Efficient Mind-Map Generation via Sequence-to-Graph and Reinforced Graph Refinement

Mengting Hu[1,2] **Honglei Guo**[3]* **Shiwan Zhao**[3] **Hang Gao**[4] **Zhong Su**[3]
[1] College of Software, Nankai University [2] Tianjin Key Laboratory of Operating System
[3] IBM Research - China [4] Institute for Public Safety Research, Tsinghua University
mthu@nankai.edu.cn, gaohang@mail.tsinghua.edu.cn
{guohl,zhaosw,suzhong}@cn.ibm.com

Abstract

A mind-map is a diagram that represents the central concept and key ideas in a hierarchical way. Converting plain text into a mind-map will reveal its key semantic structure and be easier to understand. Given a document, the existing automatic mind-map generation method extracts the relationships of every sentence pair to generate the directed semantic graph for this document. The computation complexity increases exponentially with the length of the document. Moreover, it is difficult to capture the overall semantics. To deal with the above challenges, we propose an efficient mind-map generation network that converts a document into a graph via sequence-to-graph. To guarantee a meaningful mind-map, we design a graph refinement module to adjust the relation graph in a reinforcement learning manner. Extensive experimental results demonstrate that the proposed approach is more effective and efficient than the existing methods. The inference time is reduced by thousands of times compared with the existing methods. The case studies verify that the generated mind-maps better reveal the underlying semantic structures of the document.

1 Introduction

A mind-map is a hierarchical diagram that can depict the central concept, linked with the major ideas and other ideas branch out from these (Kudelić et al., 2011; Wei et al., 2019). It is organized in cognitive structures and much easier to understand than plain text (Dhindsa et al., 2011). Thus in practice, it can be utilized for education resources, organizing, and planning. Many tools can help people *make* mind-map manually, such as FreeMind, MindGenius and Visual Mind, etc (Kudelić et al., 2011). To save human labors, some automatic methods have been proposed to *generate* mind-map from text, which focus on analyzing the semantic relations *within a sentence* by pre-defined rules (Brucks and Schommer, 2008; Rothenberger et al., 2008) or syntactic parser (Elhoseiny and Elgammal, 2012).

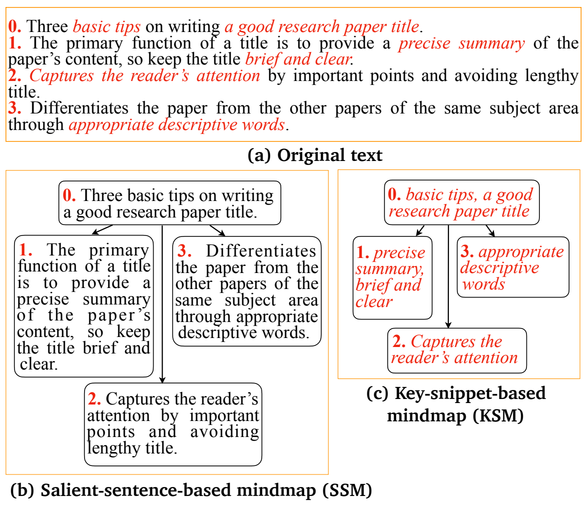

0. Three basic tips on writing a good research paper title.
1. The primary function of a title is to provide a precise summary of the paper's content, so keep the title brief and clear.
2. Captures the reader's attention by important points and avoiding lengthy title.
3. Differentiates the paper from the other papers of the same subject area through appropriate descriptive words.
(a) Original text
0. Three basic tips on writing a good research paper title.
1. The primary function of a title is to provide a precise summary of the paper's content, so keep the title brief and clear.
3. Differentiates the paper from the other papers of the same subject area through appropriate descriptive words.
2. Captures the reader's attention by important points and avoiding lengthy title.
(b) Salient-sentence-based mindmap (SSM)
0. basic tips, a good research paper title
1. precise summary, brief and clear
3. appropriate descriptive words
2. Captures the reader's attention
(c) Key-snippet-based mindmap (KSM)

Figure 1: The original text (a) is converted into mind-maps, in which a node is the entire sentence (b) or keywords (c).

Recently, researchers (Wei et al., 2019) propose to generate a mind-map automatically by detecting the semantic relation *cross sentences* in the document. It mines the structured diagram of the document, in which a node represents the meaning of a sentence in the format of the entire sentence or its keywords, and an edge represents the governing relationships between the precursor and the successor. We illustrate two types of mind-map in Figure 1, i.e. salient-sentence-based mind-map (SSM) and key-snippet-based mind-map (KSM).

Wei et al. (2019) propose a promising pipeline approach (see Figure 2(a)), which first converts the whole document into a relation graph and then prunes the extra edges to obtain the mind-map. However, the first phase tends to be less efficient since it needs to predict all the governing scores at the sentence pair level (see Figure 2(b)). The number of sentence pairs increases exponentially with

*Honglei Guo is the corresponding author.

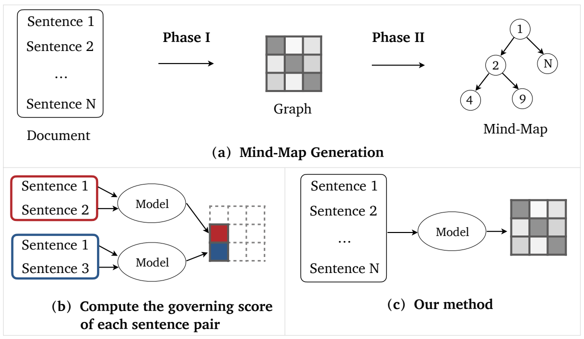


Figure 2: (a) Mind-map generation procedure. (b) MRDMF (Wei et al., 2019) predicts the relation graph at the sentence pair level. (c) Our method predicts the relation graph at the document level.

the length of the document, which raises the computational complexity. In addition, each governing score in the graph is computed separately without considering the overall semantics in the document. The sequential information of all sentences might be helpful to mine the hierarchical and structured semantics.

We propose an efficient mind-map generation network (EMGN) to address the above issues (see Figure 2(c)). The proposed method encodes all sentences sequentially and generates the graph via sequence-to-graph. It makes the first phase more efficient and can easily process multiple documents in parallel. The model training requires all the relation labels of sentence pairs in each graph. However, manually annotating costs much. We exploit DistilBert (Sanh et al., 2019) to automatically annotate a graph for each document, which provides pseudo labels to train our model. In advance, DistilBert has been fine-tuned to detect the governing relation between two sentences. The performance of DistilBert indicates it can be an "annotator" with high confidence.

