to add edges between tokens with positive PMI value.

Theoretically, the PMI between two tokens represents the information amount brought by one of the tokens or how much the uncertainty is reduced given one of the tokens. Each token in a pair of tokens would provide information to deduce another token. The more associative two tokens are, the more information one of the tokens provides. For instance, 'speed' and 'up' are known as associative since given the token 'speed', people are likely to guess the other token is 'up'. Shannon [16] imports the concept of entropy to quantify the information amount. For a set of possibilities $\{p_1, p_2, .., p_n\}$, the entropy H is defined as:

$$H = -\sum p_i log p_i$$

For an event x, the entropy of it is:

$$h(x) = -log p(x)$$

It is easy to understand why information entropy is inversely related to the possibility of an event. The less uncertain an event is, the less information it contains. For instance, 'the sun rises from the east' contains no new information since everyone knows this fact. In our case, for two tokens i, j in a sequence, to figure out what j is, the required information amount is $h(j)$. Given the token i, the required information amount to deduce j is $h(j|i)$. Therefore, $PMI(i,j)$, the information amount brought from i, can be acquired as:

$$PMI(i,j) = h(j) - h(j|i)$$

according to entropy's definition we have

$$h(j) = -log p_(j)$$

$$h(i|j) = -log p(j|i)$$

hence the equation can be written as

$$PMI(i,j) = log p(j|i) - log p(j)$$

according to conditional probability formula, we have

$$p(j|i) = \frac{p(i,j)}{p(i)}$$

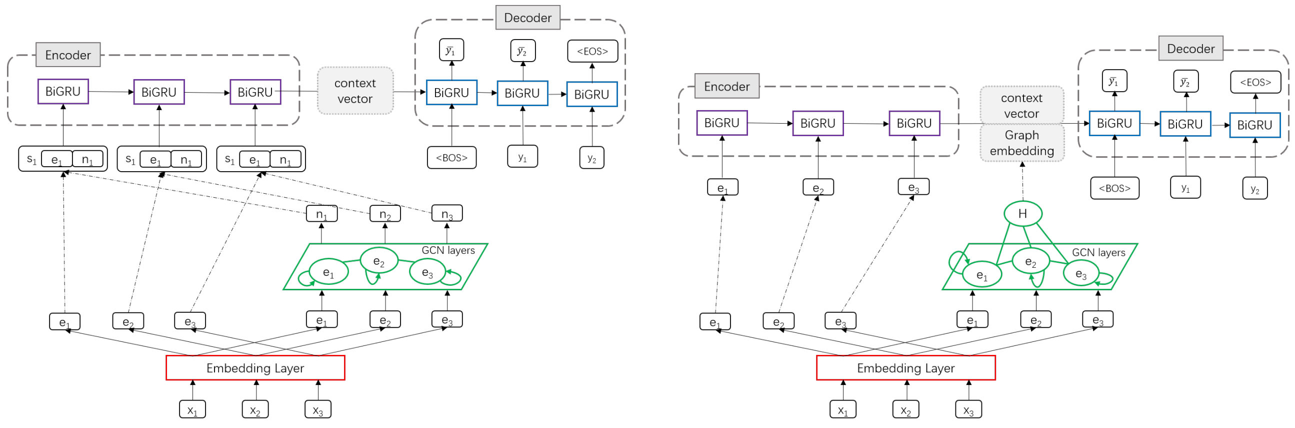
then the equation can be transformed as:

$$PMI(i,j) = log\frac{p(i,j)}{p(i)} - log p_(j) = log\frac{p(i,j)}{p(i)p(j)}$$

this is the origin of equation (2).

