Hierarchical Heterogeneous Graph Attention Network for Syntax-Aware Summarization

Zixing Song, Irwin King

The Chinese University of Hong Kong
{zxsong, king}@cse.cuhk.edu.hk

Abstract

The task of summarization often requires a non-trivial understanding of the given text at the semantic level. In this work, we essentially incorporate the constituent structure into the single document summarization via the Graph Neural Networks to learn the semantic meaning of tokens. More specifically, we propose a novel hierarchical heterogeneous graph attention network over constituency-based parse trees for syntax-aware summarization. This approach reflects psychological findings that humans will pinpoint specific selection patterns to construct summaries hierarchically. Extensive experiments demonstrate that our model is effective for both the abstractive and extractive summarization tasks on six benchmark datasets from various domains. Moreover, further performance improvement can be obtained by virtue of state-of-the-art pre-trained models. Lastly, we test our model for the code summarization task for border impact.

Introduction

Text summarization has always been a fundamental task in natural language processing (NLP) to condense a complex input to a concise expression by retaining the core information at the same time. The relevant techniques can be categorized as either extractive ones, which only need to identify salient sentences from the original text, or abstractive ones, which may generate novel words and sentences.

Graph-based methods for summarization are becoming heated topics with the rise of Graph Neural Networks (GNN) thanks to its powerful capability to model the underlying useful relationships in the text graph. However, the output summary by existing graph-based methods tends to suffer from semantic deviation from the input text as the graphs constructed in these models are mostly at the statistical level, like word-sentence graph in HSG (Wang et al. 2020). In this case, the nodes tend to be text units and edges are connected by simple statistical scores like co-occurrence, pointwise mutual information (PMI), ignoring rich syntactic and semantic information for summarization. Other latest works (Jin, Wang, and Wan 2020; Wu et al. 2021) extend them with the aid of semantic graphs directly but suffer from relatively higher computational costs due to the complex graph construction step.

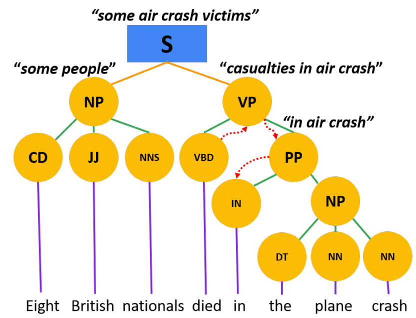

Figure 1: The constituency parsing tree for sentence *Eight British nationals died in the plane crash*. Red dotted line denotes the syntactic dependency path from *died* to *in* for the relationship *prep*. The path from *died* to *in* reflects the relation edge "died in" in the corresponding semantic graph. Each constituent can store the semantic meaning hierarchically for the final summary.

Model	Graph Level	GNN	Task
LexPageRank (2004a)	Statistical	w/o	Extractive
Graph-based Attention (2019)	Statistical	w/o	Abstractive
GatedGNN (2019)	Statistical	w/	Abstractive
SemSUM (2020)	Semantic	w/	Abstractive
DISCOBERT (2020)	Statistical	w/	Extractive
HSG (2020)	Statistical	w/	Extractive
HAHSum (2020)	Statistical	w/	Extractive
BASS (2021)	Semantic	w/o	Abstractive
Ours	**Syntactic**	**w/**	**Both**

Table 1: Comparison between our proposed model with other related graph-based summarization models.

Previous studies (Li et al. 2014; Xu and Durrett 2019) have shown that syntactic structure is beneficial for generating compressed yet informative summaries because its hierarchical structure facilitates the removal of insignificant parts and the attention for more salient ones (Figure 1). This also mimics the human way of generating summaries: fusing the semantic meaning by extracting the most significant information level by level, from words to phrases and finally to sentences. Therefore, it is natural to guide the neural summarization system with a tree-like text graph that embodies syntactic information so that it can identify summaryworthy content and compose summaries that preserve the vital meaning of the source texts. Besides, the syntactic graph is generally easier to obtain than the semantic graph and thus alleviates the computational issue in previous methods based on constructed complicated semantic graphs.

Furthermore, we choose the constituency parsing tree of the sentence as the text graph for the input of GNN in view of two primary reasons. First, the syntactic dependency relationship between tokens can be inexplicitly reflected via the path between them (as shown in Figure 1), so it has already encoded the information from the dependency-based parsing tree. Second, the constituency tree is a natural fit for extracting sub-phrases from the sentence, which is the exact case when identifying essential phrases to create summaries. Third, the extracted syntactic dependency can reflect the semantic relationship between tokens (Figure 1). That is why we favor the constituency-based tree.

Based on the aforementioned motivation, we will first utilize an off-the-shelf constituency parser to obtain the constituency tree for each sentence. Then, we propose a generic syntax-aware heterogeneous graph attention network to learn the representation for each type of node in this constructed tree-like graph. This proposed GNN model consists of two types of layers. One is the syntax-aware graph attention layer for detecting the syntactic dependency relationship between each constituent pair via meta-path, and the other is the hierarchical graph pooling layer for hierarchically gathering information from the tree.

The contributions of this work are summarized as follows.

- We propose a novel heterogeneous graph attention network for syntax-aware summarization based on the constituency tree. To the best of our knowledge, we are the first to incorporate constituency syntax for text summarization based on GNN. Moreover, our model is reasonably flexible and can be easily adapted into both abstractive and extractive tasks.
- We conduct extensive experiments on six datasets from various domains under abstractive and extractive settings to demonstrate its effectiveness. Furthermore, we investigate the potentially increased performance with the initiation of some SOTA pre-trained models.
- We apply the proposed techniques to the code summarization task by replacing the graph with the abstract syntax tree for border impact. Therefore, we further push the boundary for the graph-based models in NLP tasks.

Related Works

Neural Text Summarization

Recent years have witnessed great success of text summarization with the constant development of neural networks (Chen, Li, and King 2021; Li et al. 2019; Gao et al. 2021, 2020; Li et al. 2020), especially the recurrent neural networks (RNN) (See, Liu, and Manning 2017; Paulus, Xiong, and Socher 2018; Gehrmann, Deng, and Rush 2018) or Transformer (Liu and Lapata 2019; Zhang et al. 2020a; Bi et al. 2021). The most recent work focus on contextualized pre-trained language models (Zhong et al. 2020; Lewis et al. 2020; Liu, Dou, and Liu 2021) for further performance enhancement.

Graph Neural Networks

Graph Neural Networks (GNN) is a series of neural architectures for graph-structured data and has a wide range of applications (Song et al. 2021a,b; Yang et al. 2021). The GNN model gains tremendous popularity after the success of GCN (Kipf and Welling 2017) and GAT (Velickovic et al. 2018). GNNs use the graph topological structure along with the node and/or edge features to learn a representation vector for every node in the graph. More recently, there are a great number of NLP applications with the aid of GNN models for various tasks, like relation extraction (Zhu et al. 2019; Zhang et al. 2019), semantic role labeling (Christopoulou, Miwa, and Ananiadou 2019; Marcheggiani and Titov 2020) and text classification (Zhang and Zhang 2020; Ding et al. 2020; Xint et al. 2021).