Moreover, a meaningful mind-map tends to organize the major ideas of a document close to the root node. To achieve this goal, we design a graph refinement module to adjust the generated graph by using the documents with highlights. The highlights written by the editors summarize the key ideas of a document. During training, we leverage this prior human knowledge as a reference to refine the governing scores in the graph via self-critical reinforcement learning (Rennie et al., 2017).

In summary, the main contributions of this paper are as follows.

- We propose an efficient mind-map method that can consider the document-level semantics by sequence-to-graph.
- In the training phase, we design a graph refinement module to refine the generated graph by leveraging the manual highlights and self-critical reinforcement learning.
- Extensive experimental results demonstrate the proposed method can generate a better-performed mind-map efficiently. The inference time is reduced by thousands of times compared with the existing approaches.

2 Methodology

2.1 Problem Definition

Assume a document has N sentences $D = \{s_k\}_{k=1}^{N}$, each sentence is a word sequence $s_k = \{w_k^1, w_k^2, ..., w_k^{L_k}\}$, where L_k is the length of the sentence. We define the mind-map generation as a two-phase task.

$$D \rightarrow \mathbf{G} \rightarrow M \tag{1}$$

where the input text D is first processed to obtain the relation graph $\mathbf{G}$. Then the graph is pruned to gain the final mind-map M.

The detailed methodology for this two-phase task is described in the following sections. Concretely, we depict the network architecture for the first phase $D \rightarrow \mathbf{G}$ in Figure 3. We generate the relation graph from a document by graph detector (§2.2). The graph is simultaneously refined to make the generated mind-map more meaningful (§2.3). For the second phase $\mathbf{G} \rightarrow M$, we generate two types of mind-maps based on the graph (§2.5).

2.2 Graph Detector

As shown in Figure 3, the graph detector aims to extract the relation graph for an input document. It considers the overall semantics and obtains the graph efficiently.

2.2.1 Sentence Encoder

Given a sentence s_k, we first map it into an embedding sequence $\{\boldsymbol{e}_k^1, \boldsymbol{e}_k^2, ..., \boldsymbol{e}_k^{L_k}\}$ through a pre-trained embedding matrix GloVE (Pennington et al., 2014). Then we exploit a Bi-directional LSTM (BiLSTM) (Graves et al., 2013) to encode the embedding sequence into the hidden states $\{\boldsymbol{h}_k^1, \boldsymbol{h}_k^2, ...\boldsymbol{h}_k^{L_k}\}$. To compute the vector representation for each sentence, we apply a simple

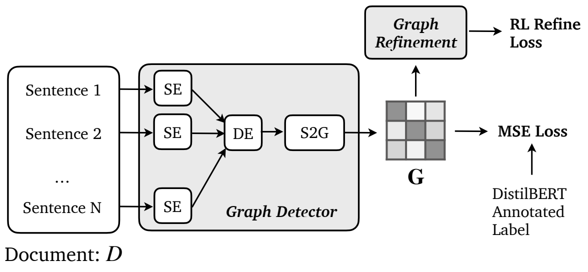


Figure 3: The network architecture of the proposed approach for converting the document to a graph (Phase I). SE, DE, and S2G refer to the sentence encoder, document encoder, and sequence-to-graph modules, respectively.

max-pooling operation over the hidden states.

$$s_k = \max(h_k^1, ..., h_k^{L_k}) \tag{2}$$

2.2.2 Document Encoder

The sequential information of sentences indicates the semantic coherence and logical structure of a document. This information is essential in understanding the entire document and extracting a clear mind-map. To model the sentence-level context, we encode the vector representations of all sentences $\{s_k\}_{k=1}^N$ with another BiLSTM and obtain $\mathbf{H} = \{h_1, h_2, ..., h_N\}$.

2.2.3 Sequence-to-Graph

In a graph $\mathbf{G}$, a node represents a sentence from the document. $\mathbf{G}_{i,j}$ is the governing score between sentence s_i and s_j, which indicates the probability that s_i semantically implies s_j. Thus the graph is directed since the governing relationships are different between $\mathbf{G}_{i,j}$ and $\mathbf{G}_{j,i}$. Inspired by (Dozat and Manning, 2017; Zhang et al., 2019), we utilize sequence-to-graph to process the sentence-level sequence into graph efficiently. Concretely, we first compute the representations of all sentences when they are regarded as the start or end nodes in the edges. Exploiting separate parameters help learn distinct representations for a sentence.

$$\begin{aligned} h_i^{(\text{start})} &= \text{MLP}^{(\text{start})}(h_i) \\ h_j^{(\text{end})} &= \text{MLP}^{(\text{end})}(h_j) \end{aligned} \tag{3}$$

where MLP is a linear transformation. Then we calculate the governing scores in $\mathbf{G} \in \mathbb{R}^{N \times N}$ with a bilinear operation or biaffine operation.

$$\begin{aligned} \mathbf{G}_{i,j} &= \text{Bilinear}(h_i^{(\text{start})}, h_j^{(\text{end})}) \\ \mathbf{G}_{i,j} &= \text{Biaffine}(h_i^{(\text{start})}, h_j^{(\text{end})}) \end{aligned} \tag{4}$$

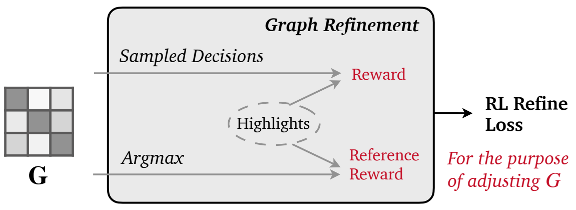


Figure 4: The idea of the graph refinement module aims to adjust the governing scores in $\mathbf{G}$ with the help of highlights. Sampling decisions and greedily selecting by argmax are both consistent with the mind-map detector (§2.5). This builds a bridge between the learning of a graph (Phase I) and extracting a mind-map from the graph (Phase II).

where Bilinear and Biaffine are defined as below.

$$\text{Bilinear}(x_1, x_2) = \sigma(x_1 \mathbf{U} x_2 + b)$$
$$\text{Biaffine}(x_1, x_2) = \sigma(x_1 \mathbf{U} x_2 + \mathbf{W}[x_1; x_2] + b)$$

where $\mathbf{U}$ and $\mathbf{W}$ are the parameter matrix, b is the bias. σ is the sigmoid operation, guaranteeing that each governing score is between 0 and 1.

