Leveraging Graph to Improve Abstractive Multi-Document Summarization

Wei Li[1], Xinyan Xiao[1]*, Jiachen Liu[1], Hua Wu[1], Haifeng Wang[1], Junping Du[2]
[1]Baidu Inc., Beijing, China
[2]Beijing University of Posts and Telecommunications
{liwei85,xiaoxinyan,liujiachen,wu_hua,wanghaifeng}@baidu.com
junpingdu@126.com

Abstract

Graphs that capture relations between textual units have great benefits for detecting salient information from multiple documents and generating overall coherent summaries. In this paper, we develop a neural abstractive multi-document summarization (MDS) model which can leverage well-known graph representations of documents such as similarity graph and discourse graph, to more effectively process multiple input documents and produce abstractive summaries. Our model utilizes graphs to encode documents in order to capture cross-document relations, which is crucial to summarizing long documents. Our model can also take advantage of graphs to guide the summary generation process, which is beneficial for generating coherent and concise summaries. Furthermore, pre-trained language models can be easily combined with our model, which further improve the summarization performance significantly. Empirical results on the WikiSum and MultiNews dataset show that the proposed architecture brings substantial improvements over several strong baselines.

1 Introduction

Multi-document summarization (MDS) brings great challenges to the widely used sequence-to-sequence (Seq2Seq) neural architecture as it requires effective representation of multiple input documents and content organization of long summaries. For MDS, different documents may contain the same content, include additional information, and present complementary or contradictory information (Radev, 2000). So different from single document summarization (SDS), cross-document links are very important in extracting salient information, detecting redundancy and generating overall coherent summaries for MDS. Graphs that capture

relations between textual units have great benefits to MDS, which can help generate more informative, concise and coherent summaries from multiple documents. Moreover, graphs can be easily constructed by representing text spans (e.g. sentences, paragraphs etc.) as graph nodes and the semantic links between them as edges. Graph representations of documents such as similarity graph based on lexical similarities (Erkan and Radev, 2004) and discourse graph based on discourse relations (Christensen et al., 2013), have been widely used in traditional graph-based extractive MDS models. However, they are not well studied by most abstractive approaches, especially the end-to-end neural approaches. Few work has studied the effectiveness of explicit graph representations on neural abstractive MDS.

In this paper, we develop a neural abstractive MDS model which can leverage explicit graph representations of documents to more effectively process multiple input documents and distill abstractive summaries. Our model augments the end-to-end neural architecture with the ability to incorporate well-established graphs into both the document representation and summary generation processes. Specifically, a graph-informed attention mechanism is developed to incorporate graphs into the document encoding process, which enables our model to capture richer cross-document relations. Furthermore, graphs are utilized to guide the summary generation process via a hierarchical graph attention mechanism, which takes advantage of the explicit graph structure to help organize the summary content. Benefiting from the graph modeling, our model can extract salient information from long documents and generate coherent summaries more effectively. We experiment with three types of graph representations, including similarity graph, topic graph and discourse graph, which all significantly improve the MDS performance.

---

*Corresponding author.

Additionally, our model is complementary to most pre-trained language models (LMs), like BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019) and XLNet (Yang et al., 2019b). They can be easily combined with our model to process much longer inputs. The combined model adopts the advantages of both our graph model and pre-trained LMs. Our experimental results show that our graph model significantly improves the performance of pre-trained LMs on MDS.

The contributions of our paper are as follows:

- Our work demonstrates the effectiveness of graph modeling in neural abstractive MDS. We show that explicit graph representations are beneficial for both document representation and summary generation.

- We propose an effective method to incorporate explicit graph representations into the neural architecture, and an effective method to combine pre-trained LMs with our graph model to process long inputs more effectively.

- Our model brings substantial improvements over several strong baselines on both Wiki-Sum and MultiNews dataset. We also report extensive analysis results, demonstrating that graph modeling enables our model process longer inputs with better performance, and graphs with richer relations are more beneficial for MDS.[1]

2 Related Work

2.1 Graph-based MDS

Most previous MDS approaches are extractive, which extract salient textual units from documents based on graph-based representations of sentences. Various ranking methods have been developed to rank textual units based on graphs to select most salient ones for inclusion in the final summary.

Erkan and Radev (2004) propose LexRank to compute sentence importance based on a lexical similarity graph of sentences. Mihalcea and Tarau (2004) propose a graph-based ranking model to extract salient sentences from documents. Wan (2008) further proposes to incorporate document-level information and sentence-to-document relations into the graph-based ranking process. A series of variants of the PageRank algorithm has been

further developed to compute the salience of textual units recursively based on various graph representations of documents (Wan and Xiao, 2009; Cai and Li, 2012). More recently, Yasunaga et al. (2017) propose a neural graph-based model for extractive MDS. An approximate discourse graph is constructed based on discourse markers and entity links. The salience of sentences is estimated using features from graph convolutional networks (Kipf and Welling, 2016). Yin et al. (2019) also propose a graph-based neural sentence ordering model, which utilizes entity linking graph to capture the global dependencies between sentences.

