Proceedings Paper:
Miao, Q, Zhu, H, Liu, J et al. (5 more authors) (2022) MuchSUM: A Multi-channel Graph Neural Network for Extractive Summarization. In: Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval. The 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR), 11-15 Jul 2022, Madrid, Spain. ACM , pp. 2617-2622. ISBN 978-1-4503-8732-3

https://doi.org/10.1145/3477495.3531906

MuchSUM: A Multi-channel Graph Neural Network for Extractive Summarization

Qianren Mao[1,2], Hongdong Zhu[1,2], Junnan Liu[1,2], Cheng Ji[1,2], Hao Peng[1,2], Jianxin Li[*,1,2], Lihong Wang[3], Zheng Wang[4]
[1]Beijing Advanced Innovation Center for Big Data and Brain Computing, Beihang University, China.
[2]The State Key Laboratory of Software Development Environment,Beihang University, China.
[3]CNCERT, China. [4]The School of Computing, University of Leeds, U.K.
{maoqr,zhuhd,liujn,jicheng,penghao,lijx}@act.buaa.edu.cn
wlh@isc.org.cn,z.wang5@leeds.ac.uk

ABSTRACT

Recent studies of extractive text summarization have leveraged BERT for document encoding with breakthrough performance. However, when using a pre-trained BERT-based encoder, existing approaches for selecting representative sentences for text summarization are inadequate since the encoder is not explicitly trained for representing sentences. Simply providing the BERT-initialized sentences to cross-sentential graph-based neural networks (GNNs) to encode semantic features of the sentences is not ideal because doing so fail to integrate other summary-worthy features like sentence importance and positions. This paper presents MuchSUM, a better approach for extractive text summarization. MuchSUM is a multi-channel graph convolutional network designed to explicitly incorporate multiple salient summary-worthy features. Specifically, we introduce three specific graph channels to encode the node textual features, node centrality features, and node position features, respectively, under bipartite word-sentence heterogeneous graphs. Then, a cross-channel convolution operation is designed to distill the common graph representations shared by different channels. Finally, the sentence representations of each channel are fused for extractive summarization. We also investigate three weighted graphs in each channel to infuse edge features for graph-based summarization modeling. Experimental results demonstrate our model can achieve considerable performance compared with some BERT-initialized graph-based extractive summarization systems.

CCS CONCEPTS

• **Artificial intelligence → Natural language processing**; • **Retrieval tasks and goals → Summarization**.

KEYWORDS

extractive summarization, multi-channel graph, text summarization, bipartite word-sentence heterogeneous graph.

*Jianxin Li is the corresponding author.

SIGIR '22, July 11–15, 2022, Madrid, Spain.

ACM ISBN 978-1-4503-8732-3/22/07.
https://doi.org/10.1145/3477495.3531906

ACM Reference Format:
Qianren Mao[1,2], Hongdong Zhu[1,2], Junnan Liu[1,2], Cheng Ji[1,2], Hao Peng[1,2], Jianxin Li[*,1,2],, Lihong Wang[3], Zheng Wang[4]. 2022. MuchSUM: A Multi-channel Graph Neural Network for Extractive Summarization. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '22), July 11–15, 2022, Madrid, Spain.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3477495.3531906

1 INTRODUCTION

Extractive text summarization produces a summary by identifying the most representative sentences in a document. Most of the existing techniques in extractive text summarization [5, 11, 14, 15, 25, 35, 37] formulate the problem as a sequence labeling task, where the labels indicate if a sentence should be included in the generated summary. Most of the recent approaches in this direction employ pre-trained language models such as BERT [1] to learn sentence representation for text summarization, achieving some of the state-of-the-art results [11, 33, 36].

While promising, the sentence representations used by existing BERT-based methods still have room for improvement. Existing language models model sentences word-by-word, but doing so ignores the semantic relations among sentences [3]. Indeed, recent studies have shown that the sentence representations given by a vanilla BERT are inadequate to distinguish sentences with different semantics [10, 19, 31].

Efforts have been made to better capture inter-sentence relationships by modeling summarization graphs. Early studies, such as unsupervised LexRank [2] and TextRank [13], built similarity graphs among sentences leveraged PageRank [16] to rank them by estimating summary-worthy features of sentence importance. Recently, some works have applied graph representation learning techniques on various semantic graphs [3, 5, 6, 17, 28, 32] with consideration of semantic similarity and the natural topology. However, they usually rely on external tools to construct the graphs, in which the error propagation problem is serious. Furthermore, there is no consensus on the best neural graph formulation to leverage the topological centrality of a cross-sentential summarization graph.