2.3 Graph Refinement

According to (Buzan, 2006), a mind-map is organized as a tree structure with the central concept as its root node, and the major ideas are connected directly to the root. Other ideas branch out from the major ideas. Therefore, a clear mind-map tends to organize the main opinions of a document close to the root node. To achieve this goal, we leverage the human-written highlights to refine the graph $\mathbf{G}$ via reinforcement learning (RL) algorithm (Williams, 1992), more specifically, self-critical RL (Rennie et al., 2017). The main idea is depicted in Figure 4.

Concretely, the graph detector module can be considered as an *agent* that follows a policy function to decide an action given a state. We regard an input document as the *state* and the extracted graph $\mathbf{G}$ as the *action* distribution. After we sample selected sentences over the graph, a delayed reward is calculated that indicates the similarity between selected sentences and highlights. Maximizing the expected reward helps to refine the governing scores in the graph $\mathbf{G}$. Next, we will introduce the graph refinement module in detail.

Policy The policy is described as below.

$$\text{Sampled Decisions} \sim \pi_\Theta(\mathbf{G}|D) \tag{5}$$

where Θ is the parameters of the graph detector, D is the document and $\mathbf{G}$ is the extracted graph. We sample sentences over the graph as follows.

Sampled Decisions The main reason why RL can improve the reward is that it accords with the *trial-and-error* process, which samples and update the parameters accordingly. To bridge with the strategy in the second phase (§2.5), i.e. detecting mind-map from a graph, we sample the upper nodes given the graph in the same way. At first, we sample a sentence as the root node of the mind-map.

$$\boldsymbol{g_0} = \mathrm{softmax}(\mathrm{rowsum}(\mathbf{G})) \tag{6}$$

where rowsum is row-wise summation. Its result is the salience score that a sentence governs all other sentences. A larger salience score indicates that a sentence is more likely to represent the key ideas of the document. We sample a root node based on multinomial distribution $\boldsymbol{g_0}$. Next, we remove the sampled root from the graph and cluster the remaining nodes into two sets, obtaining two subgraphs, i.e., $\mathbf{G}_1$ and $\mathbf{G}_2$. Similar with the root node, its two child nodes are also sampled based on the distributions $\boldsymbol{g_1} = \mathrm{softmax}(\mathrm{rowsum}(\mathbf{G}_1))$ and $\boldsymbol{g_2} = \mathrm{softmax}(\mathrm{rowsum}(\mathbf{G}_2))$, respectively.

The reason why we sample three sentences is that the average number of sentences in highlights is around 3.55. We also found that sampling more nodes does not improve performance. A possible reason is that more upper nodes introduce noise when comparing with highlights.

Reward The definition of reward is crucial for RL as it determines the optimizing objective. To ensure that the upper nodes of the mind-map represent the central concept and major ideas of the document, we treat the manual highlights as a reference. The ROUGE score (Lin, 2004) between the sampled decisions and the highlights A is used to define the reward. Multiple variants of the ROUGE score are proposed (Lin, 2004). Among them, ROUGE-1 (R-1), ROUGE-2 (R-2), ROUGE-L (R-L) are the most commonly utilized. We employ the average of ROUGE variants to define a reward function.

$$\mathrm{Sim}(X, A) = \frac{\text{R-1}(X, A) + \text{R-2}(X, A) + \text{R-L}(X, A)}{3} \tag{7}$$

Assume the sampled sentences are D_s and $D_s \subseteq D$, the reward is computed as follows.

$$r = \mathrm{Sim}(D_s, A) \tag{8}$$

RL Refine Loss According to (Sutton et al., 2000), RL training objective is to maximize the expected reward. Therefore, we define the RL loss for graph refinement is to minimize the negative reward (see Eq. (9)). More concretely, assume the sampled sentences $D_s = \{a_0, a_1, a_2\}$, where a_0 is the root, a_1 and a_2 are independent child nodes. Based on the conditional independence (Dawid, 1979), we have

$$p(D_s) = p(a_0a_1a_2) = p(a_0)p(a_1|a_0)p(a_2|a_0)$$

Therefore, $p(D_s) = g_0g_1g_2$, where g_i is the probability of the sampled sentence in $\boldsymbol{g_i}$. When we only sample one sentence as the root node, $p(D_s) = g_0$.

$$\mathcal{L}_r = -r \cdot p(D_s) = -r \prod_i g_i \tag{9}$$

To reduce the variance caused by sampling, we associate the reward with a reference reward or baseline (Rennie et al., 2017) and define it as $b = \mathrm{Sim}(D_b, A)$. D_b is the sentences chosen greedily by the argmax operation on the multinomial distributions. With the likelihood ratio trick, the optimizing objective can be formulated as.

$$\mathcal{L}_r = -(r - b) \sum_i \log(g_i) \tag{10}$$

Algorithm 1 Graph Detector Training Process

```
Input: Training data $\mathcal{B}$: $\{B^1, B^2, ..., B^K\}$
 1: Randomly initialize $\Theta$
 2: repeat
 3:     for $B^k \in \mathcal{B}$ do
 4:         Calculate graphs by graph detector
 5:         Initialize temp batch loss $\hat{\mathcal{L}} \leftarrow 0$
 6:         for each document in $B^k$ do
 7:             Compute $\mathcal{L}_g$ by Eq. (11)
 8:             Compute $\mathcal{L}_r$ by Eq. (10)
 9:             Compute joint loss $\mathcal{L}$ by Eq. (12)
10:             Update temp batch loss $\hat{\mathcal{L}} \leftarrow \hat{\mathcal{L}} + \mathcal{L}$
11:         Optimize $\Theta$ by $\hat{\mathcal{L}}/|B^k|$
12: until performance on the validation set does not improve in 3 epochs
```

2.4 Training

We train the graph detector module by a combination of two optimizing objectives, i.e., fitting the pseudo graphs annotated by DistilBert and refining the generated graphs.

Since it costs too much to manually annotating the relation labels in the graph, we automatically annotate a pseudo graph $\mathbf{Y}$ by DistilBert (Sanh et al., 2019). In advance, DistilBert is fine-tuned by sentence pairs constructed from news articles. In this way, our method can take advantage of the prior knowledge from the pre-trained model, but also the local semantic association of sentence pairs. The fine-tuning details of DistilBert will be introduced in §3.2.1. The proposed model fits the pseudo graph by a mean square error (MSE) loss.

$$\mathcal{L}_g = \frac{1}{N^2} \sum_i \sum_j (\mathbf{G}_{i,j} - \mathbf{Y}_{i,j})^2 \qquad (11)$$

where $\mathbf{Y}$ is the pseudo graph.