2.2 Abstractive MDS

Abstractive MDS approaches have met with limited success. Traditional approaches mainly include: sentence fusion-based (Banerjee et al., 2015; Filippova and Strube, 2008; Barzilay and McKeown, 2005; Barzilay, 2003), information extraction-based (Li, 2015; Pighin et al., 2014; Wang and Cardie, 2013; Genest and Lapalme, 2011; Li and Zhuge, 2019) and paraphrasing-based (Bing et al., 2015; Berg-Kirkpatrick et al., 2011; Cohn and Lapata, 2009). More recently, some researches parse the source text into AMR representation and then generate summary based on it (Liao et al., 2018).

Although neural abstractive models have achieved promising results on SDS (See et al., 2017; Paulus et al., 2018; Gehrmann et al., 2018; Celikyilmaz et al., 2018; Li et al., 2018a,b; Narayan et al., 2018; Yang et al., 2019a; Sharma et al., 2019; Perez-Beltrachini et al., 2019), it's not straightforward to extend them to MDS. Due to the lack of sufficient training data, earlier approaches try to simply transfer SDS model to MDS task (Lebanoff et al., 2018; Zhang et al., 2018; Baumel et al., 2018) or utilize unsupervised models relying on reconstruction objectives (Ma et al., 2016; Chu and Liu, 2019). Later, Liu et al. (2018) propose to construct a large scale MDS dataset (namely WikiSum) based on Wikipedia, and develop a Seq2Seq model by considering the multiple input documents as a concatenated flat sequence. Fan et al. (2019) further propose to construct a local knowledge graph from documents and then linearize the graph into a sequence to better sale Seq2Seq models to multi-document inputs. Fabbri et al. (2019) also introduce a middle-scale (about 50K) MDS news dataset (namely MultiNews), and propose an end-to-end model by incorporating traditional MMR-based
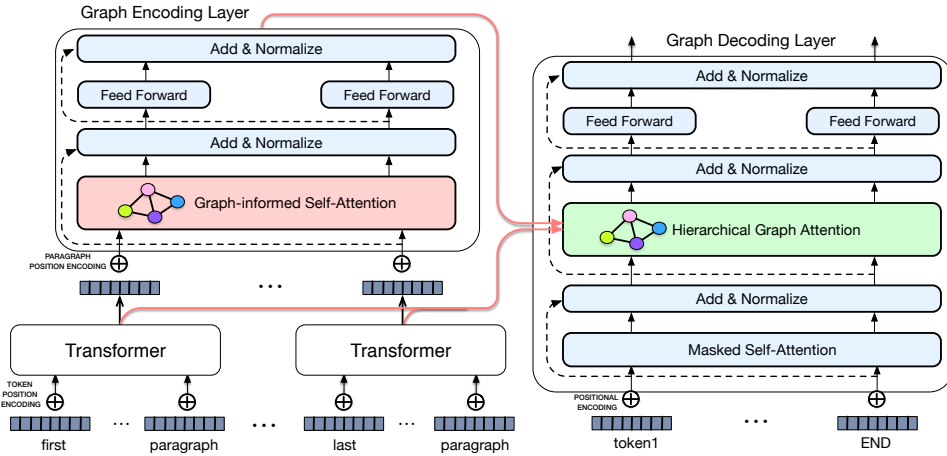
---

[1]Codes and results are in: `https://github.com/PaddlePaddle/Research/tree/master/NLP/ACL2020-GraphSum`

Figure 1: Illustration of our model, which follows the encoder-deocder architecture. The encoder is a stack of transformer layers and graph encoding layers, while the decoder is a stack of graph decoding layers. We incorporate explicit graph representations into both the graph encoding layers and graph decoding layers.

extractive model with a standard Seq2Seq model. The above Seq2Seq models haven't study the importance of cross-document relations and graph representations in MDS.

Most recently, Liu and Lapata (2019a) propose a hierarchical transformer model to utilize the hierarchical structure of documents. They propose to learn cross-document relations based on self-attention mechanism. They also propose to incorporate explicit graph representations into the model by simply replacing the attention weights with a graph matrix, however, it doesn't achieve obvious improvement according to their experiments. Our work is partly inspired by this work, but our approach is quite different from theirs. In contrast to their approach, we incorporate explicit graph representations into the encoding process via a graph-informed attention mechanism. Under the guidance of explicit relations in graphs, our model can learn better and richer cross-document relations, thus achieves significantly better performance.We also leverage the graph structure to guide the summary decoding process, which is beneficial for long summary generation. Additionally, we combine the advantages of pretrained LMs into our model.

2.3 Summarization with Pretrained LMs

Pretrained LMs (Peters et al., 2018; Radford et al.; Devlin et al., 2019; Dong et al., 2019; Sun et al., 2019) have recently emerged as a key technology for achieving impressive improvements in a wide variety of natural language tasks, including both language understanding and language generation (Edunov et al., 2019; Rothe et al., 2019). Liu and

Lapata (2019b) attempt to incorporate pre-trained BERT encoder into SDS model and achieves significant improvements. Dong et al. (2019) further propose a unified LM for both language understanding and language generation tasks, which achieves state-of-the-art results on several generation tasks including SDS. In this work, we propose an effective method to combine pretrained LMs with our graph model and make them be able to process much longer inputs effectively.

3 Model Description

In order to process long source documents more effectively, we follow Liu and Lapata (2019a) in splitting source documents into multiple paragraphs by line-breaks. Then the graph representation of documents is constructed over paragraphs. For example, a similarity graph can be built based on cosine similarities between tf-idf representations of paragraphs. Let $\mathbb{G}$ denotes a graph representation matrix of the input documents, where $\mathbb{G}[i][j]$ indicates the relation weights between paragraph P_i and P_j. Formally, the task is to generate the summary S of the document collection given L input paragraphs $P_1, \ldots, P_L$ and their graph representation $\mathbb{G}$.