Graph-based Text Summarization

Graph-based methods have been explored for text summarization since decades ago (Erkan and Radev 2004b,a). Later, with the help of GNN, a GCN-based model for multi-document summarization is purposed (Yasunaga et al. 2017) and Tan, Wan, and Xiao (2017) also design a graph-based attention mechanism for abstractive summarization. The latest works for abstractive summarization are SemSUM (Jin, Wang, and Wan 2020; Bi et al. 2021) via the semantic graphs. The other recent works (Jia et al. 2020; Wang et al. 2020) mainly focus on extractive summarization based on diverse message passing mechanisms in GNN. Details about GNN can be found in **Appendix A**.

Methodology

Our proposed model, SynapSum (as shorthand for **Syn**tax-aware Heterogeneous Gr**ap**h Attention Network for **Sum**marization), follows the dominant sequence-to-sequence framework (Sutskever, Vinyals, and Le 2014) for abstractive summarization (Figure 2) and the encoder part alone can be employed for extractive summarization (**Appendix B**). We incorporate the syntax information into the encoder to generate better representations for words, phrases, and sentences with different levels of granularity based on the constituency tree. Our model can easily be adapted for the extractive summarization task as it can learn a global representation for each sentence, and thus a graph-level classifier can be trained.

The input (document) tokens and the ground-truth output (summary) tokens are given as $x = \{x_1, x_2, \dots, x_{n_1}\}$ and $y = \{y_1, y_2, \dots, y_{n_2}\}$ respectively. n_1 and n_2 are the length of input and output tokens respectively. There are n sentences in total in the input tokens. A bidirectional LSTM encoder is first employed to get the embedding h_i^e for each input token x_i while a single LSTM decoder is then used to generate the embedding vector h_i^d for each output token y_i with the initialization hidden state for the decoder being set as $h_0^d = h_{n_1}^e$. The resulting embedding vectors will function as the initial representation for each token. The initialization can also be incorporated with some popular pre-trained models to gain potential performance increase.

Syntax-aware Heterogeneous Graph Attention Network

The encoder follows the framework of the sequence GNNs (Fernandes, Allamanis, and Brockschmidt 2019), and we incorporate the constituent structure into the GNN model so that syntactically-informed embeddings for both sentences and constituents are generated for more concise summarization. At each step t, a constituency-based parsing tree (Figure 3) is constructed based on the tth sentence in the input document. A constituency parsing tree consists of three types of nodes in our setting: the root node at the top level as the sentence node s, the leaf nodes as the word nodes w and the other non-terminal nodes at the intermediate levels as the constituent nodes c with various granularities.

Inspired by some representative models to tackle the heterogeneous graph embedding (Dong, Chawla, and Swami 2017; Fu et al. 2020), we propose a meta-path-based method to yield representations for constituency nodes by learning to detect and encode their syntactic dependency. Based on the learned representations, we design a graph pooling layer to encode the hierarchical information from bottom to top, producing the final embedding for the sentence node.

Syntax-aware graph attention layer By virtue of the heterogeneity of nodes, a node-type-specific projection matrix W_{π_i} is designed to transform nodes into a representation vector $h_{i,t} \in \mathbb{R}^d$ defined as,

$$h_{i,t} = W_{\pi_i} h_{i,t}^{\pi_i}, \tag{1}$$

where $h_{i,t}^{\pi_i}$ is the original embedding for node i at step t (tth sentence) and the node type π_i must satisfies $\pi_i \in \{s, c, w\}$ as the node can be **s**entence node, **c**onstituent node or **w**ord node. $h_{i,t}$ refers to the hidden state for node i at step t.

The goal of this layer is to detect the syntactic relationship among them and encode the syntax information for every constituent. More specifically, for each node i in the constituency tree of tth sentence, we learn its syntactic dependency with other nodes in its neighborhood $\mathcal{N}_{i,t}^{\phi}$ via the meta-path-based graph attention mechanism as Eq. (2),

$$\begin{aligned} e_{ij,t}^{\phi} &= \text{LeakyReLU}(a_{\phi,t}^T [h_{i,t} \| h_{j,t}]), \\ \alpha_{ij,t}^{\phi} &= \frac{\exp e_{ij,t}^{\phi}}{\sum_{k \in \mathcal{N}_{i,t}^{\phi}} \exp e_{ik,t}^{\phi}}. \end{aligned} \tag{2}$$

Here, $\alpha_{ij,t}^{\phi}$ is the attention score. $a_{\phi,t} \in \mathbb{R}^{2d}$ is a learnable weight vector. $\|$ is the concatenation operation. The meta-path ϕ of length l is defined as a path in the form of $\pi_1 \pi_2 \ldots \pi_l$. All nodes appearing on the meta-path (instances) ϕ of node i belong to its neighborhood $\mathcal{N}_{i,t}^{\phi}$.

Afterward, the message is generated by a weighted sum over all nodes along each meta-path in the neighborhood and then aggregated together by a mean pooling operation to update the embedding for the target node i as Eq. (3).

$$h_{i,t} = \frac{1}{|\Phi_i|} \sum_{\phi \in \Phi_i} \sigma\Big(\sum_{j \in \mathcal{N}_{i,t}^{\phi}} \alpha_{ij,t}^{\phi} h_{j,t}\Big), \tag{3}$$

where Φ_i represents the set of all the considered meta-paths and it depends on the node type π_i. σ is the activation function. A toy example is illustrated in Figure 4. This syntax-aware graph attention layer can be easily stacked by iterating Eq. (2) to Eq. (3) multiple times.

Hierarchical graph pooling layer To ensure that the sentence node's hidden state vector embodies rich and accurate semantic information and properly attends all the child constituency nodes in a hierarchical manner, another graph pooling layer is attached on top of the stacked syntax-aware graph attention layers. Note that the hidden state vector for the root sentence node is fixed in the aforementioned layers. Therefore, this graph pooling layer is designed to generate a final representation vector for the whole constituency tree by taking its hierarchical structure information into account, and it will then be assigned to the sentence node.

The graph pooling operation is conducted from bottom to top along edges in the constituency tree via another attention mechanism. As not all child nodes contribute equally to the representation of the parent nodes, the proposed attention mechanism learns the importance of each child node to its parent node in a hierarchical order. The attention score between node i and its child node j is given as Eq. (4).

$$\begin{aligned} u_{j,t} &= \tanh(W_{p,t}^{\pi_j \pi_i} h_{j,t}^{\pi_j} + b_{p,t}^{\pi_j \pi_i}), \\ \alpha_{ij,t} &= \frac{\exp(u_{j,t}^T h_{i,t}^{\pi_i})}{\sum_{k \in \text{Child}(i)} \exp(u_{k,t}^T h_{i,t}^{\pi_i})}. \end{aligned} \tag{4}$$

As the hidden state $h_{i,t}^{\pi_i}$ for each node after the aforementioned layers has encoded the neighborhood information, it can act as a context vector in Eq. (4). The importance of each child node j is measured as the similarity of $u_{j,t}$ with the parent node's context vector $h_{i,t}^{\pi_i}$, followed by a normalization. The parameters $W_{p,t}^{\pi_j \pi_i}$ and $b_{p,t}^{\pi_j \pi_i}$ in the Eq. (4) are shared across the same type of edges to align the dimension difference between the parent and child nodes.