B. Spectral Graph Convolutional Network

A symmetric normalized Laplacian matrix, defined as $L = I_n - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$, is a mathematical representation of an undirected graph. I is the identity matrix, $D_{ii} = \sum_j A_{ij}$ and A is the adjacent matrix. As a positive semidefinite matrix, L can be decomposed as $L = U \Lambda U^T$ where $U = \{U_1, U_2, ..., U_n\}$ is an orthogonal matrix composed of column vectors as unit eigenvectors and $\Lambda = diag(\lambda_1, \lambda_2, ..., \lambda_n)$ is a diagonal matrix composed of corresponding eigenvalues. Since the Fourier basis e^{-ikx} is an eigenfunction of Laplace Operator, to perform Fourier transform on graphs, the Fourier basis can be replaced with U(eigenvectors can be treated as a discrete form of eigenfunction). Therefore, Fourier transform on graphs



(a) GCN+Seq2Seq with Node Representation

(b) GCN+Seq2Seq with Graph Representation

Figure 1: Proposed Networks

is defined as $\widehat{f} = U^T f$, where $f = \{f_1, f_2, ..., f_3\}$ is a n-dimensional vector consisting of node features. The inverse Fourier transform on graphs is: $f = U\widehat{f}$. According to the convolution theorem, the Fourier transform of the convolution function is the product of the Fourier transform of the two functions. Hence, the graph convolution with input x and a filter g is defined as

$$G_g * x = U(U^T g \odot U^T x)$$

where $\odot$ denotes the element-wise product. If we define the filter as $g_\theta = U^T g$, the graph convolution can be simplified as:

$$G_{g_\theta} * x = U g_\theta U^T x \tag{3}$$

all the spectral GCNNs follow this definition, the different lines in how they manipulate and train the filter g_θ. Bruna et al. propose the first generation of spectral GCN which simply treats g_θ as the training parameter [8]. The propagation rule is defined as:

$$y = \sigma(U g_\theta U^T x)$$

where y is the activated output. The complexity of this kind of GCN is high due to the operation of spectral decomposition to get U and great number of parameters in convolution kernel(number of nodes). Chebyshev Spectral GCN is proposed to alleviate this problem [9]. Since graph Fourier transform is a function of eigenvalues, it defines $g_\theta(\Lambda)$ as the k-order polynominal form:

$$g_{\theta'}(\Lambda) \approx \sum_{k=0}^{K} \theta'_k \Lambda^k$$

Then the graph convolution is:

$$G_{g_{\theta'}} * x = \sum_{k=0}^{K} \theta'_k L^k x$$

Furthermore, a truncated expansion in terms of Chebyshev polynomials $Tk(x)$ up to k^{th} order is imported to approximate L^k:

$$g_{\theta'}(\Lambda) \approx \sum_{k=0}^{K} \theta'_k T_k(\widetilde{\Lambda})$$

where $\widetilde{\Lambda} = \frac{2}{\lambda_{max}}\Lambda - I_n$, λ_{max} is the largest eigenvalue of L. The Chebyshev polynomial is defined recursively as $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$ with $T_0(x) = 1$, $T_1(x) = x$. As a result, the graph convolution is:

$$G_{g_{\theta'}} * x = \sum_{k=0}^{K} \theta'_k T_k(\widetilde{L})x$$

And the propagation rule is:

$$y = \sigma(\sum_{k=0}^{K} \theta'_k T_k(\widetilde{L})x)$$

As an improvement over spectral GCNNS, the ChebNet reduces the computation complexity since there is no need performing spectral decomposition and L^k can be calculated in advance. Also, it reduces the number of parameters in convolution kernel from n to k, thus realizing localization within k-hop [6]. Kipf et al. propose a further simplification over ChebNet [10]. Firstly they limit the layer-wise convolution operation to $K = 1$. Although by limiting K, the network can merely extract information from first-order neighbors, further proximity extraction can be recovered by stacking multiple layers. Furthermore, they approximate $\lambda \approx 2$ since the model will adapt to the scale change during training. As a result, the graph convolution can be simplified as:

$$G_{g_{\theta'}} * x \approx \theta'_0 x + \theta'_1 (L - I_N)x = \theta'_0 x - \theta'_1 D^{-\frac{1}{2}} A D^{-\frac{1}{2}} x$$

with two parameters θ'_0 and θ'_1. These parameters are constrained as $\theta = \theta'_0 = -\theta'_1$ to alleviate overfitting and reduce operations. This leave us with the follow expression:

$$g_\theta * x \approx \theta(I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}})x$$