Our model is illustrated in Figure 1, which follows the encoder-decoder architecture (Bahdanau et al., 2015). The encoder is composed of several token-level transformer encoding layers and paragraph-level graph encoding layers which can be stacked freely. The transformer encoding layer follows the Transformer architecture introduced in Vaswani et al. (2017), encoding contextual information for tokens within each paragraph. The

graph encoding layer extends the Transformer architecture with a graph attention mechanism to incorporate explicit graph representations into the encoding process. Similarly, the decoder is composed of a stack of graph decoding layers. They extend the Transformer with a hierarchical graph attention mechanism to utilize explicit graph structure to guide the summary decoding process. In the following, we will focus on the graph encoding layer and graph decoding layer of our model.

3.1 Graph Encoding Layer

As shown in Figure 1, based on the output of the token-level transformer encoding layers, the graph encoding layer is used to encode all documents globally. Most existing neural work only utilizes attention mechanism to learn latent graph representations of documents where the graph edges are attention weights (Liu and Lapata, 2019a; Niculae et al., 2018; Fernandes et al., 2018). However, much work in traditional MDS has shown that explicit graph representations are very beneficial to MDS. Different types of graphs capture different kinds of semantic relations (e.g. lexical relations or discourse relations), which can help the model focus on different facets of the summarization task. In this work, we propose to incorporate explicit graph representations into the neural encoding process via a graph-informed attention mechanism. It takes advantage of the explicit relations in graphs to learn better inter-paragraph relations. Each paragraph can collect information from other related paragraphs to capture global information from the whole input.

Graph-informed Self-attention The graph-informed self-attention extends the self-attention mechanism to consider the pairwise relations in explicit graph representations. Let x_i^{l-1} denotes the output of the $(l-1)$-th graph encoding layer for paragraph P_i, where x_i^0 is just the input paragraph vector. For each paragraph P_i, the context representation u_i can be computed as a weighted sum of linearly transformed paragraph vectors:

$$
\begin{aligned}
\alpha_{ij} &= softmax(e_{ij} + \Re_{ij}) \\
e_{ij} &= \frac{(x_i^{l-1}W_Q)(x_j^{l-1}W_K)^T}{\sqrt{d_{head}}} \\
u_i &= \sum_{j=1}^{L} \alpha_{ij}(x_j^{l-1}W_V)
\end{aligned}
\tag{1}
$$

where W_K, W_Q and $W_V \in \mathbb{R}^{d*d}$ are parameter weights. e_{tj} denotes the latent relation weight between paragraph P_i and P_j. The main difference of our graph-informed self-attention is the additional pairwise relation bias $\Re_{ij}$, which is computed as a Gaussian bias of the weights of graph representation matrix $\mathbb{G}$:

$$
\Re_{ij} = -\frac{(1 - \mathbb{G}[i][j])^2}{2\sigma^2}
\tag{2}
$$

where σ denotes the standard deviation that represents the influence intensity of the graph structure. We set it empirically by tuning on the development dataset. The gaussian bias $R_{ij} \in (-inf, 0]$ measures the tightness between the paragraphs P_i and P_j. Due to the exponential operation in softmax function, the gaussian bias approximates to multiply the latent attention distribution by a weight $\in (0, 1]$.

In our graph-attention mechanism, the term e_{ij} in Equation 1 keeps the ability to model latent dependencies between any two paragraphs, and the term $\Re_{ij}$ incorporates explicit graph representations as prior constraints into the encoding process. This way, our model can learn better and richer inter-paragraph relations to obtain more informative paragraph representations.

Then, a two-layer feed-forward network with ReLU activation function and a high-way layer normalization are applied to obtain the vector of each paragraph x_i^l:

$$
\begin{aligned}
p_i^l &= W_{o2}ReLU(W_{o1}(u_i + x_i^{l-1})) \\
x_i^l &= LayerNorm(p_i^l + x_i^{l-1})
\end{aligned}
\tag{3}
$$

where $W_{o1} \in \mathbb{R}^{d_{ff}*d}$ and $W_{o2} \in \mathbb{R}^{d*d_{ff}}$ are learnable parameters, d_{ff} is the hidden size of the feed-forward layer.

3.2 Graph Decoding Layer

Graphs can also contribute to the summary generation process. The relations between textual units can help to generate more coherent or concise summaries. For example, Christensen et al. (2013) propose to leverage an approximate discourse graph to help generate coherent extractive summaries. The discourse relations between sentences are used to help order summary sentences. In this work, we propose to incorporate explicit graph structure into the end-to-end summary decoding process. Graph edges are used to guide the summary generation process via a hierarchical graph attention, which

is composed by a *global* graph attention and a *local* normalized attention. As other components in the graph decoding layer are similar to the Transformer architecture, we focus on the extension of hierarchical graph attention.

Global Graph Attention The global graph attention is developed to capture the paragraph-level context information in the encoder part. Different from the context attention in Transformer, we utilize the explicit graph structure to regularize the attention distributions so that graph representations of documents can be used to guide the summary generation process.