We associate each non-terminal node with an accumulation vector $h'_{i,t}$, whose dimension is the same as $h_{i,t}^{\pi_i}$, to record the information it receives from all of its child nodes. The accumulation vector for the word node is directly assigned as its corresponding embedding $h_{i,t}^{w}$. We also apply the gating mechanism to decide how much information is allowed to transfer from lower-level node j to the higher-level node i as Eq. (5) and Eq. (6).

$$z_i = \sigma(W_{g,t} \sum_{j \in \text{Child}(i)} h'_{j,t} + b_{g,t}) \tag{5}$$

$$h'_{i,t} = (1 - z_i) \odot h_{i,t}^{\pi_i} + z_i \odot (W_{\pi_i \pi_j} \sum_{j \in \text{Child}(i)} \alpha_{ij,t} h_{j,t}^{\pi_j}) \tag{6}$$

The σ in Eq. (5) is the sigmoid function, and the $\odot$ operator indicates the Hadamard product. The trainable weight $W_{\pi_i \pi_j}$ serves as a projection matrix when the node type of the parent node π_i and the child node π_j are disparate.

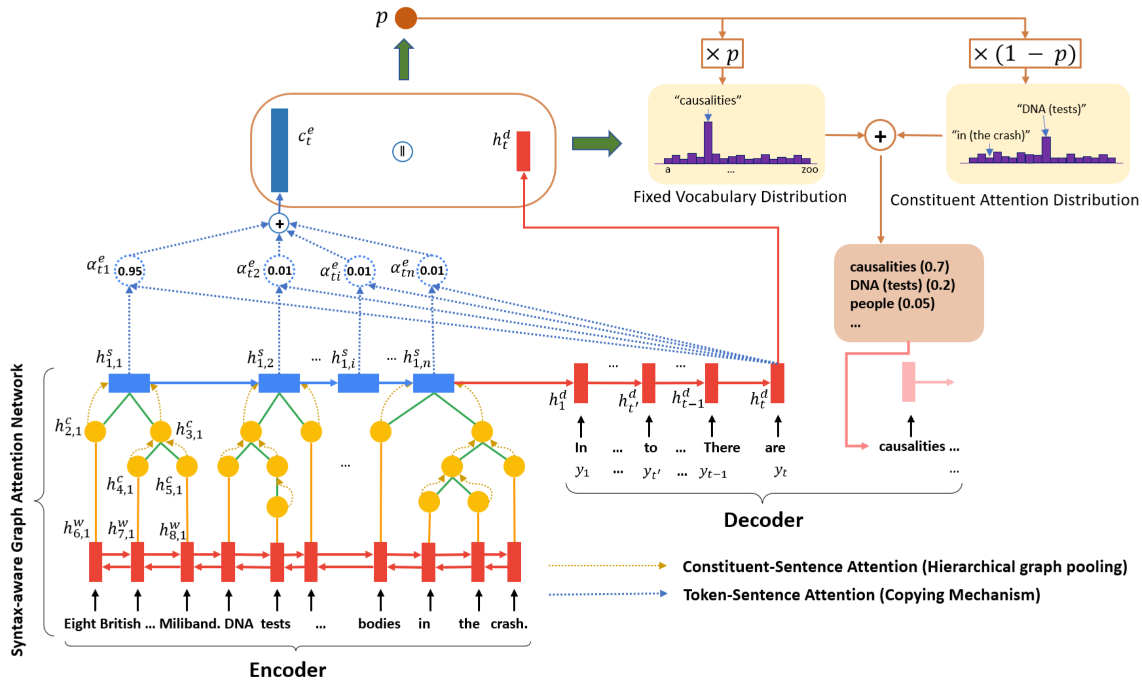

Figure 2: Overview of SynapSum model for abstractive summarization. The encoder follows Sequence GNN framework in which we propose a novel syntax-aware graph attention network, consisting of the stacked syntax-aware graph attention layer and a top hierarchical graph pooling layer. SynapSum model for extractive summarization can be found in **Appendix B**.

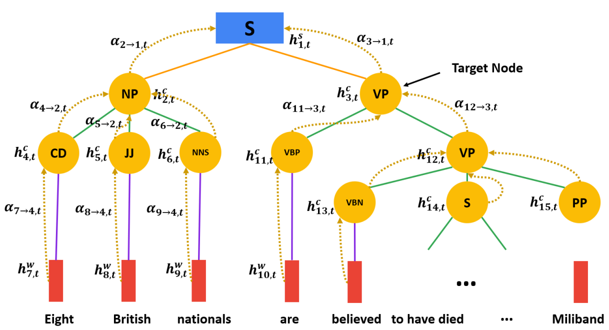

Figure 3: An example constituency tree for tth sentence. Different colors denote different types of nodes and edges. Yellow dotted directed lines show the attention mechanism in the hierarchical graph pooling layer. (Also refer to Figure 4.)

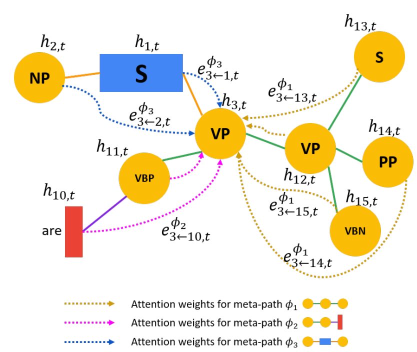

Figure 4: A toy example for updating $h_{3,t}$ (the target node in Figure 3) in syntax-aware graph attention layer via meta-path-based attention mechanism. The meta-path of length 2 is considered for constituency nodes. There are 3 types of meta-paths in total $\Phi_3 = \{\phi_1, \phi_2, \phi_3\}$ and ϕ_1 has 3 meta-path instances 3-12-13, 3-12-14 and 3-12-15.

Finally, the accumulation vector for the root sentence node $h'_{1,t}$ is obtained by iterating between Eq. (5) and Eq. (6) to generate $h'_{i,t}$ along the edges from bottom to top. Note that the index for the sentence node is always set to 1 for convenience. This hierarchical graph pooling operation is followed by another one-layer MLP to produce the initial embedding for the sentence node at the next step as $h^s_{1,t+1} = \tanh(W_{hh}h'_{1,t} + b_h)$.