Since the eigenvalues of $I_N + D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ has range in $[0, 2]$ which can lead to exploding/vanishing gradients under repeated opearation, a renormalization trick is performed as $I_N + D^{-\frac{1}{2}}AD^{-\frac{1}{2}} \rightarrow \widetilde{D}^{-\frac{1}{2}}\widetilde{A}\widetilde{D}^{-\frac{1}{2}}$ with $\widetilde{A} = A + I_N$ and $\widetilde{D}_{ii} = \sum_j \widetilde{A}_{ij}$ Then the propagation rule between layers is defined as:

$$H^{l+1} = \sigma(\widetilde{D}^{-\frac{1}{2}}\widetilde{A}\widetilde{D}^{-\frac{1}{2}}H^lW^l) \quad (4)$$

where H^l denotes the activated matrix in l^{th} layer($H^0 = x$)and W^l denotes the layer-specific trainable weight matrix.

C. Spatial Graph Convolutional Network

In this article, we employ graph attention network (GAT) as the spatial GCN. The spectral GCNs aggregate neighbors with fixed weights. Specifically, the weight between node i and node j is $\frac{1}{\sqrt{D_{ii}D_{jj}}}$ where D_{ii} denotes the degree of node i. GAT imports attention mechanism into the aggregation process. It concatenates the features of two nodes and assigns weights to the edge according to the similarity. The input is a set of node features $H = \{h_1, h_2, h_3, ..., h_n\}$, and the output is a set of node representation$\widehat{H} = \{\widehat{h_1}, \widehat{h_2}, \widehat{h_3}, ...\widehat{h_n}\}$ containing structural information. Similar to spectral GCNs, the Laplace matrix is used to represent the graph. GAT performs an aggregation on the input as: $H^{l+1} = Agg(L, H^l)$ where H^l denotes the node representation in l^{th} layer and L denotes the Laplace matrix. Specifically, In each layer, each node firstly experiences a linear transformation with a shared weight matrix as:

$$e_{ij} = a(Wh_i, Wh_j)$$

where a is the shared attention matrix initialized as

$$a_{ij} = \begin{cases} rand(-1, 1), & L_{ij} > 0 \\ 0, & L_{ij} \leq 0 \end{cases}$$

e_{ij} is the attention score of node j to node i indicating the importance of node j to node i. In this article, only the first-order neighbors attend the attention mechanism. For a node i with its neighbors $j \in N_i$ where N_i is the first-order receptive field of node i, the attention weight is defined as:

$$\alpha_{ij} = \frac{exp(LeakyReLu(a(Wh_i, Wh_j)))}{\sum_{k \in N_i} exp(LeakyReLu(a(Wh_i, Wh_k)))}$$

Having obtained the normalized attention weight, by aggregating the neighbors' features of each node, we can get the new node representation of the whole graph. The aggregating function is defined as:

$$\widehat{h_i} = \sigma(\sum_{j \in N_i} \alpha_{ij}h_i)$$

Similar with self-attention mechanism in Transformers, multi-head attention is also applicable here to stabilize the learning process.

D. GCN+Seq2Seq with Node Representation

This model constructs graphs for each input sequence and gets the node representation of each token. Then the node representation is concatenated with token vectors as the inputs of the Seq2Seq structure. Figure 1-a illustrates the architecture of this model.

To be clear, we name the adopted spectral GCN in this article as KGCN. In detail, the input is a sequence $X = \{x_1, x_2, .., x_m\}$ containing m tokens where x_i denotes the i^th word in the sequence. The first layer is an embedding layer that maps each token to a 300-dimensional vector $e_i \in R^{300}$. Subsequently, these vector sequence $\{e_1, e_2, .., e_m\}$ is input into the GCN layer as node features. Each sequence is modeled as a graph represented as an adjacent matrix A defined above. We employ a two-layer GCN network since Kipf[10] suggested that a network with two layers performs well. For KGCN, the output $N = \{n_1, n_2, .., n_m\}$ is obtained as:

$$N = tanh(D^{-\frac{1}{2}}AD^{-\frac{1}{2}}ReLu(D^{-\frac{1}{2}}AD^{-\frac{1}{2}}XW_0)W_1) \quad (5)$$