Let y_t^{l-1} denotes the output of the $(l-1)$-th graph decoding layer for the t-th token in the summary. We assume that each token will align with several related paragraphs and one of them is at the central position. Since the prediction of the central position depends on the corresponding query token, we apply a feed-forward network to transform y_t^{l-1} into a positional hidden state, which is then mapped into a scalar s_t by a linear projection:

$$s_t = L * sigmoid(U_p^T tanh(W_p y_t^{l-1})) \quad (4)$$

where $W_p \in \mathbb{R}^{d*d}$ and $U_p \in \mathbb{R}^d$ denote weight matrix. s_t indicates the central position of paragraphs that are mapped by the t-th summary token. With the central position, other paragraphs are determined by the graph structure. Then an attention distribution over all paragraphs under the regularization of the graph structure can be obtained:

$$\beta_{tj} = softmax(e_{tj} - \frac{(1 - \mathbb{G}[s_t][j])^2}{2\sigma^2}) \quad (5)$$

where e_{tj} denotes the attention weight between token vector y_t^{l-1} and paragraph vector x_j, which is computed similarly to Equation 1. The global context vector can be obtained as a weighted sum of paragraph vectors: $g_t = \sum_{j=1}^L \beta_{tj} x_j$

In our decoder, graphs are also modeled as a Gaussian bias. Different from the encoder, a central mapping position is firstly decided and then graph relations corresponding to that position are used to regularize the attention distributions β_{tj}. This way, the relations in graphs are used to help align the information between source input and summary output globally, thus guiding the summary decoding process.

Local Normalized Attention Then, a local normalized attention is developed to capture the token-level context information within each paragraph.

The local attention is applied to each paragraph independently and normalized by the global graph attention. This way, our model can process longer inputs effectively.

Let $\gamma_{t,ji}$ denotes the local attention distributions of the t-th summary token over the i-th token in the j-th input paragraph, the normalized attention is computed by:

$$\hat{\gamma}_{t,ji} = \gamma_{t,ji} \beta_{tj} \quad (6)$$

and the local context vector can be computed as a weighted sum of token vectors in all paragraphs: $l_t = \sum_{j=1}^L \sum_{k=1}^n \hat{\gamma}_{t,ji} x_{ji}$

Finally, the output of the hierarchical graph attention component is computed by concatenating and linearly transforming the global and local context vector:

$$d_t = U_d^T [g_t, l_t] \quad (7)$$

where $U_d \in \mathbb{R}^{2d*d}$ is a weight matrix. Through combining the local and global context, the decoder can utilize the source information more effectively.

3.3 Combined with Pre-trained LMs

Our model can be easily combined with pre-trained LMs. Pre-trained LMs are mostly based on sequential architectures which are more effective on short text. For example, both BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019) are pre-trained with maximum 512 tokens. Liu and Lapata (2019b) propose to utilize BERT on single document summarization tasks. They truncate the input documents to 512 tokens on most tasks. However, thanks to the graph modeling, our model can process much longer inputs. A natural idea is to combine our graph model with pretrained LMs so as to combine the advantages of them. Specifically, the token-level transformer encoding layer of our model can be replaced by a pre-trained LM like BERT.

In order to take full advantage of both our graph model and pre-trained LMs, the input documents are formatted in the following way:

> [CLS] first paragraph [SEP] [CLS] second paragraph [SEP] ... [CLS] last paragraph [SEP]

Then they are encoded by a pre-trained LM, and the output vector of the "[CLS]" token is used as the vector of the corresponding paragraph. Finally, all paragraph vectors are fed into our graph encoder to learn global representations. Our graph decoder is further used to generate the summaries.

4 Experiments

4.1 Experimental Setup

Graph Representations We experiment with three well-established graph representations: similarity graph, topic graph and discourse graph. The similarity graph is built based on tf-idf cosine similarities between paragraphs to capture lexical relations. The topic graph is built based on LDA topic model (Blei et al., 2003) to capture topic relations between paragraphs. The edge weights are cosine similarities between the topic distributions of the paragraphs. The discourse graph is built to capture discourse relations based on discourse markers (e.g. however, moreover), co-reference and entity links as in Christensen et al. (2013). Other types of graphs can also be used in our model. In our experiments, if not explicitly stated, we use the *similarity graph* by default as it has been most widely used in previous work.

WikiSum Dataset We follow Liu et al. (2018) and Liu and Lapata (2019a) in treating the generation of lead Wikipedia sections as a MDS task. The source documents are reference webpages of the Wikipedia article and top 10 search results returned by Google, while the summary is the Wikipedia article's first section. As the source documents are very long and messy, they are split into multiple paragraphs by line-breaks. Further, the paragraphs are ranked by the title and top ranked paragraphs are selected as input for MDS systems. We directly utilize the ranking results from Liu and Lapata (2019a) and top-40 paragraphs are used as source input. The average length of each paragraph and the target summary are 70.1 tokens and 139.4 tokens, respectively. For the seq2seq baselines, paragraphs are concatenated as a sequence in the ranking order, and lead tokens are used as input. The dataset is split into 1,579,360 instances for training, 38,144 for validation and 38,205 for testing, similar to Liu and Lapata (2019a). We build similarity graph representations over paragraphs on this dataset.