It is worth noting that the root sentence node attends over all the constituency nodes with the help of a hierarchical graph pooling operation. The attention score between the sentence node (index fixed as 1) with any constituency node j at step t is given as Eq. (7)

$$\alpha_{1j,t} = \prod_{i=1}^{j-1} \alpha_{v_i v_{i+1},t}, \tag{7}$$

where a unique path $v_1, v_2, \ldots, vj$ exists between the root sentence node i and the constituency node j with $v_1 = 1$ and $v_j = j$. It is easy to see that node v_{i+1} must be a child node of node v_i and each term $\alpha_{v_i v_{i+1},t}$ is given by Eq. (4).

Token Generator and Constituency Pointer

With the purpose of avoiding the out-of-vocabulary (OOV) issue (See, Liu, and Manning 2017), we adopt the copying mechanism (Gu et al. 2016). To reduce the number of repetitions during decoding steps (Sankaran et al. 2016), we first calculate the encoder context vector c_t^e as Eq. (8). h_t^d is the hidden state at decoding time step t.

$$\alpha_{ti}^e = \frac{\exp((W_w h_t^d)^T W_e (W_s h_{1,i}^s))}{\sum_{j=1}^{n} \exp((W_w h_t^d)^T W_e (W_s h_{1,j}^s))}$$
$$c_t^e = \sum_{i=1}^{n} \alpha_{ti}^e h_{1,i}^s. \tag{8}$$

Then we define a binary value u_t as a switch at each decoding step t to decide whether to generate a token from the dictionary or copy a certain constituent with distribution $p(u_t = 1) = \sigma(W_u[h_t^d \| c_t^e] + b_u)$. On the one hand, the token generation follows the probability distribution as $p(y_t|u_t = 0) = \text{softmax}(W_{gen}[h_t^d \| c_t^e] + b_{gen})$. On the other, the constituent pointer mechanism utilizes the attention score appearing in the hierarchical graph pooling layer as the probability $p(y_t = c_{j,i}[0]|u_t = 1) = \alpha_{ti}^e \alpha_{1j,i}$ to copy the first token in the specific constituency. Here, α_{ti}^e and $\alpha_{1j,i}$ is given by Eq. (8) and Eq. (7) respectively. $c_{j,i}[0]$ represents the first token in the constituency node j in sentence i. The final distribution for the output token $p(y_t)$ is given as $p(u_t = 1)p(y_t|u_t = 1) + (1 - p(u_t = 1))p(y_t|u_t = 0)$.

Loss Function

For the abstractive summarization task, the final loss function is based on the maximum-likelihood training objective with the ground-truth summary sequence $y = \{y_1, y_2, \ldots, y_{n_2}\}$ and the input token sequence x as $\mathcal{L} = -\sum_{t=1}^{n_2} \log p(y_t|y_1, \ldots, y_{t-1}, x)$. For the extractive summarization task, we convert the problem into a graph-level classification problem and directly train a binary classifier on top of the graph pooling layer (**Appendix B**).

Experiment

Datasets

We choose six datasets to evaluate our model. The data split is described in Table 2. **CNN/DM** (Hermann et al. 2015; See, Liu, and Manning 2017) is the most widely used dataset that contains news articles and associated highlights as reference summaries. **New York Times (NYT)** (Sandhaus 2008) is another dataset that is composed of news articles and associated gold summaries. We follow the split in (Kedzie, McKeown, and III 2018). **XSum** (Narayan, Cohen, and Lapata 2018) is a larger dataset that consists of one-sentence summaries of articles from BBC. **Reddit** (Kim, Kim, and Kim 2019) is a social media dataset collecting posts from Reddit. TIFU-long version is used. **WikiHow** (Koupaee and Wang 2018) is another large-scale dataset extracted from a knowledge base website. **PubMed** dataset comes from an academic paper database. All datasets are used for the **single-document summarization** task.

Dataset	Domain	Split			Avg. Len		#Ext
		Train	Valid	Test	Doc.	Sum.	
CNN/DM	News Posts	287K	13K	11K	766.1	58.2	3
NYT	News Posts	44K	5K	6K	1183.2	110.8	4
XSum	News Posts	203K	11K	11K	430.2	23.3	2
Reddit	Social Media	42K	0.6K	0.6K	482,2	28.0	2
WikiHow	Knowledge Base	168K	6K	6K	580.8	62.6	4
PubMed	Academic Paper	83K	5K	5K	44.0	209.5	6

Table 2: Datasets statistics. **#Ext** is the number of the extracted sentences for the extractive summarization task.

Baseline Models

Abstractive models **Pointer-generator** (See, Liu, and Manning 2017) is the first model that copies words from the source text via pointing and coverage mechanism. **Intra-attention** (Paulus, Xiong, and Socher 2018) is a deep reinforced model with intra-attention. **Graph-based Attention** (Tan, Wan, and Xiao 2017) is the early work to use the graph for summarization. **Bottom-up** (Gehrmann, Deng, and Rush 2018) model uses a bottom-up attention step. **Gated GNN** (Fernandes, Allamanis, and Brockschmidt 2019) model extends the sequence model with a GNN to reason about long-distance relations. **SemSUM** (Jin, Wang, and Wan 2020) and **BASS** (Wu et al. 2021) are the state-of-the-art GNN models for the abstractive summarization task.

Extractive models **NeuSUM** (Zhou et al. 2018) model jointly learns to score and select sentences. **BanditSum** (Dong et al. 2018) model extracts summarization as a contextual bandit. **JECS** (Xu and Durrett 2019) is another extractive model via syntactic compression. **HSG** (Wang et al. 2020) is the state-of-the-art GNN model for extractive summarization based on the statistical text graph only. **DISCOBERT** (Xu et al. 2020) and **HANSum** (Jia et al. 2020) are also two latest GNN-based models, pre-trained with BERT (Devlin et al. 2019) model.

Implementation Details

Graph Construction We train the state-of-the-art constituency parser[1] on the English Penn Treebank (PTB) dataset[2] via self-attention mechanism. After that, we can use the trained parser to get the constituency parsing tree of the sentences in our tested datasets. We use the same set of parameters suggested in the original paper (Mrini et al. 2020).

Parameters Setting We set our model parameters based on some preliminary experiments on the development set. There are three types of nodes in the constituency parsing tree: sentence node, constituency node, and word node. The dimension of the embedding for them is 4096, 512, 128, respectively by default. In the syntax-aware graph attention layer, the node embedding for the root sentence node and the leaf word nodes are fixed without updating. The meta-paths for each constituency node are all length-two paths starting from itself. The number of stacked syntax-aware graph attention layers ranges from 2 to 5. We only stack one layer of hierarchical graph pooling to avoid the issue

[1]https://github.com/KhalilMrini/LAL-Parser

[2]https://catalog.ldc.upenn.edu/LDC2015T13

of over-smoothing. We choose the Adam optimizer with an initial learning rate 0.0001, momentum values $\beta_1 = 0.9$, $\beta_2 = 0.999$ and weight decay $\epsilon = 10^{-5}$. We feed the graph into our model in a mini-batch fashion with a size of 256. In addition, during the decoding step, a beam search strategy is utilized with the beam size of 3. More details can be found in **Appendix C**.