There is an extensive body of work [9, 26, 36] showing that the success of an extractive summarization system heavily relies on learning sentence position information. For example, the work in [11] empirically shows that LEAD-3 gives similar performance compared with abstractive or extractive Transformer models. Zhong et al. [35] find that Transformer encoder [24] equipped with lexical embeddings has similar or even inferior performance

to the model that only equips positional embeddings. These studies provide compelling evidence, showing the positive impact of position features on extractive summarization.

We present MuchSUM, a multi-channel convolutional graph neural network (GNN) for modeling summarization graphs. MuchSUM is designed to explicitly integrate summary-related features, like sentence semantics and importance and position. Specifically, we introduce a semantic encoding channel to learn sentence linguistic features, a centrality encoding channel to learn sentence importance features, and a position encoding channel to learn sentence position features. Besides, considering the same topology of the heterogeneous bipartite word-sentence graph used in the three channels, we use the common convolution module optimized by the consistency constraints to distill the 'common' property among three specific feature spaces. Meanwhile, the three feature channels are optimized by disparity constraints to ensure the specific embeddings in each feature space. The rationale is that these summary-worthy features aggregated by the same bipartite topological structures complement can be fused to derive deeper inter-sentence relationships for better recognizing summary sentences. We evaluate MuchSUM by applying it to the CNN/DailyMail benchmark dataset. Experimental results show that our approach outperforms BERT-initialized summarization graph models with distinctive performance gains.

This paper makes the following contributions:

- It is the first to transfer the multi-channel GCN for extractive summarization under bipartite word-sentence graphs.
- It provides a comprehensive study on how to aggregate three kinds of summary-worthy features in an explicit manner based on the multi-channel graph convolution operation.
- It demonstrates how weighted graphs can be employed to aggregate node features of neighbors in word-sentence graphs.

2 MUCHSUM: THE PROPOSED METHOD

We present an overall structure of the proposed MuchSUM in Figure 1, where an input will be fed into the three convolutional graph modules, semantics graph $\mathcal{G}_S = (\mathbf{A}, \mathbf{X}_S)$, centrality graph $\mathcal{G}_C = (\mathbf{A}, \mathbf{X}_C)$ and position graph $\mathcal{G}_P = (\mathbf{A}, \mathbf{X}_P)$. Further, considering that the three specific graphs have common features because of their unique topology, we use a common convolution module with parameter sharing strategy to learn the common embeddings $\mathbf{Z}_{ms}$, $\mathbf{Z}_{mc}$ and $\mathbf{Z}_{mp}$. During optimizing by downstream sentences classification, a consistency constraint $\mathcal{L}_C$ is employed to enhance the common property of $\mathbf{Z}_{ms}$, $\mathbf{Z}_{mc}$ and $\mathbf{Z}_{mp}$. Meanwhile, a disparity constraint $\mathcal{L}_d$ is to ensure the independence between $\mathbf{Z}_S$ and $\mathbf{Z}_{ms}$, $\mathbf{Z}_C$ and $\mathbf{Z}_{mc}$, as well as $\mathbf{Z}_P$ and $\mathbf{Z}_{mp}$. Then we fuse these six embeddings $\mathbf{Z}_S$, $\mathbf{Z}_C$, $\mathbf{Z}_P$, $\mathbf{Z}_{ms}$, $\mathbf{Z}_{mc}$ and $\mathbf{Z}_{mp}$ optimally to obtain the final embedding $\mathbf{Z}$ for the sentence binary classification task.

2.0.1 Node Lexical Feature Encoding Channel. We explore the potential of BERT to initialize textual representations of graph nodes. Thus, the textual features of graph nodes are initialized by BERT. Specifically, the single input document after tokenization is denoted $D = \{w_{1s_1}, ..., s_1, ..., w_{1s_n}, ..., s_n\}$ in which token of w_{is_j} is i-th word in j-th sentence and the sentence representation s_i is i-th [CLS] token in BERT. The initialized node representation $\mathbf{H}_s^{(0)} = BERT(\{w_{1s_1}, ..., s_1, ..., w_{1s_n}, ..., s_n\})$. The lexical feature $\mathbf{X}_s$ of graph nodes is, $\mathbf{X}_s = \mathbf{H}_s^{(0)}$. We use GCN [8] as the module to encode