MultiNews Dataset Proposed by Fabbri et al. (2019), MultiNews dataset consists of news articles and human-written summaries. The dataset comes from a diverse set of news sources (over 1500 sites). Different from the WikiSum dataset, MultiNews is more similar to the traditional MDS dataset such as DUC, but is much larger in scale. As in Fabbri et al. (2019), the dataset is split into 44,972 instances for

training, 5,622 for validation and 5,622 for testing. The average length of source documents and output summaries are 2103.5 tokens and 263.7 tokens, respectively. For the seq2seq baselines, we truncate N input documents to L tokens by taking the first L/N tokens from each source document. Then we concatenate the truncated source documents into a sequence by the original order. Similarly, for our graph model, the input documents are truncated to M paragraphs by taking the first M/N paragraphs from each source document. We build all three types of graph representations on this dataset to explore the influence of graph types on MDS.

Training Configuration We train all models with maximum likelihood estimation, and use label smoothing (Szegedy et al., 2016) with smoothing factor 0.1. The optimizer is Adam (Kingma and Ba, 2015) with learning rate 2, β_1=0.9 and β_2=0.998. We also apply learning rate warmup over the first 8,000 steps and decay as in (Vaswani et al., 2017). Gradient clipping with maximum gradient norm 2.0 is also utilized during training. All models are trained on 4 GPUs (Tesla V100) for 500,000 steps with gradient accumulation every four steps. We apply dropout with probability 0.1 before all linear layers in our models. The number of hidden units in our models is set as 256, the feed-forward hidden size is 1,024, and the number of heads is 8. The number of transformer encoding layers, graph encoding layers and graph decoding layers are set as 6, 2 and 8, respectively. The parameter σ is set as 2.0 after tuning on the validation dataset. During decoding, we use beam search with beam size 5 and length penalty with factor 0.6. Trigram blocking is used to reduce repetitions.

For the models with pretrained LMs, we apply different optimizers for the pretrained part and other parts as in (Liu and Lapata, 2019b). Two Adam optimizers with β_1=0.9 and β_2=0.999 are used for the pretrained part and other parts, respectively. The learning rate and warmup steps for the pretrained part are set as 0.002 and 20000, while 0.2 and 10000 for other parts. Other model configurations are in line with the corresponding pretrained LMs. We choose the base version of BERT, RoBERTa and XLNet in our experiments.

4.2 Evaluation Results

We evaluate our models on both the WikiSum and MultiNews datasets to validate the efficiency of them on different types of corpora. The summa-

Model	R-1	R-2	R-L
Lead	38.22	16.85	26.89
LexRank	36.12	11.67	22.52
FT	40.56	25.35	34.73
BERT+FT	41.49	25.73	35.59
XLNet+FT	40.85	25.29	35.20
RoBERTa+FT	42.05	27.00	36.56
T-DMCA	40.77	25.60	34.90
HT	41.53	26.52	35.76
GraphSum	42.63	27.70	36.97
GraphSum+RoBERTa	**42.99**	**27.83**	**37.36**

Table 1: Evaluation results on the WikiSum test set using ROUGE F_1. R-1, R-2 and R-L are abbreviations for ROUGE-1, ROUGE-2 and ROUGE-L, respectively.

Model	R-1	R-2	R-L
Lead	41.24	12.91	18.84
LexRank	41.01	12.69	18.00
PG-BRNN	43.77	15.38	20.84
HiMAP	44.17	16.05	21.38
FT	44.32	15.11	20.50
RoBERTa+FT	44.26	16.22	22.37
HT	42.36	15.27	22.08
GraphSum	45.02	16.69	22.50
G.S.(Similarity)+RoBERTa	45.93	17.33	23.33
G.S.(Topic)+RoBERTa	**46.07**	17.42	23.21
G.S.(Discourse)+RoBERTa	45.87	**17.56**	**23.39**

Table 2: Evaluation results on the MultiNews test set. We report the **summary-level** ROUGE-L value. The results of different graph types are also compared.

rization quality is evaluated using ROUGE F_1 (Lin and Och, 2004). We report unigram and bigram overlap (ROUGE-1 and ROUGE-2) between system summaries and gold references as a means of assessing informativeness, and the longest common subsequence (ROUGE-L[2]) as a means of accessing fluency.

Results on WikiSum Table 6 summarizes the evaluation results on the WikiSum dataset. Several strong extractive baselines and abstractive baselines are also evaluated and compared with our models. The first block in the table shows the results of extractive methods Lead and LexRank (Erkan and Radev, 2004). The second block shows the results of abstractive methods: (1) FT (Flat Transformer), a transformer-based encoder-decoder model on a flat token sequence; (2) T-DMCA, the best performing model of Liu et al. (2018); (3) HT (Hierarchical Transformer), a model with hierarchical transformer encoder and flat transformer decoder, proposed by Liu and Lapata (2019a). We report their results following Liu and Lapata (2019a). The last block shows the results of our models, which are feed with 30 paragraphs (about 2400 tokens) as input. The results show that all abstractive models outperform the extractive ones. Compared with FT, T-DMCA and HT, our model GraphSum achieves significant improvements on all three metrics, which demonstrates the effectiveness of our model.