Automatic Evaluation

We evaluate the quality of the summarization based on ROUGE (Lin 2004). We report unigram and bigram overlap (ROUGE-1 and ROUGE-2) between generated or extracted summaries and gold summaries to assess informativeness. The longest common subsequence (ROUGE-L) on sentence-level is reported for evaluating fluency. We additionally report BERTScore F1[3] that correlates with human evaluation better than ROUGE (Zhang et al. 2020b). Examples of system output can be referred to **Appendix D** for case study.

Evaluation w/o pre-training In our original model, no pre-trained models are involved so we remove any pre-trained models in the baseline models for fair comparison in this section. We show the results of our proposed model against recently released summarization models on three news datasets in Table 3. We classify all the baselines into two groups: non-graph-based models and graph-based models. From the results, we can see that graph-based methods generally show comparable performance with the non-graph-based ones when GNN models are incorporated, as the case for Gated GNN and SemSUM. Furthermore, SynapSum outperforms the existing popular graph-based models and the listed non-graph-based models without pre-training in abstractive and extractive settings. SynapSum achieves the improvement by converting each sentence into a hierarchical graph with finer granularity and generating representations for words, constituents, and sentences simultaneously with rich syntactic and semantic meanings, as the BERTScore increase shows. However, the graphs used in other models tend only to capture the relationship between words and sentences, which leads to the loss of semantic meanings of the phrases as a whole. Therefore, our model can better mimic the human way of conducting summarization hierarchically, from words to phrases to sentences and documents, finally. We also demonstrate its efficacy on summarization datasets from other domains in Table 4.

Evaluation w/ pre-training The most recent summarization models are typically instantiated with some contextualized pre-trained models like BERT (Devlin et al. 2019). Consequently, it is essential to demonstrate the potential performance improvement of our proposed model with the help of pre-trained models. We use two popular recent pre-trained models: BART (Lewis et al. 2020) and MatchSum (Zhong et al. 2020) for abstractive and extensive summarization tasks respectively. More specifically, we firstly pre-train on XSum dataset as an extra corpus dataset, following the setting used in PEGASUS (Zhang et al. 2020a). Then we feed

Model	CNN/DM				NYT				XSum			
	R-1	R-2	R-L	BS	R-1	R-2	R-L	BS	R-1	R-2	R-L	BS
Oracle	55.76	33.22	51.83	-	55.84	38.39	50.00	-	-	-	-	-
Non-graph-based Methods (Abstractive)												
Pointer-generator	36.44	15.66	33.42	37.63	43.15	26.98	40.14	47.48	29.70	9.21	23.24	29.13
Intra-attention	38.30	14.81	35.49	38.17	43.86	27.10	40.11	47.83	29.46	9.73	24.37	29.66
Bottom-up	41.22	18.68	**38.34**	39.64	47.38	30.37	41.81	48.75	29.83	9.95	24.92	29.52
Graph-based Methods (Abstractive)												
Graph-based Attention	30.35	9.80	20.07	35.44	42.93	26.75	39.22	47.29	25.41	6.27	18.83	27.58
Gated GNN	38.10	16.11	33.23	37.69	44.37	27.70	40.65	48.12	27.54	8.93	22.18	28.25
SemSUM	40.03	18.56	37.58	39.45	**47.86**	28.83	40.60	48.61	29.73	9.34	24.86	29.74
BASS	39.76	17.96	36.33	38.58	45.43	29.46	41.07	48.47	29.11	9.39	24.72	29.57
SynapSum-Abs	**41.52**	**18.82**	38.29	**39.72**	47.21	**30.93**	**41.84**	**48.93**	**30.18**	**11.43**	**25.97**	**30.19**
Non-graph-based Methods (Extractive)												
BanditSum	41.50	18.70	37.60	39.38	46.94	25.71	41.43	48.55	-	-	-	-
NeuSUM	41.59	19.01	37.98	39.46	47.35	27.03	42.05	49.34	-	-	-	-
JECS	41.77	18.53	37.92	39.25	47.70	28.57	41.99	49.26	-	-	-	-
Graph-based Methods (Extractive)												
DISCOBERT w/o BERT	43.26	20.11	39.45	40.12	46.94	27.64	41.83	49.79	-	-	-	-
HSG	42.31	19.51	38.74	39.87	46.89	27.26	42.58	49.40	-	-	-	-
HAHSum w/o ALBERT	43.74	20.84	39.93	40.31	48.32	30.61	**43.37**	50.86	-	-	-	-
SynapSum-Ext	**44.32**	**21.07**	**40.36**	**41.04**	**48.93**	**31.98**	43.34	**51.30**	-	-	-	-

Table 3: Automatic evaluation results of our proposed model against recently released summarization models on three news datasets. XSum dataset is used for abstractive summarization only, as suggested by the authors of the dataset. All the models are not involved with pre-trained models. For a fair comparison, we substitute the corresponding pre-trained component in DISCOBERTand HAHSum with the same bidirectional LSTM used in our model. All scores have a 95% confidence interval of at most ± 0.25 as reported by the official scripts.

Model	Reddit (Social Media)				WikiHow (Knowledge Base)				PubMed (Academic Papers)			
	R-1	R-2	R-L	BS	R-1	R-2	R-L	BS	R-1	R-2	R-L	BS
Oracle	36.21	13.74	28.93	-	35.59	12.98	32.68	-	45.12	20.33	40.19	-
Abstractive												
Graph-based Attention	15.67	3.56	13.28	73.53	22.86	3.90	22.13	68.52	32.74	6.09	26.35	81.43
Gated GNN	21.46	3.41	16.91	75.86	26.02	5.15	24.68	70.74	34.70	8.35	30.20	83.69
SemSUM	**23.28**	**4.08**	**18.67**	**77.37**	27.21	5.95	26.84	71.48	36.32	10.78	31.60	85.02
BASS	22.06	3.85	17.26	76.48	26.84	6.07	25.92	71.06	35.88	9.70	31.42	84.67
SynapSum-Abs	22.13	3.76	17.64	76.69	**27.73**	**6.21**	**27.88**	**71.37**	**37.65**	**11.30**	**32.98**	**85.25**
Extractive												
DISCOBERT w/o BERT	20.94	4.12	15.18	74.22	27.45	5.90	24.59	71.15	36.33	9.26	30.88	83.96
HSG	24.96	**6.29**	**20.21**	78.64	30.27	8.46	28.95	72.08	40.37	13.90	36.03	86.14
HAHSum w/o ALBERT	23.91	5.30	19.14	77.90	28.36	7.93	28.01	71.89	39.95	12.50	34.32	85.88
SynapSum-Ext	**25.02**	6.17	20.02	**78.71**	**31.79**	**8.94**	**29.56**	**73.46**	**41.20**	**14.88**	**36.79**	**87.33**

Table 4: Automatic evaluation results of our proposed model against recently released graph-based summarization models on three datasets from different domains.

the word embeddings as the initial hidden states for word nodes in SynapSum to substitute for the original bidirectional LSTM and further fine-tune the model. For fairness, we also replace the corresponding component for word embedding in DISCOBERT and HAHSum with the same pre-trained model. Besides, we add the same pre-training step for other models due to the lack of pre-training in their original models. We investigate the potential effects of pre-trained on graph-based methods and compare the results with the SOTA pre-training methods for text summarization. We summarize the results in Table 5, and it shows that SynapSum performs better than all the other graph-based models. It is obvious that pre-training can further enhance the performance of SynapSum, and it even shows comparable results with the current SOTA pure pre-trained model RefSum (Liu, Dou, and Liu 2021).