Then we combine the MSE loss and graph refinement as an overall training objective to optimize the parameters Θ. The entire training process of the proposed model is described in Algorithm 1.

$$\mathcal{L} = \mathcal{L}_g + \lambda \mathcal{L}_r \qquad (12)$$

where λ balances the effect of graph refinement.

2.5 Mind-Map Detector

In this section, we introduce how to generate a mind-map from a graph, i.e. $\mathbf{G} \rightarrow M$. The graph $\mathbf{G}$ covers all the sentences in the document, which might be redundant. To highlight the major ideas, we convert the graph into a mind-map through the strategy proposed by (Wei et al., 2019) to prune the extra edges. The algorithm works recursively to determine the governing relationship of sentences. First, it chooses a governor by picking the highest row-wise aggregation scores in the graph. Then except for the governor, it clusters the remaining nodes into two sub-groups with k-means algorithm. The sub-groups are processed recursively to extract the final mind-map. We enclose the full algorithm in the Appendix.

We extract two types of mind-map, i.e. salient-sentence-based mind-map (SSM) and key-snippet-based mind-map (KSM). Given the graph $\mathbf{G}$ of a document, we first prune it into SSM, and then extract the key phrases in each sentence (Rose et al., 2010) to obtain the KSM. Therefore, SSM and KSM have the same structure. The only difference is that a node in SSM is a sentence, while a node in KSM is the key phrases. In the case of KSM, if no key phrase is found, the whole sentence is kept in the mind-map.

3 Experiments

3.1 Dataset

We build an evaluation benchmark with 135 articles, which are randomly selected from CNN news articles (Hermann et al., 2015; Cheng and Lapata, 2016). The size of the benchmark is about 98,181 words. The average length of the news article is about 727 words. Two experts manually annotate the ground-truth mind-maps for these articles. If one of the experts disagrees with any content of the mind-map, they discuss to reach consensus. In the experiments, the benchmark is split into two datasets: a testing set $\mathcal{D}_t$ with 120 articles and a validation set $\mathcal{D}_v$ with 15 articles.

3.2 Experimental Settings

3.2.1 Automatically Annotate Graphs for EMGN Training

Sentence Pairs for Fine-tuning DistilBert To save time in fine-tuning and subsequent annotation, we choose DistilBert as the *"annotator"* to obtain the relationships of all sentence pairs in the graph.

To construct the training pairs for fine-tuning DistilBert, we first randomly select 90k CNN news articles $\mathcal{D}_{news}$, which has no overlap with the benchmark. Each news consists of content and highlights. Because highlights summarize the major concepts of the news, they are regarded as the governors. To find the sentence pairs with governing relationships, we exploit TFIDF as the similarity metric. Concretely, a highlight governs each sentence in a paragraph when it is similar to one or some sentences in the paragraph. The negative samples are generated randomly.

In this way, we build a large-scale training corpus, which has 641,476 pairs from these news articles. Then we split all the pairs into 600k for training and 41,476 for testing.

Fine-tuning DistilBert Using the training pairs, we fine-tune DistilBert for 3 epochs with a learning rate of 5e-5 and a training batch size of 32. The accuracy and F1 on the testing pairs are both more than 99.35%. Thus DistilBert can annotate pseudo graphs with high confidence.

Annotate Pseudo Graphs We select 44,450 articles from $\mathcal{D}_{news}$ by setting the max length of sentence in the article as 50 and max number of sentences as 50[1]. After annotating these articles by

[1]According to our statistical analysis on the CNN dataset, the average length of articles is 33.87 sentences and the average length of sentences is 21.25 words.

Models		SSM				KSM			
		R-1	R-2	R-L	Avg	R-1	R-2	R-L	Avg
Compared Methods	Random	32.71	23.51	30.08	28.77	29.73	26.50	29.67	28.63
	LexRank	34.53	25.04	31.79	30.45	31.04	27.75	31.00	29.93
	MRDMF	38.19	29.51	35.72	34.47	33.18	30.26	33.08	32.18
	DistilBert	42.15	33.34	39.66	38.38	40.00	36.92	39.92	38.95
Model Variants	EMGN(root)	46.04	38.05	43.73	42.61	43.28	**40.69**	43.23	42.40
	EMGN(root)+greedy	43.27	35.11	40.93	39.77	40.30	37.62	40.24	39.39
	EMGN-GR	45.06†	37.08†	42.75†	41.63†	41.62†	39.07†	41.57†	40.75†
	EMGN(biaffine)	45.73	37.62	43.35	42.23	42.90	40.15	42.84	41.96
Full Model	EMGN	**46.14**‡	**38.21**‡	**43.84**‡	**42.73**‡	**43.33**‡	40.67‡	**43.28**‡	**42.43**‡

Table 1: Evaluation results of the salient-sentence-based mind-map (SSM) and key-snippet-based mind-map (KSM) in terms of R-1 (%), R-2 (%), R-L (%) and the average score (%). The marker † refers to p-value<0.01 when comparing with DistilBert. The marker ‡ refers to p-value<0.01 when comparing with EMGN-GR.

DistilBert, they are exploited to train our mind-map generation model EMGN.

3.2.2 Mind-Map Evaluation

We evaluate the generated mind-map by *comparing the tree similarity with the human-annotated mind-map* (Wei et al., 2019). We first remove the weak edges from a generated mind-map to ensure that it has the same number of edges as the annotation. The similarity between two edges is computed as below. We utilize Sim as Eq. (7).

$$f(s_i \rightarrow s_j, s_a \rightarrow s_b) = \frac{\mathrm{Sim}(s_i, s_a) + \mathrm{Sim}(s_j, s_b)}{2}$$

Then for each edge in the annotation, the strategy finds the most similar edge in the generated mind-map. The final score is the average similarity score of all greedily selected pairs.

3.2.3 Implementation Details

We initialize the word embeddings with 50-dimension GloVE (Pennington et al., 2014) and fine-tune during training. All other parameters are initialized by sampling from a normal distribution of $\mathcal{N}(0, 0.02)$. The hidden size of BiLSTM is set to be 25×2. The models are optimized by Adam (Kingma and Ba, 2015) with a learning rate of 1e-4. The batch size is 64. And λ is set to 0.01. We employ an early stop strategy during training if the evaluation score on the validation set $\mathcal{D}_v$ does not improve in 3 epochs, and the best model is selected for evaluating testing set $\mathcal{D}_t$. For all baselines and our model, the reported results are the average score of 5 runs.