For GAT, the propagation rule is:

$$N = tanh(Agg(D^{-\frac{1}{2}}AD^{-\frac{1}{2}}, elu(Agg(D^{-\frac{1}{2}}AD^{-\frac{1}{2}}, X))) \quad (6)$$

where Agg represents the aggregation function defined in part C. After that, the node representation $N = \{n_1, n_2, .., n_m\}$ is concatenated with the token vectors as the input $S = \{s_1, s_2, ..., s_m\}$, defined as $S = [E; N]$, of Seq2Seq structure. Both the encoder and decoder of Seq2Seq contain a two-layer bidirectional GRU structure [17], their forward propagation rules are similar. At time step t, each encoder or decoder unit receives an input token and a last hidden state, then outputs a hidden state to the next unit, defined as:

$$h_t^{enc} = Encoder(s_t, h_{t-1}^{enc})$$

$$h_t^{dec} = Decoder(\widetilde{y}_t, h_{t-1}^{dec})$$

where h_t denotes the hidden state and at time step t and s_t, $\widetilde{y}_t$ denote the encoder input and decoder input at time step t respectively. As for the encoder, $h_0^{enc} = 0$, whereas for the decoder, $h_0^{dec} = h_{last}^{enc}$. The operation in decoders is a little more complex than that in encoders. The teacher force mechanism is imported into the training process. During testing, the current decoder output is treated as the input of the next decoder unit. However, this is inefficient during training because an unconverged model is likely to make mistakes, thus affecting all the next units. Therefore, by importing teacher force, the decoder is fed with labels to avoid error accumulation. Since there is no support from labels in the testing process, the model should not completely rely on the labels. Typically, a teacher force ratio is imported defining the possibility of using teacher force mode. Furthermore, the beam search mechanism is also implemented in the decoder, which caches several outputs with the lowest cost at each step and selects the optimal combination finally. The output of the decoder in each step is a possibility distribution of all tokens and the model is trained to minimize the loss defined as:

$$loss = -\frac{1}{n}\sum_{i}^{n}(y_i)log(\widetilde{y}_i) \tag{7}$$

where n denotes the vocabulary length, y denotes the labels represented as one-hot encoding and $\widetilde{y}$ denotes the decoder output activated by Softmax function.

E. GCN+Seq2Seq with Graph Representation

As Figure 1-b demonstrates, the improvement of these network lines in the incremental information extracted from the graph structure. Inspired by Xu et al. [18], we add a node to the graph modeled from input sequences, called supernode, connecting all the other nodes with weights as 1. The feature of this node is initialized as a zero matrix since the graph is undirected, the supernode also transfers information to other nodes, by setting it as a zero matrix, at least it will not affect other nodes at the first layer. Also, this node has no self-loop because it is meaningless to learn from a zero matrix. The supernode is expected to distill and compress information from all the nodes and edges from the graph by stacking two layers, thus it can be treated as graph representation. By concatenating the graph representation and the context vector, the decoder can learn from both semantic and structural information.

To make the graph representation compatible with the context vector, we set the output dimension of GCN layers equal to the hidden layer dimension of the decoder and simply stack the graph representation fourth vertically to make it consistent with the dimension of the context vector.

IV. Experiment

A. Experiment Environment

Our experimental environment is as follows: CPU Intel Xeon E5-2678 v3, RAM: Dual 2.50GHz, GPU: Dual Nvidia GeForce GTX 1080 Ti.

B. Dataset Description

All the tests were run on the Gigaword dataset containing 3,993,608 items. Due to computation ability restriction, we randomly select 100,000 items as training data, 10,000 items as validation data, and 10,000 items as test data. Each item contains an English Article and its title, the title can be treated as the summary. As for the training data, the longest article has 38 tokens and the longest title has 11 tokens.

C. Implementation

We adopted Glove-300 as the pre-trained word embedding weight to get 300-dimensional word embedding. Both the encoder and decoder contain a two-layer bidirectional GRU structure whose hidden dimension is 512. The teacher force ratio is 0.5 and to accelerate training, the beam search range is set to 1, thus it is equivalent to greedy search. Both GCNs have two layers with hidden dimensions as 512, in Seq2Seq with node representation model, the output dimension is 300, whereas, in Seq2Seq with graph representation model, the output dimension is 256. The dropout is 0.5 and the PMI window size is 3. We use mini-batch stochastic gradient descent to minimize negative log-likelihood with batch size as 16 and learning rate as 0.0001. We trained each model until the validation loss has not decreased for three epochs.