Human Evaluation

We further access the proposed model by eliciting human judgment (Fabbri et al. 2021). For CNN/DM dataset, we randomly choose 100 samples from the testing set, along with a list of generated summaries by different models, and then present them to several qualified volunteers. These annotators are requested to rank these summaries by taking the fol-

[3]The robertalarge_L17_noidf_version is used for the BERTScore model.

Model	CNN/DM				NYT			
	R-1	R-2	R-L	BS	R-1	R-2	R-L	BS
Oracle	55.76	33.22	51.83	-	55.84	38.39	50.00	-
Abstractive								
Graph-based Attention	30.35	9.80	20.07	35.44	42.93	26.75	39.22	47.29
Graph-based Attention w/ BART	32.05	10.24	23.70	39.10	43.27	28.22	41.23	49.30
Gated GNN	38.10	16.11	33.23	37.69	44.37	27.70	40.65	48.12
Gated GNN w/ BART	41.38	17.77	34.02	39.91	48.08	30.41	41.98	49.75
SemSUM	40.03	18.56	37.58	39.45	47.86	28.83	40.60	48.61
SemSUM w/ BART	42.33	21.03	38.35	41.63	48.42	30.59	42.31	50.04
RefSum-Abs (SOTA PTM)	**44.96**	**21.50**	**41.43**	42.26	48.30	30.74	42.85	**50.30**
SynapSum-Abs	41.52	18.82	38.29	39.72	47.21	30.93	41.84	48.93
SynapSum-Abs w/ BART	44.32	21.93	41.10	**42.34**	**48.45**	**31.48**	**43.27**	50.11
Extractive								
DISCOBERT	43.26	20.11	39.45	40.12	46.94	27.64	41.83	49.79
DISCOBERT w/ MatchSum	43.37	20.42	40.23	40.78	49.79	30.32	42.95	50.86
HSG	42.31	19.51	38.74	39.87	46.89	26.26	42.58	49.40
HSG w/ MatchSum	43.24	20.33	39.60	40.36	48.45	29.02	43.14	50.78
HAHSum	43.74	20.84	39.93	40.31	48.32	30.61	43.37	50.86
HAHSum w/ MatchSum	44.57	21.14	40.53	41.44	49.36	31.41	44.97	51.44
RefSum-Ext (SOTA PTM)	**46.12**	22.46	**42.92**	43.03	50.27	32.96	**46.50**	53.25
SynapSum-Ext	44.32	21.07	40.36	41.07	48.93	31.98	43.34	51.30
SynapSum-Ext w/ MatchSum	46.08	**22.48**	42.71	**43.10**	**50.30**	**33.02**	45.33	**53.53**

Table 5: Automatic evaluation results of our proposed model against other graph-based summarization models w/ & w/o pre-training models.

Model	1st	2nd	3rd	4th	5th	Mean Rating
Graph-based Attention (2017)	0.14	0.24	0.28	0.12	0.22	-0.04*
Gated GNN (2019)	0.24	0.18	0.11	0.29	0.18	0.01*
SemSUM (2020)	0.46	0.26	0.20	0.03	0.05	1.05
BASS (2021)	0.31	0.17	0.27	0.19	0.06	0.48*
SynapSum-Abs	0.48	0.32	0.13	0.04	0.03	**1.18**

Table 6: Overall ranking results of summaries by human evaluation on CNN/DM dataset. The larger mean rating indicates better quality. * denotes the overall mean rating of the corresponding model is significantly outperformed by SynapSum via Welch's t-test ($p < 0.01$).

lowing criteria into account: fluency (is the summary grammatically correct?), informativeness (does the summary capture important facts?), and succinctness (is there any repetition in the summary?). The ranking value is 1 (best) to 5 (worst), and ties are allowed. We also normalize the ranking value by converting 1, 2, 3, 4, 5 to 2, 1, 0, -1, -2 respectively. The final rating value is obtained by averaging the scores for all the test samples. Table 6 summarizes the results of human evaluation on four baseline models and the proposed model for the abstractive summarization task. Based on these results, our model performs better than others. Detailed settings and results can be found in **Appendix E**.

Model Analysis

Syntactic Efficacy Analysis To further demonstrate the efficacy of introducing syntactic information into summarization, we use various constituency parsers with different levels of quality. If better metrics of the generated summaries can be obtained when a higher-quality parser is utilized during the graph construction step, the efficacy and the benefits of the syntax structure can be empirically proven because the performance would not even vary with respect to the quality of the constituency parser if syntactic information were not helpful. The results and details are shown in **Appendix F**. It clearly shows that BERTScore increases as the quality of the parser rises.

Ablation Study We perform the ablation study to investigate the potential influence of different components. We design five settings: (1) we delete the bidirectional LSTM layer for word node embedding and directly use random initialization; (2) we remove the syntax-aware graph attention layer and use the accumulation vector in hierarchical graph pooling operation; (3) we take off the hierarchical graph pooling layer and update the sentence node's hidden state in syntax-aware graph attention layer. (4) For the abstractive setting, we may also eliminate the copying mechanism. We conduct the experiments and list all the results in Table 7 and it shows that the graph attention layer is the most influential module.

Models	R-1	R-2	R-L	BS
SynapSum-Abs	**41.52**	**18.82**	**38.29**	**39.72**
w/o LSTM	41.40	18.47	38.04	39.60
w/o graph attention layer	41.04	17.96	37.78	36.89
w/o graph pooling layer	41.12	18.04	37.75	37.03
w/o copying mechanism	41.26	18.50	37.81	38.27

Table 7: Ablation study on CNN/DM dataset for the abstractive summarization task based on automatic evaluation scores.

Model	Python			Javascript		
	Prec.	Rec.	F1	Prec.	Rec.	F1
code2seq (2019)	35.79	24.85	29.34	30.18	19.88	23.97
GREAT (2020)	35.07	31.59	33.24	31.20	26.84	28.86
S&C (2021)	**36.40**	33.66	34.97	**35.06**	**29.61**	**32.11**
SynapSum-Code	35.93	**34.21**	**35.04**	32.25	25.76	28.64

Table 8: Code summarization results (partial) on the CSN dataset. (F1 denotes micro F1.)