The full results are presented in the Appendix.

3.3 Experimental Methods

Compared Methods We validate the effectiveness of the proposed method by comparing with the following baselines.

- **Random**: We randomly sample a graph $\mathbf{G}$ for an input document. Each governing score $\mathbf{G}_{i,j}$ ranges from zero to one.
- **LexRank**: It computes the governing score of sentence pair by the cosine similarity of their TFIDF vectors. It follows the well-known LexRank algorithm (Erkan and Radev, 2004), which is an extension of PageRank in the document summarization domain.
- **MRDMF** (Wei et al., 2019): This is the state-of-the-art semantic mind-map generation work. It presents a multi-perspective recurrent detector to extract the governing relationship and then prunes the extra edges.
- **DistilBert** (Sanh et al., 2019): It is a lighter version of BERT (Devlin et al., 2019). It provides the pseudo graphs for our method training.

Method Variants The proposed full model is the efficient mind-map generation network (EMGN), with bilinear operation in the sequence-to-graph module. We explore the impact of individual modules by comparing with its variants. Minus (-) means removing the module from the full model.

- **EMGN(root)**: It only samples root node for refining the graph.
- **EMGN(root)+greedy**: It chooses root node by greedily selecting the sentence with maximum similarity with highlights.

Models	SSM Avg		KSM Avg	
	≤ 25	> 25	≤ 25	> 25
Random	31.94	26.63	32.53	26.22
LexRank	33.54	28.07	33.99	27.28
MRDMF	39.83	30.58	36.76	28.69
DistilBert	44.36	34.34	44.27	34.89
EMGN(root)	49.91	37.32	49.39	37.24
EMGN-GR	49.22	36.22	46.81	36.14
EMGN(biaffine)	49.54	37.00	48.36	37.13
EMGN	**50.23**	**37.38**	**49.48**	**37.44**

Table 2: Evaluation results by splitting the testing set with the number of sentences in a document. Among all 120 files in $\mathcal{D}_t$, there are 50 files with the number of sentences ≤ 25, and 70 files with the number > 25.

Models	$\mathcal{D}_t$	$\mathcal{D}_v$
LexRank	349.21	95.24
MRDMF	467.07	62.49
DistilBert	1219.51	201.04
EMGN-GR	0.13	0.02
EMGN(root)	3.18	0.44
EMGN(root) w/o sample	0.14	0.02
EMGN	10.27	1.42
EMGN w/o sample	0.15	0.02

Table 3: Total inference time (second) in Phase I of different methods. "w/o sample" indicates that removing the sampling in graph refinement module since we do not need to sample during evaluation.

- **EMGN-GR**: It removes the graph refinement (GR) module from EMGN, which lefts the graph detector module for the purpose of sequence-to-graph.
- **EMGN(biaffine)**: It computes the graph by biaffine operation in EMGN.

3.4 Experimental Results

Overall Results The experimental results for two types of mind-maps are displayed in Table 1. Firstly, we find that our method EMGN significantly outperforms MRDMF by 8.26% on SSM and 10.25% on KSM in terms of the average score. This indicates the effectiveness of EMGN. Then comparing DistilBert and EMGN, we can see that EMGN achieves significant improvements. This shows that the proposed method successfully exploits the pseudo labels and further improves the performances. Finally, we observe that DistilBert consistently outperforms MRDMF. Thus DistilBert is more effective in matching sequences than the multi-hop attention of MRDMF. Annotating pseudo graphs by DistilBert has higher confidence, which contributes to the subsequent learning of the proposed approach.

Compare with Model Variants We further investigate the impact of individual components (see the second part of Table 1). We observe that EMGN-GR significantly outperforms DistilBert. By leveraging the sequential information of the document and early stop strategy, EMGN-GR can prevent overfitting the pseudo graphs and extract better mind-maps than DistilBert. By comparing EMGN-GR and EMGN, we found that EMGN significantly outperforms EMGN-GR. This verifies that the graph refinement module can successfully refine the graph to obtain a meaningful mind-map.

Then, by comparing EMGN(root) and EMGN(root)+greedy, we see that EMGN(root) gains many improvements. A possible reason is that EMGN(root)+greedy only greedily enlarging the governing scores in the graph for a specific sentence, i.e. the one which has the maximum ROUGH similarity with highlights. This might ignore the exploration on other nodes. EMGN(root) performs better by sampling more sentences and compares them relatively. Finally, we found that EMGN performs slightly better than EMGN(root). This shows that refining more upper nodes achieves better performance than only refining the root node.

Effects of the Document Length Table 2 displays the evaluation results by splitting the testing set $\mathcal{D}_t$ with the document length, i.e. the number of sentences in a document. We found that EMGN consistently achieves the best performances. By comparing the results on two subsets, we can see that the results are highly related to the number of sentences in the article. It is still very challenging to extract meaningful mind-maps for the longer articles.

3.5 Further Analysis

Inference Time To validate the efficiency of the proposed method, we compare the inference time of the testing set $\mathcal{D}_t$ and validation set $\mathcal{D}_v$ (see Table 3). Since all methods share Phase II, we only report the inference time of Phase I in Table 3 to show the merits of the proposed method. The total inference time of Phase II is around 23.54 seconds in D_t and 2.94 seconds in D_v.

We set the batch size of MRDMF and DistilBert as 256 (256 sentence pairs in a batch). The batch size of EMGN related methods is 32 (32 documents

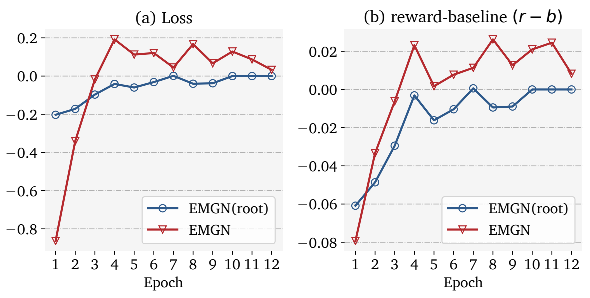
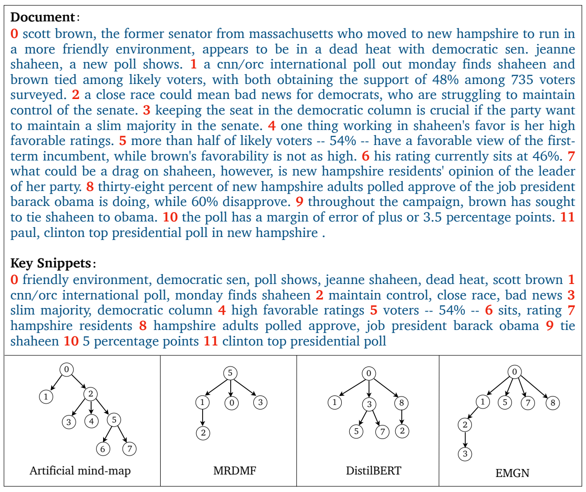


Figure 5: The loss and reward curves in the graph refinement module.

in a batch). We observe that the inference time of the existing methods, e.g. MRDMF and DistilBert, are more than 3,000 times compared with our method. As depicted in Figure 2, the main reason is that we significantly reduce the computational complexity of a relation graph from the sentence pair level to the document level.