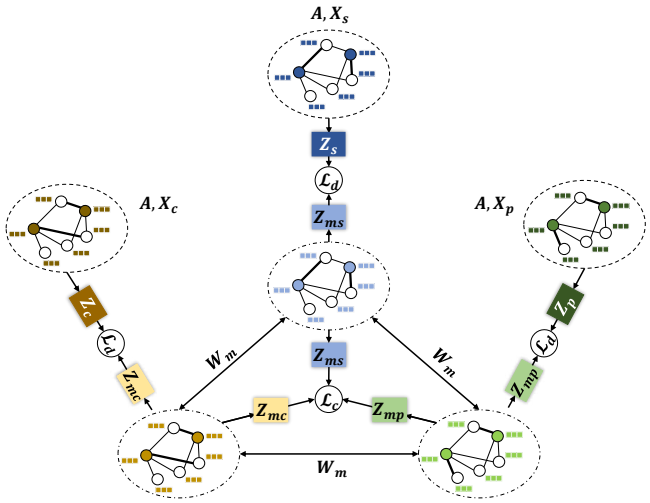


Figure 1: The overview architecture of the MuchSUM with three specific graph convolutional channels and a common convolutional channel shared by the three graph channels. We denote three specific channels as Node Lexical Feature Encoding Channel $(\mathrm{A}, \mathrm{X}_s)$, Node Centrality Feature Encoding Channel $(\mathrm{A}, \mathrm{X}_c)$ and Node Position Feature Encoding Channel $(\mathrm{A}, \mathrm{X}_p)$. In the bipartite word-sentence heterogeneous graph, each sentence node (solid node) is connected to its contained word-related nodes (hollow nodes) and takes the weight of the relation as their edge feature. Different thicknesses of edges represents different edge weights.

lexical channel, which is defined as:

$$\mathbf{Z}_s^{(k+1)} = Relu\left(\widetilde{\mathbf{D}}^{-\frac{1}{2}}\widetilde{\mathbf{A}}\widetilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{Z}_s^{(k-1)}\mathbf{W}_s^{(k)}\right), \tag{1}$$

where $\widetilde{\mathbf{D}}$ is the diagonal degree matrix of $\widetilde{\mathbf{A}}$ and $\widetilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ which is the adjacency matrix of the lexical graph and $A_{i,j} = \{0, 1\}$. $\mathbf{W}_s^{(k)} \in \mathbb{R}^{d_{k-1} \times d_k}$ is parameter matrix of the lexical channel. $\mathbf{Z}_s^{(0)} = \mathbf{X}_s$ is the input representation matrix of the model. Each row $\mathbf{Z}_i \in \mathbb{R}^d$ is the d-dimension representation of node i. We denote the last layer output embedding as $\mathbf{Z}_S$ in lexical feature space.

Moreover, we consider edge weights in the semantic graph channel and infuse TF-IDF values in the $A_{i,j}$ as the edge weights. The TF is the number of times w_i occurring in s_j, and the IDF is made as the inverse function of the out-degree of w_i.

2.0.2 Node Centrality Feature Encoding Channel. For graph summarization, keywords (excluding stop words) appear in many sentences, and key sentences share many keywords. The centrality is to quantify the importance of the nodes that are the closest to all other nodes in the network.

In MuchSUM, we develop a centrality feature encoding channel to learn the sentence importance. The centrality feature X_c of graph nodes is $\mathbf{X}_c = \mathbf{H}_c^{(0)}$ and $\mathbf{H}_c^{(0)} \in \mathbb{R}^{n \times d}$ is initialized by four typical centrality features[1]: `degree` centrality, `Katz` centrality, `closeness` centrality and `load` centrality. The dimension of each

[1]The detail introduction are shown in https://networkx.org/documentation/stable/reference/algorithms/centrality.html

feature embedding occupies $\mathbb{R}^{n\times(d/4)}$. We also use tools of feature discretization[2] to decompose each feature into a set of bins, here equally distributed in width. The discrete feature values are then one-hot encoded for each centrality and will be appended as the final $H_c^{(0)}$. The centrality channel is defined as:

$$\mathbf{Z}_c^{(k+1)} = Relu\left(\widetilde{\mathbf{D}}^{-\frac{1}{2}}\widetilde{\mathbf{A}}\widetilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{Z}_i^{(k-1)}\mathbf{W}_c^{(k)}\right), \tag{2}$$

where $\mathbf{W}_c^{(k)} \in \mathbb{R}^{d_{k-1}\times d_k}$ is parameter matrix of the centrality channel. $\mathbf{Z}_c^{(0)} = \mathbf{X}_c$ is the input representation matrix of the model. The last layer output embedding is denoted as $\mathbf{Z}_C$.

Moreover, we also consider the edge weights calculated by the *betweenness* centrality for edges. We then infuse the edge weights into the $A_{i,j}$ as the weight matrix.