Time Complexity Analysis As our improvement lies in the encoder, the main overhead is the syntax-aware graph attention layer with the time complexity with $O(|V|F_1F_2 + |E|F_2)$. Here, F_1 and F_2 are embedding sizes for two consecutive layers, which are fixed in practice. Therefore, our method is linear to the number of nodes $|V|$ and meta-path based node pairs $|E|$, which is scalable and on par with baseline graph-based models. The actual running time results are presented with details in **Appendix G**.

Code Summarization Extension For broader impact, we show the experimental results on the code summarization. The goal is to generate the function's name given the body of the function, similar to the abstractive text summarization. Our model is a natural fit as we can replace the constituency tree with the abstract syntax tree (AST) for the code summarization task. We follow the experimental setting described in the work (Zügner et al. 2021) on CSN dataset (Husain et al. 2019). The partial results are shown in Table 8 (Details can be found in **Appendix H**). It illustrates that our model is flexible and can even achieve comparable results with latest models exclusively designed for code summarization.

Conclusion

This paper introduces another graph-based method for text summarization, which utilizes syntactic information as guidance based on the constituency tree. We leave how to incorporate graph information in the decoder as future work.

Acknowledgments

We would like to thank Dr. Wang Chen for his detailed suggestions on the manuscript. The work described in this paper was partially supported by CUHK 2410021, Research Impact Fund (RIF), R5034-18 and the Amazon AWS Machine Learning Research Award.

References

Alon, U.; Brody, S.; Levy, O.; and Yahav, E. 2019. code2seq: Generating Sequences from Structured Representations of Code. In *ICLR (Poster)*. OpenReview.net.

Bi, K.; Jha, R.; Croft, W. B.; and Celikyilmaz, A. 2021. AREDSUM: Adaptive Redundancy-Aware Iterative Sentence Ranking for Extractive Document Summarization. In *EACL*, 281–291. Association for Computational Linguistics.

Chen, W.; Li, P.; and King, I. 2021. A Training-free and Reference-free Summarization Evaluation Metric via Centrality-weighted Relevance and Self-referenced Redundancy. In *ACL/IJCNLP (1)*, 404–414. Association for Computational Linguistics.

Christopoulou, F.; Miwa, M.; and Ananiadou, S. 2019. Connecting the Dots: Document-level Neural Relation Extraction with Edge-oriented Graphs. In *EMNLP/IJCNLP (1)*, 4924–4935. Association for Computational Linguistics.

Devlin, J.; Chang, M.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT (1)*, 4171–4186. Association for Computational Linguistics.

Ding, K.; Wang, J.; Li, J.; Li, D.; and Liu, H. 2020. Be More with Less: Hypergraph Attention Networks for Inductive Text Classification. In *EMNLP (1)*, 4927–4936. Association for Computational Linguistics.

Dong, Y.; Chawla, N. V.; and Swami, A. 2017. metapath2vec: Scalable Representation Learning for Heterogeneous Networks. In *KDD*, 135–144. ACM.

Dong, Y.; Shen, Y.; Crawford, E.; van Hoof, H.; and Cheung, J. C. K. 2018. BanditSum: Extractive Summarization as a Contextual Bandit. In *EMNLP*, 3739–3748. Association for Computational Linguistics.

Erkan, G.; and Radev, D. R. 2004a. LexPageRank: Prestige in Multi-Document Text Summarization. In *EMNLP*, 365–371. ACL.

Erkan, G.; and Radev, D. R. 2004b. LexRank: Graph-based Lexical Centrality as Salience in Text Summarization. *J. Artif. Intell. Res.*, 22: 457–479.

Fabbri, A. R.; Kryscinski, W.; McCann, B.; Xiong, C.; Socher, R.; and Radev, D. R. 2021. SummEval: Re-evaluating Summarization Evaluation. *Trans. Assoc. Comput. Linguistics*, 9: 391–409.

Fernandes, P.; Allamanis, M.; and Brockschmidt, M. 2019. Structured Neural Summarization. In *ICLR (Poster)*. OpenReview.net.

Fu, X.; Zhang, J.; Meng, Z.; and King, I. 2020. MAGNN: Metapath Aggregated Graph Neural Network for Heterogeneous Graph Embedding. In *WWW*, 2331–2341. ACM / IW3C2.

Gao, Y.; Li, J.; Lyu, M. R.; and King, I. 2021. Open-Retrieval Conversational Machine Reading. *CoRR*, abs/2102.08633.

Gao, Y.; Wu, C.; Li, J.; Joty, S. R.; Hoi, S. C. H.; Xiong, C.; King, I.; and Lyu, M. R. 2020. Discern: Discourse-Aware Entailment Reasoning Network for Conversational Machine Reading. In *EMNLP (1)*, 2439–2449. Association for Computational Linguistics.

Gehrmann, S.; Deng, Y.; and Rush, A. M. 2018. Bottom-Up Abstractive Summarization. In *EMNLP*, 4098–4109. Association for Computational Linguistics.

Gu, J.; Lu, Z.; Li, H.; and Li, V. O. K. 2016. Incorporating Copying Mechanism in Sequence-to-Sequence Learning. In *ACL (1)*. The Association for Computer Linguistics.

Hellendoorn, V. J.; Sutton, C.; Singh, R.; Maniatis, P.; and Bieber, D. 2020. Global Relational Models of Source Code. In *ICLR*. OpenReview.net.

Hermann, K. M.; Kocisky, T.; Grefenstette, E.; Espeholt, L.; Kay, W.; Suleyman, M.; and Blunsom, P. 2015. Teaching machines to read and comprehend. In *Advances in neural information processing systems*, 1693–1701.

Husain, H.; Wu, H.; Gazit, T.; Allamanis, M.; and Brockschmidt, M. 2019. CodeSearchNet Challenge: Evaluating the State of Semantic Code Search. *CoRR*, abs/1909.09436.

Jia, R.; Cao, Y.; Tang, H.; Fang, F.; Cao, C.; and Wang, S. 2020. Neural Extractive Summarization with Hierarchical Attentive Heterogeneous Graph Network. In *EMNLP (1)*, 3622–3631. Association for Computational Linguistics.

Jin, H.; Wang, T.; and Wan, X. 2020. SemSUM: Semantic Dependency Guided Neural Abstractive Summarization. In *AAAI*, 8026–8033. AAAI Press.

Kedzie, C.; McKeown, K. R.; and III, H. D. 2018. Content Selection in Deep Learning Models of Summarization. In *EMNLP*, 1818–1828. Association for Computational Linguistics.

Kim, B.; Kim, H.; and Kim, G. 2019. Abstractive Summarization of Reddit Posts with Multi-level Memory Networks. In *NAACL-HLT (1)*, 2519–2531. Association for Computational Linguistics.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR (Poster)*. OpenReview.net.

Koupaee, M.; and Wang, W. Y. 2018. WikiHow: A Large Scale Text Summarization Dataset. *CoRR*, abs/1810.09305.

Lewis, M.; Liu, Y.; Goyal, N.; Ghazvininejad, M.; Mohamed, A.; Levy, O.; Stoyanov, V.; and Zettlemoyer, L. 2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *ACL*, 7871–7880. Association for Computational Linguistics.