D. Evaluation Criteria

We implemented ROUGE-1, ROUGE-2 and ROUGE-L [19] to evaluate the quality of summary texts. ROUGE-1 and ROUGE-2 follow the definition of ROUGE-N which denotes an n-gram recall between a candidate summary and reference summaries, defined as:

$$Rouge{-}N = \frac{\sum_{S\in RefSum}\sum_{gram_n\in S}Count_{match}(gram_n)}{\sum_{S\in RefSum}\sum_{gram_n\in S}Count(gram_n)}$$

where $RefSum$ denotes the reference summaries, n stands for the length of the n-gram, $Count_{match(gram_n)}$ denotes the number of n-grams co-occurring in the candidate summary and reference summaries. As for summary-level ROUGE-L, it takes the union longest common sub-sequence matches between the candidate summaries and candidate summaries. Given a reference summary X of m words and a candidate summary Y, the summary-level ROUGE-L can be computed as:

$$ROUGE - L = \frac{LCS(X,Y)}{m}$$

where $LCS(X,Y)$ is the length of the longest common sub-sequences between X and Y.

V. Results and Discussion

The main results are presented in Table 1 and the validation loss during training is illustrated in figure 2. According to the Table, generally, the scores are greatly lower than most research since we only use 100,000 items as training data. Specifically, according to the table, we note that node-based GAT-enhanced model successfully outperforms the basic RNN-based Seq2Seq model, whereas the other GCN-enhanced models perform worse, especially the node-based KGCN-enhanced model. In terms of the validation loss figure, the node-based models reach a lower validation loss than the basic Seq2Seq models while graph-based models stop converging when the loss is still relatively high. It is counter-common sense that the node-based KGCN-enhanced model reaches low validation loss but performs worst in ROUGE evaluation. The reason is that the training parameters of this kind of spectral-based GCN is highly related with the Laplace matrix, which represents the information of the whole graph. Any change of the graph structure will cause the change of the Laplace matrix, thus affecting the training parameters. Therefore, spectral-based GCNs are merely suitable for transductive tasks where the test data is exposed to the model during the training process. From this perspective, the structural information acquired from KGCN can be treated as noise. Therefore, it is reasonable that graph-based KGCN-enhanced model perform better than the node-based one which imports more noise information. Regarding to the GAT-enhanced model, since the node-based one outperforms the graph-based one, it can be safe to conclude that the structural information imported by GAT is meaningful. The graph representation is simply concatenated with the context vector, the model may not understand the representation well since they are from different spaces. The node-based models are effective because the model learns the node representation along with token embedding through the encoder. However, in graph-based models, the graph representation is directly fed into the decoder so that the model cannot learn this representation thoroughly.

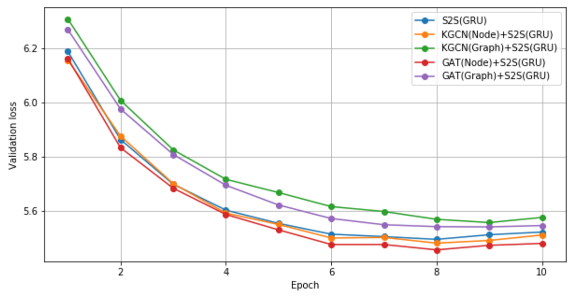


Figure 2: Validation loss during training

TABLE I: ROUGE SCORE

	ROUGE-1	ROUGE-2	ROUGE-L
S2S(GRU)	20.66	4.31	22.62
KGCN(node)+S2S(GRU)	9.39	1.03	10.02
KGCN(graph)+S2S(GRU)	19.33	3.72	21.25
GAT(node)+S2S(GRU)	**20.74**	**4.56**	**23.02**
GAT(graph)+S2S(GRU)	20.03	4.04	21.71

VI. Conclusion

In conclusion, to compensate for the weakness of conventional encoder-decoder architecture in building long-distance relationships, we presented two approaches combining two kinds of Graph Convolutional Networks and Encoder-decoder architectures to import additional structural information. Treating the traditional Seq2Seq model as the baseline, we compared five models on the Gigaword dataset The result shows that the node-based GAT-enhanced model outperforms the basic RNN-based Seq2Seq models, indicating GAT can be used to import structural information into a RNN-based Seq2Seq architecture.