2.0.3 Node Position Feature Encoding Channel. It has been proved by Ke et al. [7] that the design of a separate [3] position encoding can remove the randomness in token-to-position or position-to-token correlations in graphs. Inspired by their works, we compute the positional correlation in a separate channel with its own parameterizations. The position feature $\mathbf{X}_p = \mathbf{H}_p^{(0)}$ is to be learned to obtain the sequential nature of tokens in the original text. $\mathbf{H}_p^{(0)} \in \mathbb{R}^{n\times d}$ is initialized by absolute position features, inspired by the BERT's position embeddings. Further, the position channel is defined as:

$$\mathbf{Z}_p^{(k+1)} = Relu\left(\widetilde{\mathbf{D}}^{-\frac{1}{2}}\widetilde{\mathbf{A}}\widetilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{Z}_p^{(k-1)}\mathbf{W}_p^{(k)}\right), \tag{3}$$

where $\mathbf{W}_p^{(k)} \in \mathbb{R}^{d_{k-1}\times d_k}$ is parameter matrix of the position channel. $\mathbf{Z}_p^{(0)} = \mathbf{X}_p$ is the input representation matrix of the model. The last layer output embedding is denoted as $\mathbf{Z}_P$.

Besides, inspired by the relative position embeddings proposed by Shaw et al. [21], we normalize the relative position as the relative position weights into the $A_{i,j}$ to capture relative position differences between nodes:

$$A_{i,j} = clip(j-i,k)/\sum\nolimits_j clip(j-i,k), \tag{4}$$

$$clip(x,k) = max(k, min(2k, x+k)). \tag{5}$$

Here, we obtain the weight $A_{i,j}$ of relative position between node i and j. By using the node relative position weights as edge weight in the adjacency matrix, this channel expects the neighboring positions are embedded closer than the faraway ones.

2.0.4 Common Convolution Module. The common convolution channel with parameter sharing is to get the common features shared by the three specific channels since these channels are under the same topology of the word-sentence heterogeneous graph. Specifically, the lexical feature embedding Z_{ms} from lexical graph channel is transformed into the common convolution channel:

$$\mathbf{Z}_{ms}^{(k+1)} = Relu\left(\widetilde{\mathbf{D}}^{-\frac{1}{2}}\widetilde{\mathbf{A}}\widetilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{Z}_{ms}^{(k-1)}\mathbf{W}_m^{(k)}\right), \tag{6}$$

where, Z_{ms}^{k-1} is the lexical feature embedding of the $k-1$ layer and $Z_{ms}^{(0)} = X_s$. Similarly, the centrality feature and position feature embedding can be calculated in the same way:

$$\mathbf{Z}_{mc}^{(k+1)} = Relu\left(\widetilde{\mathbf{D}}^{-\frac{1}{2}}\widetilde{\mathbf{A}}\widetilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{Z}_{mc}^{(k-1)}\mathbf{W}_m^{(k)}\right), \tag{7}$$

$$\mathbf{Z}_{mp}^{(k+1)} = Relu\left(\widetilde{\mathbf{D}}^{-\frac{1}{2}}\widetilde{\mathbf{A}}\widetilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{Z}_{mp}^{(k-1)}\mathbf{W}_m^{(k)}\right). \tag{8}$$

Then, the final output of common embedding is:

$$Z_M = (Z_{ms} + Z_{mc} + Z_{mp})/3. \tag{9}$$

2.0.5 Graph Representation Readout. Now we have three specific representations $\mathbf{Z}_S$, $\mathbf{Z}_C$, $\mathbf{Z}_P$, and one common representations $\mathbf{Z}_M$. Then we read out these representations and fuse them into the representations of final sentences for the sentence binary classification. We use the attention mechanism to fuse these representations adaptively and then to obtain the final sentence representations $\mathbf{Z}$:

$$\mathbf{Z} = \mathbf{a}_S \cdot \mathbf{Z}_S + \mathbf{a}_C \cdot \mathbf{Z}_C + \mathbf{a}_P \cdot \mathbf{Z}_P + \mathbf{a}_M \cdot \mathbf{Z}_M, \tag{10}$$

where $\mathbf{a}_S, \mathbf{a}_C, \mathbf{a}_P, \mathbf{a}_M \in \mathbb{R}^{n\times 1}$ are three learnable attention values of n sentence-related nodes with representations $\mathbf{Z}_S$, $\mathbf{Z}_C$, $\mathbf{Z}_P$, and one common representations $\mathbf{Z}_M$, respectively.