Furthermore, we develop several strong base-

lines which combine the Flat Transformer with pre-trained LMs. We replace the encoder of FT by the base versions of pre-trained LMs, including BERT+FT, XLNet+FT and RoBERTa+FT. For them, the source input is truncated to 512 tokens [3]. The results show that the pre-trained LMs significantly improve the summarization performance. As RoBERTa boosts the summarization performance most significantly, we also combine it with our GraphSum model, namely GraphSum+RoBERTa [4]. The results show that GraphSum+RoBERTa further improves the summarization performance on all metrics, demonstrating that our graph model can be effectively combined with pre-trained LMs. The significant improvements over RoBERTa+FT also demonstrate the effectiveness of our graph modeling even with pre-trained LMs.

Results on MultiNews Table 7 summarizes the evaluation results on the MultiNews dataset. Similarly, the first block shows two popular extractive baselines, and the second block shows several strong abstractive baselines. We report the results of Lead, LexRank, PG-BRNN, HiMAP and FT following Fabbri et al. (2019). The last block shows the results of our models. The results show that our model GraphSum consistently outperforms all baselines, which further demonstrate the effectiveness of our model on different types of corpora. We also compare the performance of RoBERTa+FT and GraphSum+RoBERTa, which show that our model significantly improves all metrics.

---

[2]For fair comparison with previous work (Liu and Lapata, 2019a; Liu et al., 2018), we report the summary-level ROUGE-L results on both the two datasets. The sentence-level ROUGE-L results are reported in the Appendix.

[3]Longer inputs don't achieve obvious improvements.

[4]As XLNet and BERT achieve worse results than RoBERTa, we only report the results of GraphSum+RoBERTa

Len	Model	R-1	R-2	R-L
	HT	41.08	25.83	35.25
500	GraphSum	**41.55**	**26.24**	**35.59**
	▽	+0.47	+0.41	+0.34
	HT	41.41	26.46	35.79
800	GraphSum	**41.70**	**26.87**	**36.10**
	▽	+0.29	+0.41	+0.31
	HT	41.53	26.52	35.76
1600	GraphSum	**42.48**	**27.52**	**36.66**
	▽	+0.95	+1.00	+0.90
	HT	41.68	26.53	35.73
2400	GraphSum	**42.63**	**27.70**	**36.97**
	▽	+0.95	+1.17	+1.24
	HT	41.71	26.58	35.81
3000	GraphSum	**42.36**	**27.47**	**36.65**
	▽	+0.65	+0.89	+0.84

Table 3: Comparison of different input length on the WikiSum test set using ROUGE F_1. ▽ indicates the improvements of GraphSum over HT.

The above evaluation results on both WikiSum and MultiNews dataset both validate the effectiveness of our model. The proposed method to modeling graph in end-to-end neural model greatly improves the performance of MDS.

4.3 Model Analysis

We further analyze the effects of graph types and input length on our model, and validate the effectiveness of different components of our model by ablation studies.

Effects of Graph Types To study the effects of graph types, the results of GraphSum+RoBERTa with similarity graph, topic graph and discourse graph are compared on the MultiNews test set. The last block in Table 7 summarizes the comparison results, which show that the topic graph achieves better performance than similarity graph on ROUGE-1 and ROUGE-2, and the discourse graph achieves the best performance on ROUGE-2 and ROUGE-L. The results demonstrate that graphs with richer relations are more helpful to MDS.

Effects of Input Length Different lengths of input may affect the summarization performance seriously for Seq2Seq models, so most of them restrict the length of input and only feed the model with hundreds of lead tokens. As stated by Liu and Lapata (2019a), the FT model achieves the best performance when the input length is set to 800

Model	Rouge-1	Rouge-2	Rouge-L
GraphSum	**42.63**	**27.70**	**36.97**
w/o graph dec	42.06	27.13	36.33
w/o graph enc	40.61	25.90	35.26

Table 4: Ablation study on the WikiSum test set.

tokens, while longer input hurts performance. To explore the effectiveness of our GraphSum model on different length of input, we compare it with HT on 500, 800, 1600, 2400 and 3000 tokens of input respectively. Table 3 summarizes the comparison results, which show that our model outperforms HT on all length of input. More importantly, the advantages of our model on all three metrics tend to become larger as the input becomes longer. The results demonstrate that modeling graph in the end-to-end model enables our model process much longer inputs with better performance.

Ablation Study Table 4 summarizes the results of ablation studies aiming to validate the effectiveness of individual components. Our experiments confirmed that incorporating well-known graphs into the encoding process by our graph encoder (see w/o graph enc) and utilizing graphs to guide the summary decoding process by our graph decoder (w/o graph dec) are both beneficial for MDS.

4.4 Human Evaluation

In addition to the automatic evaluation, we also access system performance by human evaluation. We randomly select 50 test instances from the WikiSum test set and 50 from the MultiNews test set, and invite 3 annotators to access the outputs of different models independently. Annotators access the overall quality of summaries by ranking them taking into account the following criteria: (1) *Informativeness*: does the summary convey important facts of the input? (2) *Fluency*: is the summary fluent and grammatical? (3) *Succinctness*: does the summary avoid repeating information? Annotators are asked to ranking all systems from 1(best) to 5 (worst). Ranking could be the same for different systems if they have similar quality. For example, the ranking of five systems could be 1, 2, 2, 4, 5 or 1, 2, 3, 3, 3. All systems get score 2, 1, 0, -1, -2 for ranking 1, 2, 3, 4, 5 respectively. The rating of each system is computed by averaging the scores on all test instances.