Loss and Reward in Graph Refinement The graph refinement aims to optimize the upper nodes of the mind-map for better revealing the major ideas of the document. We achieve this goal by optimizing the reward of sampled decisions. In Figure 5, we plot the average loss $\mathcal{L}_r$ and average reward (Eq. 10) in each epoch of the training process. In Figure 5(a), it can be seen that the loss $\mathcal{L}_r$ gradually converges as training in both EMGN(root) and EMGN. Also in Figure 5(b), the reward are gradually increasing as the training epochs and finally reach a relatively stable value. The training curves further prove that the proposed graph refinement module helps improve the similarity between the upper nodes and human-written highlights.

Case Study In Figure 6, we depict the generated mind-maps by varied methods for a CNN news[2]. Comparing with the artificial mind-map, MRDMF chooses an inaccurate sentence as the root node. DistilBert and EMGN both generate the mind-map which represents the major ideas of the document. However, some relations between nodes in DistilBert are meaningless, such as the governing relation from sentence 8 to sentence 2. EMGN generates a mind-map that captures the central concept and grasps the directed relations among sentences. This is because our method considers the sequential information of an article and understands it as a whole. The case study further verifies that our method effectively generates a mind-map to reveal



Document:
0 scott brown, the former senator from massachusetts who moved to new hampshire to run in a more friendly environment, appears to be in a dead heat with democratic sen. jeanne shaheen, a new poll shows. 1 a cnn/orc international poll out monday finds shaheen and brown tied among likely voters, with both obtaining the support of 48% among 735 voters surveyed. 2 a close race could mean bad news for democrats, who are struggling to maintain control of the senate. 3 keeping the seat in the democratic column is crucial if the party want to maintain a slim majority in the senate. 4 one thing working in shaheen's favor is her high favorable ratings. 5 more than half of likely voters -- 54% -- have a favorable view of the first-term incumbent, while brown's favorability is not as high. 6 his rating currently sits at 46%. 7 what could be a drag on shaheen, however, is new hampshire residents' opinion of the leader of her party. 8 thirty-eight percent of new hampshire adults polled approve of the job president barack obama is doing, while 60% disapprove. 9 throughout the campaign, brown has sought to tie shaheen to obama. 10 the poll has a margin of error of plus or 3.5 percentage points. 11 paul, clinton top presidential poll in new hampshire .

Key Snippets:
0 friendly environment, democratic sen, poll shows, jeanne shaheen, dead heat, scott brown 1 cnn/orc international poll, monday finds shaheen 2 maintain control, close race, bad news 3 slim majority, democratic column 4 high favorable ratings 5 voters -- 54% -- 6 sits, rating 7 hampshire residents 8 hampshire adults polled approve, job president barack obama 9 tie shaheen 10 5 percentage points 11 clinton top presidential poll

Figure 6: Case study for SSM and KSM.

the underlying semantic structure for a document.

4 Related Works

A mind-map is a hierarchical diagram to reveal the semantic structure for a document. The salient-sentence-based mind-map (SSM) is similar with extractive summarization (Zhong et al., 2020), which aims to choose some key sentences from the document to describe the main ideas. Similar but completely different, mind-map generation can reveal not only the main ideas, but also the key semantic logic structure of the document.

One previous work LexRank (Erkan and Radev, 2004) computes an adjacency matrix of the graph representation of sentences based on intra-sentence cosine similarity. However, the lexical similarity of some sentence pairs with semantic relation may be zero. Generating a mind-map from this graph representation tends to be less-meaningful, which is also indicated in the experiments (see §3.4). In addition, a few extractive summarization works employ graph techniques. For instance, a bipartite graph for sentence and entity nodes (Parveen and Strube, 2015) or a weighted graph with topic nodes (Parveen et al., 2015) are proposed to improve ranking the sentences in a document. Recently, Wang et al. (2020) propose to build a heterogeneous graph to learn the correlation between word and sentence nodes, which helps to select better-summarizing sentences. Though these works involve learning the graph knowledge, such graphs are hard to derive a mind-map that focuses on the governing relationships between sentences.

Another related direction is the policy-based reinforcement learning (Williams, 1992; Sutton et al.,

[2]Article is available at https://www.cnn.com/2014/09/15/politics/new-hampshire-senate-poll/index.html

2000). Previous methods (Xiong et al., 2017; Xiao et al., 2020) usually affect the training of the main task by a policy network with separate parameters. Different from them, we directly regard the main network as the policy network and its output graph as the action distribution. Then the main network is optimized simultaneously when maximizing the expected reward.

5 Conclusion

We propose an efficient mind-map generation network that converts a document into a graph via sequence-to-graph. To ensure a meaningful mind-map, we design a graph refinement module to adjust the graph by leveraging the highlights in a reinforcement learning manner. Extensive experimental results demonstrate that the proposed approach is more effective and efficient than the existing methods. The inference time is reduced by thousands of times compared with the existing approaches. The case studies further verify that the generated mind-map can reveal the underlying semantic structures of a document.

Acknowledgements

We sincerely thank all the anonymous reviewers for providing valuable feedback. This work is supported by the National Science and Technology Major Project, China (Grant No. 2018YFB0204304).

References

Claudine Brucks and Christoph Schommer. 2008. Assembling actor-based mind-maps from text stream. *arXiv preprint arXiv:0810.4616*.

Tony Buzan. 2006. The mind map book. Penguin Books.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 484–494.

A Philip Dawid. 1979. Conditional independence in statistical theory. *Journal of the Royal Statistical Society: Series B (Methodological)*, 41(1):1–15.

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 (NAACL-HLT)*, pages 4171–4186.

Harkirat S Dhindsa, O Roger Anderson, et al. 2011. Constructivist-visual mind map teaching approach and the quality of students' cognitive structures. *Journal of Science Education and Technology*, 20(2):186–200.