2.0.6 Optimization. We use the disparity constraint and consistency constraint reserved from AM-GCN [27]. The disparity constraint is used to ensure the independence between $\mathbf{Z}_S$ and $\mathbf{Z}_{ms}$, $\mathbf{Z}_C$ and $\mathbf{Z}_{mc}$, as well as $\mathbf{Z}_P$ and $\mathbf{Z}_{mp}$:

$$\mathcal{L}_d = HSIC(\mathbf{Z}_S, \mathbf{Z}_{ms}) + HSIC(\mathbf{Z}_C, \mathbf{Z}_{mc}) + HSIC(\mathbf{Z}_P, \mathbf{Z}_{mp}), \tag{11}$$

where the function of $HSIC$ is the Hilbert-Schmidt Independence Criterion [22] which is a kernel method to measure the statistical dependence between two variables.

The consistency constraint is designed to enhance the commonality among these embedding matrices $\mathbf{Z}_{ms}$, $\mathbf{Z}_{mc}$ and $\mathbf{Z}_{mp}$. Firstly, they are normalized and then transformed into similarity matrices separately. Three corresponding similarity matrices are S_S, S_C and S_P. $\mathrm{S}_S = \mathbf{Z}_{ms} \cdot \mathbf{Z}_{ms}^T$, $\mathrm{S}_C = \mathbf{Z}_{mc} \cdot \mathbf{Z}_{mc}^T$ and $\mathrm{S}_P = \mathbf{Z}_{mp} \cdot \mathbf{Z}_{mp}^T$. Then, the three similarity matrices are optimized by L_2-normalization-based consistency constraint:

$$\mathcal{L}_c = \|\mathrm{S}_S - \mathrm{S}_C\|_F^2 + \|\mathrm{S}_C - \mathrm{S}_P\|_F^2 + \|\mathrm{S}_P - \mathrm{S}_S\|_F^2. \tag{12}$$

The consistency constraint implies that the two similarity matrices are similar. We use the final sentence representation $\mathbf{Z}$ in Eq.(10) for the sentence binary classification. The predication layer is a sigmoid classifier:

$$\hat{\mathrm{Y}} = \sigma(\mathbf{W} \cdot \mathbf{Z} + \mathbf{b}). \tag{13}$$

The classification loss is a cross-entropy loss:

$$\mathcal{L}_y = -\sum_{l\in L}\sum_{i\in n}\sum_{c\in C} \mathrm{Y}_l ln\hat{\mathrm{Y}}, \tag{14}$$

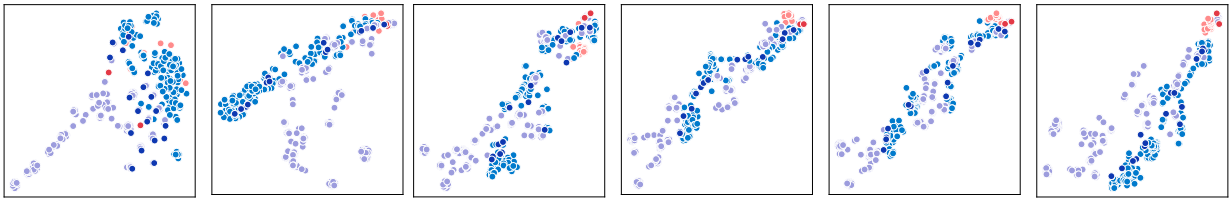
where $l \in L$ is a summarization graph of the training set and the $i \in n$ is the i-th sentence node in the graph. $c \in C$ is the real label of the i-th sentence node. Now we have the following overall objective function by combining the node classification task:

$$\mathcal{L} = \mathcal{L}_y + \alpha\mathcal{L}_d + \beta\mathcal{L}_c, \tag{15}$$

where α and β are parameters of the consistency and disparity constraint terms.

[2] https://scikit-learn.org/0.20/modules/generated/sklearn.preprocessing.KBinsDiscretizer.html

[3] It should be noted that the position embedding in the BERT-based lexical encoding channel is used to distinguish the lexical semantics of the same word in different positions. The separate position embedding here differentiates the position features.



(a) 1st Epoch (b) 2nd Epoch (c) 3rd Epoch (d) 4th Epoch (e) 5th Epoch (f) 6th Epoch

Figure 2: Visualization of the learned node embeddings in testing each epoch. Red nodes are words (light) and sentence (heavy) in labels of summary, while blue nodes are related to words (light) and sentences (heavy) in non-summaries. Purple nodes are words shared by sentences between summaries and non-summaries.

Table 1: Results of our proposed models against released salient BERT-based extractive summarization systems[4] on CNN/DailyMail test sets using ROUGE F1. We obtain ROUGE scores using the official `ROUGE-1.5.5.pl` script. ° means our carefully re-implementation using their source code to replace their Glove [18] embeddings into BERT-initialized embeddings. ♠ means BERT-initialized graph-based models.