Table 5 summarizes the comparison results of five systems. Both the percentage of ranking results

6239

Model	1	2	3	4	5	Rating
FT	0.18	0.21	0.23	0.16	0.22	-0.03*
R.B.+FT	0.32	0.22	0.17	0.19	0.10	0.49*
HT	0.21	0.32	0.12	0.15	0.20	0.19*
GraphSum	0.42	0.30	0.17	0.10	0.01	1.02
G.S.+R.B.	0.54	0.24	0.10	0.08	0.04	**1.16**

Table 5: Ranking results of system summaries by human evaluation. 1 is the best and 5 is the worst. The larger rating denotes better summary quality. R.B. and G.S. are the abbreviations of RoBERTa and GraphSum, respectively. $*$ indicates the overall ratings of the corresponding model are significantly (by Welch's t-test with $p < 0.01$) outperformed by our models GraphSum and GraphSum+RoBERTa.

and overall ratings are reported. The results demonstrate that GraphSum and GraphSum+RoBERTa are able to generate higher quality summaries than other models. Specifically, the summaries generated by GraphSum and GraphSum+RoBERTa usually contains more salient information, and are more fluent and concise than other models. The human evaluation results further validates the effectiveness of our proposed models.

5 Conclusion

In this paper we explore the importance of graph representations in MDS and propose to leverage graphs to improve the performance of neural abstractive MDS. Our proposed model is able to incorporate explicit graph representations into the document encoding process to capture richer relations within long inputs, and utilize explicit graph structure to guide the summary decoding process to generate more informative, fluent and concise summaries. We also propose an effective method to combine our model with pre-trained LMs, which further improves the performance of MDS significantly. Experimental results show that our model outperforms several strong baselines by a wide margin. In the future we would like to explore other more informative graph representations such as knowledge graphs, and apply them to further improve the summary quality.

Acknowledgments

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations*.

Siddhartha Banerjee, Prasenjit Mitra, and Kazunari Sugiyama. 2015. Multi-document abstractive summarization using ilp based multi-sentence compression. In *Proceedings of 24th International Joint Conference on Artificial Intelligence*.

Regina Barzilay. 2003. *Information Fusion for Multi-document Summarization: Paraphrasing and Generation*. Ph.D. thesis, Columbia University.

Regina Barzilay and Kathleen R McKeown. 2005. Sentence fusion for multidocument news summarization. *Computational Linguistics*, 31(3):297–328.

Tal Baumel, Matan Eyal, and Michael Elhadad. 2018. Query focused abstractive summarization: Incorporating query relevance, multi-document coverage, and summary length constraints into seq2seq models. *arXiv preprint arXiv:1801.07704*.

Taylor Berg-Kirkpatrick, Dan Gillick, and Dan Klein. 2011. Jointly learning to extract and compress. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, volume 1, pages 481–490. Association for Computational Linguistics.

Lidong Bing, Piji Li, Yi Liao, Wai Lam, Weiwei Guo, and Rebecca J Passonneau. 2015. Abstractive multi-document summarization via phrase selection and merging. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, volume 1, pages 1587–1597.

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

Xiaoyan Cai and Wenjie Li. 2012. Mutually reinforced manifold-ranking based relevance propagation model for query-focused multi-document summarization. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(5):1597–1607.

Asli Celikyilmaz, Antoine Bosselut, Xiaodong He, and Yejin Choi. 2018. Deep communicating agents for abstractive summarization. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, volume 1, pages 1662–1675.

Janara Christensen, Stephen Soderland, Oren Etzioni, et al. 2013. Towards coherent multi-document summarization. In *Proceedings of the 2013 conference of the North American chapter of the association for*

computational linguistics: Human language technologies, pages 1163–1173.

Eric Chu and Peter Liu. 2019. Meansum: a neural model for unsupervised multi-document abstractive summarization. In *International Conference on Machine Learning*, pages 1223–1232.

Trevor Anthony Cohn and Mirella Lapata. 2009. Sentence compression as tree transduction. *Journal of Artificial Intelligence Research*, 34:637–674.

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, pages 4171–4186.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. In *Proceedings of the 33rd Conference on Neural Information Processing Systems (NeurIPS 2019)*, pages 13042–13054.

Sergey Edunov, Alexei Baevski, and Michael Auli. 2019. Pre-trained language model representations for language generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, volume 1, pages 4052–4059.

Günes Erkan and Dragomir R Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of artificial intelligence research*, 22:457–479.

Alexander R Fabbri, Irene Li, Tianwei She, Suyi Li, and Dragomir R Radev. 2019. Multi-news: a large-scale multi-document summarization dataset and abstractive hierarchical model. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1074–1084.

Angela Fan, Claire Gardent, Chloé Braud, and Antoine Bordes. 2019. Using local knowledge graph construction to scale seq2seq models to multi-document inputs. 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 4184–4194.

Patrick Fernandes, Miltiadis Allamanis, and Marc Brockschmidt. 2018. Structured neural summarization. In *Proceedings of the 7th International Conference on Learning Representations*.

Katja Filippova and Michael Strube. 2008. Sentence fusion via dependency graph compression. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 177–185. Association for Computational Linguistics.

Sebastian Gehrmann, Yuntian Deng, and Alexander M Rush. 2018. Bottom-up abstractive summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4098–4109.