Timothy Dozat and Christopher D Manning. 2017. Deep biaffine attention for neural dependency parsing. In *The 5th International Conference on Learning Representations (ICLR)*, pages 1–8.

Mohamed Elhoseiny and Ahmed Elgammal. 2012. English2mindmap: An automated system for mindmap generation from english text. In *2012 IEEE International Symposium on Multimedia*, pages 326–331. IEEE.

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.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *The 38th International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 6645–6649.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1693–1701.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *The 3rd International Conference on Learning Representations (ICLR)*.

Robert Kudelić, Mladen Konecki, and Mirko Maleković. 2011. Mind map generator software model with text mining algorithm. In *Proceedings of the 33rd International Conference on Information Technology Interfaces (ITI)*, pages 487–494. IEEE.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81.

Daraksha Parveen, Hans-Martin Ramsl, and Michael Strube. 2015. Topical coherence for graph-based extractive summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1949–1954.

Daraksha Parveen and Michael Strube. 2015. Integrating importance, non-redundancy and coherence in graph-based extractive summarization. In *Proceedings of the 24th International Conference on Artificial Intelligence (IJCAI)*, page 1298–1304.

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

Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. 2017. Self-critical sequence training for image captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7008–7024.

Stuart Rose, Dave Engel, Nick Cramer, and Wendy Cowley. 2010. Automatic keyword extraction from individual documents. *Text mining: applications and theory*, 1:1–20.

T Rothenberger, S Oez, E Tahirovic, and Christoph Schommer. 2008. Figuring out actors in text streams: Using collocations to establish incremental mind-maps. *arXiv preprint arXiv:0803.2856*.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. In *The 5th Workshop on Energy Efficient Machine Learning and Cognitive Computing of NeurIPS*.

Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1057–1063.

Danqing Wang, Pengfei Liu, Yining Zheng, Xipeng Qiu, and Xuanjing Huang. 2020. Heterogeneous graph neural networks for extractive document summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 6209–6219.

Yang Wei, Honglei Guo, Jinmao Wei, and Zhong Su. 2019. Revealing semantic structures of texts: Multi-grained framework for automatic mind-map generation. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 5247–5254.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Ya Xiao, Chengxiang Tan, Zhijie Fan, Qian Xu, and Wenye Zhu. 2020. Joint entity and relation extraction with a hybrid transformer and reinforcement learning based model. *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 9314–9321.

Wenhan Xiong, Thien Hoang, and William Yang Wang. 2017. Deeppath: A reinforcement learning method for knowledge graph reasoning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 564–573.

Sheng Zhang, Xutai Ma, Kevin Duh, and Benjamin Van Durme. 2019. AMR parsing as sequence-to-graph transduction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 80–94.

Ming Zhong, Pengfei Liu, Yiran Chen, Danqing Wang, Xipeng Qiu, and Xuanjing Huang. 2020. Extractive summarization as text matching. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 6197–6208.

Models	SSM			
	R-1 (%)	R-2 (%)	R-L (%)	Avg (%)
Random	32.71 ± 0.44	23.51 ± 0.40	30.08 ± 0.40	28.77 ± 0.41
LexRank	34.53 ± 0.06	25.04 ± 0.07	31.79 ± 0.06	30.45 ± 0.06
MRDMF	38.19 ± 0.29	29.51 ± 0.35	35.72 ± 0.30	34.47 ± 0.31
DistilBert	42.15 ± 0.19	33.34 ± 0.25	39.66 ± 0.21	38.38 ± 0.22
EMGN(root)	46.04 ± 0.15	38.05 ± 0.20	43.73 ± 0.18	42.61 ± 0.17
EMGN(root)+greedy	43.27 ± 0.87	35.11 ± 1.01	40.93 ± 0.92	39.77 ± 0.93
EMGN-GR	45.06 ± 0.64	37.08 ± 0.61	42.75 ± 0.63	41.63 ± 0.63
EMGN(biaffine)	45.73 ± 0.70	37.62 ± 0.91	43.35 ± 0.79	42.23 ± 0.80
EMGN	**46.14 ± 0.15**	**38.21 ± 0.21**	**43.84 ± 0.17**	**42.73 ± 0.17**

Table 4: Full evaluation results of salient-sentence-based mind-map (SSM).

Models	KSM			
	R-1 (%)	R-2 (%)	R-L (%)	Avg (%)
Random	29.73 ± 0.51	26.50 ± 0.57	29.67 ± 0.51	28.63 ± 0.53
LexRank	31.04 ± 0.33	27.75 ± 0.37	31.00 ± 0.34	29.93 ± 0.35
MRDMF	33.18 ± 0.43	30.26 ± 0.38	33.08 ± 0.43	32.18 ± 0.41
DistilBert	40.00 ± 0.47	36.92 ± 0.45	39.92 ± 0.48	38.95 ± 0.47
EMGN(root)	43.28 ± 0.03	**40.69 ± 0.17**	43.23 ± 0.03	42.40 ± 0.07
EMGN(root)+greedy	40.30 ± 0.61	37.62 ± 0.53	40.24 ± 0.61	39.39 ± 0.58
EMGN-GR	41.62 ± 0.96	39.07 ± 0.84	41.57 ± 0.96	40.75 ± 0.92
EMGN(biaffine)	42.90 ± 0.64	40.15 ± 0.77	42.84 ± 0.64	41.96 ± 0.68
EMGN	**43.33 ± 0.38**	40.67 ± 0.39	**43.28 ± 0.37**	**42.43 ± 0.38**

Table 5: Full evaluation results of key-snippet-based mind-map (KSM).

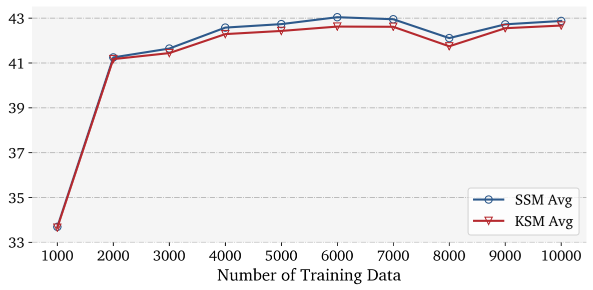


Figure 7: Effects of training data scale for EMGN.