ID	Model	ROUGE-1	ROUGE-2	ROUGE-L
1	Oracle [11]	55.61	32.84	51.88
2	LEAD-3 [20]	40.42	17.62	36.67
3	TextRank (TF-IDF) [30]	33.22	11.80	29.60
4	TextRank (BERT) [30]	30.80	9.60	27.40
5	PacSum (TF-IDF) [34]	39.20	16.30	35.30
6	PacSum (BERT) [34]	40.70	17.80	36.90
7	PnBERT [36]	42.39	19.51	38.69
8	PnBERT w. RL	42.69	19.60	38.85
9	BERT [29]	43.07	19.94	39.44
10	HiBERT [33]	42.37	19.95	38.83
11	Multi-GraS [6]	43.16	20.14	39.49
12	Multi-GraS °♠ (BERT)	43.22	20.15	39.50
13	BERTSumExt [11]	43.25	20.24	39.63
14	HSG [25]	42.95	19.76	39.23
15	HSG °♠ (BERT)	43.30	19.81	39.35
16	DiscoBERT ♠ [29]	43.77	20.85	40.67
17	MuchSUM (ours)	43.85	20.93	40.72

3 EXPERIMENTS SETTINGS

Summarization Dataset. We conduct experiments on the classical summarization CNN/Dailymail dataset [4]. We use the standard splits of Hermann et al. [4] for training, validation, and testing (90,266/1,220/1,093 CNN documents and 196,961/12,148/10,397 DailyMail documents). For data prepossessing, we split sentences with the Stanford CoreNLP toolkit [12] and then preprocess the dataset following the same setting as See et al. [20] have done. The ground truth labels, which we call ORACLE, are extracted with the greedy approach released by Wang et al. [25].

Implementation Details. We train our extractive model on a Tesla V100-PCIE-32GB GPU card. The top-3 checkpoints are trained based on the evaluation loss on the validation set and the averaged results on the test set were reported. We use the vocabulary of BERT with 30522 tokens. We filter stop words and punctuations when creating word-related nodes. The filtered source text keeps the original sequence of tokens used to represent the graph's word-related and sentence-related nodes. The input bipartite word-sentence heterogeneous graph[5] is truncated with 512 token nodes. The linguistic feature channel of MuchSUM uses BERT-base-uncased for initialization with blocks N=12, the hidden size H=768, and the number of self-attention heads A=12. In the centrality channel of MuchSUM, the feature score is decomposed into 10 bins, here equally distributed in width. The position graph is also initialized with 768-dimensional feature embeddings. During training, we use a batch size of 32 and apply Adam optimizer with β_1 =0.9, and β_2 =0.999. Our learning rate schedule follows Liu and Lapata [11], Vaswani et al. [24] with warming-up: $lr = 2e^{-3} \cdot min(step^{-0.5}, step \cdot warmup^{-1.5})$ and the step of warmup is 10,000. We set the dropout with probability p=0.1 in all graph layers to prevent overfitting. The best scores of parameters α and β are 3e-5 and 5e-8, respectively.

[5]Available at https://github.com/RingBDStack/MuchSum/SumGraph.

Table 2: Ablation study on CNN/DailyMail test sets.

Model	ROUGE-1	ROUGE-2	ROUGE-L
MuchSUM w/o $\mathcal{G}_S$	40.26↓3.59	18.76↓2.17	36.14↓4.58
MuchSUM w/o $\mathcal{G}_P$	43.70↓0.15	20.56↓0.37	40.53↓0.19
MuchSUM w/o $\mathcal{G}_C$	43.67↓0.18	20.77↓0.16	40.59↓0.13
MuchSUM w/o Common Graph Channel	43.62↓0.23	20.80↓0.13	40.50↓0.22
MuchSUM w/o Weighted Graph	43.72↓0.13	20.71↓0.22	40.43↓0.29

4 EXPERIMENTAL RESULTS.

The experimental results of extractive summarization on the dataset of CNN/DailyMail are presented in Table 1. Row 1 is the sentence-based oracle. Row 2-6 list unsupervised baseline models. Row 7-15 are supervised extractive models, all of which are BERT-based variants. Among them, HSG [25] and DiscoBERT [29] are graph-based methods with BERT initialization, and HSG has the same bipartite word-sentence graphs as ours.

Compared with the models in rows 2-6, our proposed neural graph-based summarization surpasses all unsupervised methods. Compared with the models in rows 7-15, our MuchSUM substantially outperforms all the pure BERT-based methods and BERT-initialized summarization graph models with a noticeable margin on ROUGE performance. In particular, compared with HSG (BERT) with the same bipartite word-sentence graph, our graph model has distinct improvements, showing the effectiveness of MuchSUM, which allows the model to better capture sentence relations by modeling other summary-worthy features. To further demonstrate the effectiveness of our proposed model in distinguishing summary

[5]We have verified these systems by their released source code. It should be noted that there exist some graph-based extractive models, like HAHSum [5] and FS^3 [3]. Despite our best efforts, we could not examine or re-produce these models since their implementations (graph construction results and summarization model) are not available in the public domain.

sentences, we use the output embedding on the last layer of MuchSUM before softmax and plot the learned embedding of the test set using t-SNE [23]. Apparently, the learned node embedding has an apparent intra-class similarity and distinct boundaries among summary and non-summary words and sentences classes.