In future work, the models will be trained on larger data to get a more convincing result. Also, we will try to adopt structural information with GCNs into more advanced models such as pointer network and transformer, to further explore its effectiveness.

Acknowledgment

This study is partially supported by the AI University Research Centre (AI-URC) through the XJTLU Key Program Special Fund (KSF-P-02) and KSF-A-17.

References

[1] I. Sutskever, O. Vinyals, Q. V. Le, Sequence to sequence learning with neural networks, Advances in Neural Information Processing Systems 27 (2014) 3104–3112.
[2] D. Bahdanau, K. H. Cho, Y. Bengio, Neural machine translation by jointly learning to align and translate, in: 3rd International Conference on Learning Representations, ICLR 2015, 2015.
[3] R. Jia, P. Liang, Adversarial examples for evaluating reading comprehension systems, in: EMNLP, Association for Computational Linguistics, Copenhagen, Denmark, 2017, pp. 2021–2031. doi:10.18653/v1/D17-1215.
[4] W. Chen, L. Chen, Y. Xie, W. Cao, Y. Gao, X. Feng, Multi-range attentive bicomponent graph convolutional network for traffic forecasting, AAAI 34 (2020) 3529–3536. doi:10.1609/aaai.v34i04.5758.
[5] L. Wu, P. Sun, R. Hong, Y. Fu, X. Wang, M. Wang, Socialgcn: An efficient graph convolutional network based model for social recommendation, in: AAAI, 2019.
[6] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, P. S. Yu, A comprehensive survey on graph neural networks, IEEE Transactions on Neural Networks and Learning Systems 32 (2021) 4–24.
[7] J. Zhou, G. Cui, Z. Zhang, C. Yang, Z. Liu, M. Sun, Graph neural networks: A review of methods and applications, ArXiv abs/1812.08434 (2018).
[8] J. Bruna, W. Zaremba, A. Szlam, Y. LeCun, Spectral networks and deep locally connected networks on graphs, in: 2nd International Conference on Learning Representations, ICLR 2014, 2014.
[9] M. Defferrard, X. Bresson, P. Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering, in: NIPS, 2016.
[10] T. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, ArXiv abs/1609.02907 (2017).
[11] W. L. Hamilton, Z. Ying, J. Leskovec, Inductive representation learning on large graphs, in: NIPS, 2017.
[12] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, Y. Bengio, Graph attention networks, ArXiv abs/1710.10903 (2018).
[13] M. Yasunaga, R. Zhang, K. Meelu, A. Pareek, K. Srinivasan, D. R. Radev, Graph-based neural multi-document summarization, ArXiv abs/1706.06681 (2017).
[14] P. Fernandes, M. Allamanis, M. Brockschmidt, Structured neural summarization, in: International Conference on Learning Representations, 2019.
[15] L. Yao, C. Mao, Y. Luo, Graph convolutional networks for text classification, in: AAAI, 2019.
[16] C. E. Shannon, A mathematical theory of communication, The Bell System Technical Journal 27 (3) (1948) 379–423. doi:10.1002/j.1538-7305.1948.tb01338.x.
[17] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning phrase representations using RNN encoder–decoder for statistical machine translation, in: EMNLP, Association for Computational Linguistics, Doha, Qatar, 2014, pp. 1724–1734. doi:10.3115/v1/D14-1179.
[18] K. Xu, L. Wu, Z. Wang, Y. Feng, V. Sheinin, Graph2seq: Graph to sequence learning with attention-based neural networks, ArXiv abs/1804.00823 (2018).
[19] C.-Y. Lin, ROUGE: A package for automatic evaluation of summaries, in: Text Summarization Branches Out, Association for Computational Linguistics, Barcelona, Spain, 2004, pp. 74–81.