Pierre-Etienne Genest and Guy Lapalme. 2011. Framework for abstractive summarization using text-to-text generation. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*, pages 64–73. Association for Computational Linguistics.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Logan Lebanoff, Kaiqiang Song, and Fei Liu. 2018. Adapting the neural encoder-decoder framework from single to multi-document summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4131–4141.

Wei Li. 2015. Abstractive multi-document summarization with semantic information extraction. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1908–1913.

Wei Li, Xinyan Xiao, Yajuan Lyu, and Yuanzhuo Wang. 2018a. Improving neural abstractive document summarization with explicit information selection modeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1787–1796.

Wei Li, Xinyan Xiao, Yajuan Lyu, and Yuanzhuo Wang. 2018b. Improving neural abstractive document summarization with structural regularization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4078–4087.

Wei Li and Hai Zhuge. 2019. Abstractive multi-document summarization based on semantic link network. *IEEE Transactions on Knowledge and Data Engineering*.

Kexin Liao, Logan Lebanoff, and Fei Liu. 2018. Abstract meaning representation for multi-document summarization. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1178–1190.

Chin-Yew Lin and Franz Josef Och. 2004. Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, pages 605–612. Association for Computational Linguistics.

Peter J Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. 2018. Generating wikipedia by summarizing long sequences. In *Proceedings of the 6th International Conference on Learning Representations*.

Yang Liu and Mirella Lapata. 2019a. Hierarchical transformers for multi-document summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5070–5081.

Yang Liu and Mirella Lapata. 2019b. Text summarization with pretrained encoders. 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 3728–3738.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Shulei Ma, Zhi-Hong Deng, and Yunlun Yang. 2016. An unsupervised multi-document summarization framework based on neural document model. In *Proceedings of the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1514–1523.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 404–411.

Shashi Narayan, Shay B Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807.

Vlad Niculae, André FT Martins, and Claire Cardie. 2018. Towards dynamic computation graphs via sparse latent structure. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 905–911.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *Proceedings of the 6th International Conference on Learning Representations*.

Laura Perez-Beltrachini, Yang Liu, and Mirella Lapata. 2019. Generating summaries with topic templates and structured convolutional decoders. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5107–5116.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, volume 1, pages 2227–2237.

Daniele Pighin, Marco Cornolti, Enrique Alfonseca, and Katja Filippova. 2014. Modelling events through memory-based, open-ie patterns for abstractive summarization. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 892–901.

Dragomir Radev. 2000. A common theory of information fusion from multiple text sources step one: cross-document structure. In *1st SIGdial workshop on Discourse and dialogue*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training.

Sascha Rothe, Shashi Narayan, and Aliaksei Severyn. 2019. Leveraging pre-trained checkpoints for sequence generation tasks. *arXiv preprint arXiv:1907.12461*.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 1073–1083.

Eva Sharma, Luyang Huang, Zhe Hu, and Lu Wang. 2019. An entity-driven framework for abstractive summarization. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3278–3289.

Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. 2019. Ernie 2.0: A continual pre-training framework for language understanding. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Xiaojun Wan. 2008. An exploration of document impact on graph-based multi-document summarization. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 755–762. Association for Computational Linguistics.

Xiaojun Wan and Jianguo Xiao. 2009. Graph-based multi-modality learning for topic-focused multi-document summarization. In *Proceedings of the*

21st International Joint Conference on Artificial Intelligence.

Lu Wang and Claire Cardie. 2013. Domain-independent abstract generation for focused meeting summarization. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1395–1405.

Wenmian Yang, Weijia Jia, Wenyuan Gao, Xiaojie Zhou, and Yutao Luo. 2019a. Interactive variance attention based online spoiler detection for time-sync comments. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 1241–1250. ACM.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019b. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems (NIPS 2019)*, pages 5754–5764.

Michihiro Yasunaga, Rui Zhang, Kshitijh Meelu, Ayush Pareek, Krishnan Srinivasan, and Dragomir Radev. 2017. Graph-based neural multi-document summarization. In *Proceedings of the 21st Conference on Computational Natural Language Learning*, pages 452–462.

Yongjing Yin, Linfeng Song, Jinsong Su, Jiali Zeng, Chulun Zhou, and Jiebo Luo. 2019. Graph-based neural sentence ordering. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 5387–5393.

Jianmin Zhang, Jiwei Tan, and Xiaojun Wan. 2018. Towards a neural network approach to abstractive multi-document summarization. *arXiv preprint arXiv:1804.09010.*

A Appendix

We report the sentence-level ROUGE-L evaluation results of our models on both the two datasets, so that future work can compare with them conveniently.

Model	R-1	R-2	R-L
RoBERTa+FT	42.05	27.00	40.05
GraphSum	42.63	27.70	40.13
GraphSum+RoBERTa	**42.99**	**27.83**	**40.97**

Table 6: Evaluation results on the WikiSum test set with **sentence-level** ROUGE-L value.

Model	R-1	R-2	R-L
RoBERTa+FT	44.26	16.22	40.64
GraphSum	45.02	16.69	41.11
G.S.(Similarity)+RoBERTa	45.93	17.33	42.02
G.S.(Topic)+RoBERTa	**46.07**	17.42	**42.22**
G.S.(Discourse)+RoBERTa	45.87	**17.56**	42.00

Table 7: Evaluation results on the MultiNews test set with **sentence-level** ROUGE-L value.