To better understand the contribution of different graph channels to the performance, we conducted an ablation study, and the results are shown in Table 2. First, we have separately removed three specific graph channels and the edge weights among three graph convolution layers. Significantly, the semantics graph channel is the most important than the importance graph channel and the position graph channel. Besides, the common convolutional graph channel plays an essential role in enlarging the advantage of MuchSUM because there are 'common' properties in the summary sentences. Moreover, the introduction of weighted edge features also shows their effectiveness for distinguishing summary sentences.

4.1 Conclusions

To extract a good summary from a document, we propose a novel multi-channel graph-based neural network to incorporate sentence semantics, sentence importance, and sentence position and their combinations in graphs. These features are fused adaptively for extractive summarization. Our model outperforms salient neural graph models initialized by BERT-based language models.

5 ACKNOWLEDGMENTS

This work is supported in part by the NSFC through grant U20B2053.

REFERENCES

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT*. Association for Computational Linguistics, 4171–4186.

[2] Günes Erkan and Dragomir R. Radev. 2004. LexRank: Graph-based Lexical Centrality as Salience in Text Summarization. *J. Artif. Intell. Res.* 22 (2004), 457–479.

[3] Yong Guan, Shaoru Guo, Ru Li, Xiaoli Li, and Hongye Tan. 2021. Frame Semantic-Enhanced Sentence Modeling for Sentence-level Extractive Text Summarization. In *EMNLP*. Association for Computational Linguistics, 4045–4052.

[4] Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching Machines to Read and Comprehend. In *NIPs*. 1693–1701.

[5] Ruipeng Jia, Yanan Cao, Hengzhu Tang, Fang Fang, Cong Cao, and Shi Wang. 2020. Neural Extractive Summarization with Hierarchical Attentive Heterogeneous Graph Network. In *EMNLP*. Association for Computational Linguistics, 3622–3631.

[6] Baoyu Jing, Zeyu You, Tao Yang, Wei Fan, and Hanghang Tong. 2021. Multiplex Graph Neural Network for Extractive Text Summarization. In *EMNLP*. Association for Computational Linguistics, 133–139.

[7] Guolin Ke, Di He, and Tie-Yan Liu. 2021. Rethinking Positional Encoding in Language Pre-training. In *ICLR*. OpenReview.net.

[8] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*. OpenReview.net.

[9] Logan Lebanoff, Kaiqiang Song, Franck Dernoncourt, Doo Soon Kim, Seokhwan Kim, Walter Chang, and Fei Liu. 2019. Scoring Sentence Singletons and Pairs for Abstractive Summarization. In *ACL, Volume 1: Long Papers*. Association for Computational Linguistics, 2175–2189.

[10] Bohan Li, Hao Zhou, Junxian He, Mingxuan Wang, Yiming Yang, and Lei Li. 2020. On the Sentence Embeddings from Pre-trained Language Models. In *EMNLP*. Association for Computational Linguistics, 9119–9130.

[11] Yang Liu and Mirella Lapata. 2019. Text Summarization with Pretrained Encoders. In *EMNLP-IJCNLP*. Association for Computational Linguistics, 3728–3738.

[12] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *ACL, System Demonstrations*. The Association for Computer Linguistics, 55–60.

[13] Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing Order into Text. In *EMNLP*. Association for Computational Linguistics, 404–411.

[14] Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. SummaRuNNer: A Recurrent Neural Network Based Sequence Model for Extractive Summarization of Documents. In *AAAI*. AAAI Press, 3075–3081.

[15] Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Ranking Sentences for Extractive Summarization with Reinforcement Learning. In *NAACL-HLT, Volume 1 (Long Papers)*. Association for Computational Linguistics, 1747–1759.

[16] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. *The PageRank Citation Ranking: Bringing Order to the Web*. Technical Report. Stanford InfoLab.

[17] Hao Peng, Jianxin Li, Qiran Gong, Yangqiu Song, Yuanxing Ning, Kunfeng Lai, and Philip S. Yu. 2019. Fine-grained Event Categorization with Heterogeneous Graph Convolutional Networks. In *IJCAI*, Sarit Kraus (Ed.). ijcai.org, 3238–3245.