Li, C.; Liu, Y.; Liu, F.; Zhao, L.; and Weng, F. 2014. Improving Multi-documents Summarization by Sentence Compression based on Expanded Constituent Parse Trees. In *EMNLP*, 691–701. ACL.

Li, J.; Gao, Y.; Bing, L.; King, I.; and Lyu, M. R. 2019. Improving Question Generation With to the Point Context. In *EMNLP/IJCNLP (1)*, 3214–3224. Association for Computational Linguistics.

Li, J.; Li, Z.; Mou, L.; Jiang, X.; Lyu, M. R.; and King, I. 2020. Unsupervised Text Generation by Learning from Search. In *NeurIPS*.

Lin, C.-Y. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*, 74–81. Barcelona, Spain: Association for Computational Linguistics.

Liu, Y.; Dou, Z.; and Liu, P. 2021. RefSum: Refactoring Neural Summarization. In *NAACL-HLT*, 1437–1448. Association for Computational Linguistics.

Liu, Y.; and Lapata, M. 2019. Text Summarization with Pretrained Encoders. In *EMNLP/IJCNLP (1)*, 3728–3738. Association for Computational Linguistics.

Marcheggiani, D.; and Titov, I. 2020. Graph Convolutions over Constituent Trees for Syntax-Aware Semantic Role Labeling. In *EMNLP (1)*, 3915–3928. Association for Computational Linguistics.

Mrini, K.; Dernoncourt, F.; Tran, Q. H.; Bui, T.; Chang, W.; and Nakashole, N. 2020. Rethinking Self-Attention: Towards Interpretability in Neural Parsing. In *EMNLP (Findings)*, 731–742. Association for Computational Linguistics.

Narayan, S.; Cohen, S. B.; and Lapata, M. 2018. Don't Give Me the Details, Just the Summary! Topic-Aware Convolutional Neural Networks for Extreme Summarization. In *EMNLP*, 1797–1807. Association for Computational Linguistics.

Paulus, R.; Xiong, C.; and Socher, R. 2018. A Deep Reinforced Model for Abstractive Summarization. In *ICLR (Poster)*. OpenReview.net.

Sandhaus, E. 2008. The New York Times Annotated Corpus.

Sankaran, B.; Mi, H.; Al-Onaizan, Y.; and Ittycheriah, A. 2016. Temporal Attention Model for Neural Machine Translation. *CoRR*, abs/1608.02927.

See, A.; Liu, P. J.; and Manning, C. D. 2017. Get To The Point: Summarization with Pointer-Generator Networks. In *ACL (1)*, 1073–1083. Association for Computational Linguistics.

Song, Z.; Meng, Z.; Zhang, Y.; and King, I. 2021a. Semi-supervised Multi-label Learning for Graph-structured Data. In *CIKM*, 1723–1733. ACM.

Song, Z.; Yang, X.; Xu, Z.; and King, I. 2021b. Graph-based Semi-supervised Learning: A Comprehensive Review. *CoRR*, abs/2102.13303.

Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to Sequence Learning with Neural Networks. In *NIPS*, 3104–3112.

Tan, J.; Wan, X.; and Xiao, J. 2017. Abstractive Document Summarization with a Graph-Based Attentional Neural Model. In *ACL (1)*, 1171–1181. Association for Computational Linguistics.

Velickovic, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph Attention Networks. In *ICLR (Poster)*. OpenReview.net.

Wang, D.; Liu, P.; Zheng, Y.; Qiu, X.; and Huang, X. 2020. Heterogeneous Graph Neural Networks for Extractive Document Summarization. In *ACL*, 6209–6219. Association for Computational Linguistics.

Wu, W.; Li, W.; Xiao, X.; Liu, J.; Cao, Z.; Li, S.; Wu, H.; and Wang, H. 2021. BASS: Boosting Abstractive Summarization with Unified Semantic Graph. In *ACL/IJCNLP (1)*, 6052–6067. Association for Computational Linguistics.

Xint, Y.; Xu, L.; Guo, J.; Li, J.; Sheng, X.; and Zhou, Y. 2021. Label Incorporated Graph Neural Networks for Text Classification. In *2020 25th International Conference on Pattern Recognition (ICPR)*, 8892–8898. Los Alamitos, CA, USA: IEEE Computer Society.

Xu, J.; and Durrett, G. 2019. Neural Extractive Text Summarization with Syntactic Compression. In *EMNLP/IJCNLP (1)*, 3290–3301. Association for Computational Linguistics.

Xu, J.; Gan, Z.; Cheng, Y.; and Liu, J. 2020. Discourse-Aware Neural Extractive Text Summarization. In *ACL*, 5021–5031. Association for Computational Linguistics.

Yang, X.; Song, Z.; King, I.; and Xu, Z. 2021. A Survey on Deep Semi-supervised Learning. *CoRR*, abs/2103.00550.

Yasunaga, M.; Zhang, R.; Meelu, K.; Pareek, A.; Srinivasan, K.; and Radev, D. R. 2017. Graph-based Neural Multi-Document Summarization. In *CoNLL*, 452–462. Association for Computational Linguistics.

Zhang, H.; and Zhang, J. 2020. Text Graph Transformer for Document Classification. In *EMNLP (1)*, 8322–8327. Association for Computational Linguistics.

Zhang, J.; Zhao, Y.; Saleh, M.; and Liu, P. J. 2020a. PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization. In *ICML*, volume 119 of *Proceedings of Machine Learning Research*, 11328–11339. PMLR.

Zhang, N.; Deng, S.; Sun, Z.; Wang, G.; Chen, X.; Zhang, W.; and Chen, H. 2019. Long-tail Relation Extraction via Knowledge Graph Embeddings and Graph Convolution Networks. In *NAACL-HLT (1)*, 3016–3025. Association for Computational Linguistics.

Zhang, T.; Kishore, V.; Wu, F.; Weinberger, K. Q.; and Artzi, Y. 2020b. BERTScore: Evaluating Text Generation with BERT. In *ICLR*. OpenReview.net.

Zhong, M.; Liu, P.; Chen, Y.; Wang, D.; Qiu, X.; and Huang, X. 2020. Extractive Summarization as Text Matching. In *ACL*, 6197–6208. Association for Computational Linguistics.

Zhou, Q.; Yang, N.; Wei, F.; Huang, S.; Zhou, M.; and Zhao, T. 2018. Neural Document Summarization by Jointly Learning to Score and Select Sentences. In *ACL (1)*, 654–663. Association for Computational Linguistics.

Zhu, H.; Lin, Y.; Liu, Z.; Fu, J.; Chua, T.; and Sun, M. 2019. Graph Neural Networks with Generated Parameters for Relation Extraction. In *ACL (1)*, 1331–1339. Association for Computational Linguistics.

Zügner, D.; Kirschstein, T.; Catasta, M.; Leskovec, J.; and Günnemann, S. 2021. Language-Agnostic Representation Learning of Source Code from Structure and Context. In *ICLR*. OpenReview.net.