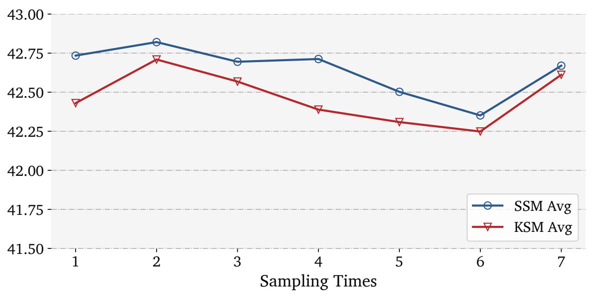


Figure 8: Effects of sampling times for EMGN.

A Software and Hardware

We use Pytorch to implement all models (Python 3.5). The operating system is Red Hat Enterprise Linux 7.8. DistilBert is trained on Tesla K80. All other models are trained on GTX 980.

We compare the inference time of all the models in the same software and hardware environments.

B Results Appendix

B.1 Full Results with Standard Deviations

As shown in Table 4, Table 5 and Table 6, we display the full experimental results, including the average score and the standard deviation of 5 runs.

B.2 Effects of Training Data Scale

We also investigate the impact of the training data size on the performance. We totally annotate pseudo graph labels for 44,450 documents by DistilBert. The performance curves of EMGN with different training scale are depicted in Figure 7. It shows that training the proposed model EMGN does not require too many labeled documents. The performance scores are improved significantly by changing the data scale from 1000 to 2000. The results grow steadily as adding more training data. A possible explanation is that compared with the ground-truth graph, the pseudo graph labels by DistilBert are still less accurate and might have redun-

Models	SSM Avg		KSM Avg	
	≤ 25	> 25	≤ 25	> 25
Random	31.94 ± 1.84	26.63 ± 0.82	32.53 ± 1.87	26.22 ± 0.81
LexRank	33.54 ± 0.25	28.07 ± 0.09	33.99 ± 0.27	27.28 ± 0.07
MRDMF	39.83 ± 0.18	30.58 ± 0.23	36.76 ± 0.24	28.69 ± 0.27
DistilBert	44.36 ± 0.34	34.34 ± 0.14	44.27 ± 0.78	34.89 ± 0.39
EMGN(root)	49.91 ± 0.39	37.32 ± 0.34	49.39 ± 0.49	37.24 ± 0.49
EMGN-GR	49.22 ± 0.48	36.22 ± 0.47	46.81 ± 1.05	36.14 ± 0.53
EMGN(biaffine)	49.54 ± 0.68	37.00 ± 0.64	48.36 ± 0.56	37.13 ± 0.75
EMGN	**50.23 ± 0.59**	**37.38 ± 0.24**	**49.48 ± 0.52**	**37.44 ± 0.61**

Table 6: Full evaluation results of splitting testing set with the number of sentences in a document.

Algorithm 2 Salient-Sentence-based Mind-map Generator.

```
Require: a document $D$
Ensure: the nodes $\mathbf{C}_s$, and the edges $\mathbf{E}_s$
 1: function RECURSIVE($\mathbf{G}$,$\mathbf{C}$,$\mathbf{E}$,$root$)
 2:     $k \leftarrow length(\mathbf{G})$
 3:     $\boldsymbol{g} \leftarrow \text{rowsum}(\mathbf{G})$
 4:     $governor \leftarrow s_i, s.t., \boldsymbol{g}(i) = \max(\boldsymbol{g})$
 5:     if $k > 0$ and ($k = 1$ or $\boldsymbol{g}(i)/k > 0.5$) then
 6:         $\mathbf{C} \leftarrow \mathbf{C} \cup \{governor\}$
 7:         $\mathbf{E} \leftarrow \mathbf{E} \cup \{(root, governor)\}$
 8:         $\mathbf{G}' \leftarrow \mathbf{G} - \{\mathbf{G}_i\}$
 9:         $root \leftarrow governor$
10:     if $k \leq 1$ then return
11:     $\mathbf{G}_1$, $\mathbf{G}_2 \leftarrow$ clustering($\mathbf{G}'$,2)
12:     recursive($\mathbf{G}_1$,$\mathbf{C}$,$\mathbf{E}$,$root$)
13:     recursive($\mathbf{G}_2$,$\mathbf{C}$,$\mathbf{E}$,$root$)
14: function GENERATOR($D$)
15:     Obtain graph $\mathbf{G}$ for document $D$
16:     $\mathbf{C}_s \leftarrow \varnothing$, $\mathbf{E}_s \leftarrow \varnothing$
17:     recursive($\mathbf{G}$, $\mathbf{C}_s$, $\mathbf{E}_s$, $Null$)
18:     return $\mathbf{C}_s$, $\mathbf{E}_s$
```

dant patterns.

Therefore, in the experiments section, all our models are trained with 5k labeled articles.

B.3 Effects of Sampling Times

In the graph refinement module, we sample the upper nodes and improve their ROUGE similarity with the human-written highlights. Figure 8 shows the performance curves of different sampling times for each document. We found that more sampling times do not improve the performances of EMGN significantly, while requires much more time to train the model. Therefore, in the experiments section, we set the sampling times as one for both EMGN(root) and EMGN.

Algorithm 3 Evaluation for Mind-map.

```
Require: the edges of the manual mind-map $\mathbf{E}_1$, the edges of the generated mind-map $\mathbf{E}_2$
Ensure: the edge similarity score $f$, and the most similar pair $ms$
 1: function SIMFUNCTION($\mathbf{E}_1$, $\mathbf{E}_2$)
 2:     $r \leftarrow 0$
 3:     truncate($\mathbf{E}_2$),$s.t.|\mathbf{E}_2| = |\mathbf{E}_1|$
 4:     for ($s_i \rightarrow s_j$) in $\mathbf{E}_1$ do
 5:         $ms \leftarrow (None, None)$
 6:         for ($s_a \rightarrow s_b$) in $\mathbf{E}_2$ do
 7:             if $f(s_i \rightarrow s_j, ms[0] \rightarrow ms[1])$
 8:                 $< f(s_i \rightarrow s_j, s_a \rightarrow s_b)$ then
 9:                 $ms \leftarrow (s_a \rightarrow s_b)$
10:         $r \leftarrow f(s_i \rightarrow s_j, ms[0] \rightarrow ms[1]) + r$
11:         $\mathbf{E}_2 \leftarrow \mathbf{E}_2 - ms$
12:     return $r/|\mathbf{E}_1|$
```

C Mind-Map Generator

The algorithm for mind-map generator is enclosed in Algorithm 2. After calculating the graph $\mathbf{G}$ for a document, the nodes and edges are built recursively. It is worth noting that the graph refinement module also follows this recursive way.

We also enclose the algorithm for mind-map evaluation in Algorithm 3.