[18] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global Vectors for Word Representation. In *EMNLP*. ACL, 1532–1543.

[19] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *EMNLP-IJCNLP*, Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (Eds.). Association for Computational Linguistics, 3980–3990.

[20] Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get To The Point: Summarization with Pointer-Generator Networks. In *ACL, Volume 1: Long Papers*. Association for Computational Linguistics, 1073–1083.

[21] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-Attention with Relative Position Representations. In *NAACL-HLT, Volume 2, Short Papers*, Marilyn A. Walker, Heng Ji, and Amanda Stent (Eds.). Association for Computational Linguistics, 464–468.

[22] Le Song, Alexander J. Smola, Arthur Gretton, Karsten M. Borgwardt, and Justin Bedo. 2007. Supervised feature selection via dependence estimation. In *(ICML (ACM International Conference Proceeding Series)*, Vol. 227. ACM, 823–830.

[23] Hinton G Van der Maaten L. 2008. Visualizing Data using T-SNE. 9, 11 (2008).

[24] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *NIPs*. 5998–6008.

[25] Danqing Wang, Pengfei Liu, Yining Zheng, Xipeng Qiu, and Xuanjing Huang. 2020. Heterogeneous Graph Neural Networks for Extractive Document Summarization. In *ACL*. Association for Computational Linguistics, 6209–6219.

[26] Danqing Wang, Pengfei Liu, Ming Zhong, Jie Fu, Xipeng Qiu, and Xuanjing Huang. 2019. Exploring Domain Shift in Extractive Text Summarization. *CoRR* abs/1908.11664 (2019).

[27] Xiao Wang, Meiqi Zhu, Deyu Bo, Peng Cui, Chuan Shi, and Jian Pei. 2020. AM-GCN: Adaptive Multi-channel Graph Convolutional Networks. In *KDD*. ACM, 1243–1253.

[28] Jiacheng Xu, Zhe Gan, Yu Cheng, and Jingjing Liu. 2019. Discourse-Aware Neural Extractive Model for Text Summarization. *CoRR* abs/1910.14142 (2019).

[29] Jiacheng Xu, Zhe Gan, Yu Cheng, and Jingjing Liu. 2020. Discourse-Aware Neural Extractive Text Summarization. In *ACL*. Association for Computational Linguistics, 5021–5031.

[30] Shusheng Xu, Xingxing Zhang, Yi Wu, Furu Wei, and Ming Zhou. 2020. Unsupervised Extractive Summarization by Pre-training Hierarchical Transformers. In *EMNLP (Findings of ACL)*, Vol. EMNLP 2020. Association for Computational Linguistics, 1784–1795.

[31] Yuanmeng Yan, Rumei Li, Sirui Wang, Fuzheng Zhang, Wei Wu, and Weiran Xu. 2021. ConSERT: A Contrastive Framework for Self-Supervised Sentence Representation Transfer. In *ACL/IJCNLP,Volume 1: Long Papers*. Association for Computational Linguistics, 5065–5075.

[32] Michihiro Yasunaga, Rui Zhang, Kshitijh Meelu, Ayush Pareek, Krishnan Srinivasan, and Dragomir R. Radev. 2017. Graph-based Neural Multi-Document Summarization. In *CoNLL*. Association for Computational Linguistics, 452–462.

[33] Xingxing Zhang, Furu Wei, and Ming Zhou. 2019. HIBERT: Document Level Pre-training of Hierarchical Bidirectional Transformers for Document Summarization. In *ACL , Volume 1: Long Papers*. Association for Computational Linguistics, 5059–5069.

[34] Hao Zheng and Mirella Lapata. 2019. Sentence Centrality Revisited for Unsupervised Summarization. In *ACL , Volume 1: Long Papers*. Association for Computational Linguistics, 6236–6247.

[35] Ming Zhong, Pengfei Liu, Yiran Chen, Danqing Wang, Xipeng Qiu, and Xuanjing Huang. 2020. Extractive Summarization as Text Matching. In *ACL*. Association for Computational Linguistics, 6197–6208.

[36] Ming Zhong, Pengfei Liu, Danqing Wang, Xipeng Qiu, and Xuanjing Huang. 2019. Searching for Effective Neural Extractive Summarization: What Works and What's Next. In *ACL , Volume 1: Long Papers*. Association for Computational Linguistics, 1049–1058.

[37] Qingyu Zhou, Nan Yang, Furu Wei, Shaohan Huang, Ming Zhou, and Tiejun Zhao. 2018. Neural Document Summarization by Jointly Learning to Score and Select Sentences. In *ACL , Volume 1: Long Papers*. Association for Computational Linguistics, 654–663.