Structure Learning for Headline Generation

Ruqing Zhang, Jiafeng Guo, Yixing Fan, Yanyan Lan, and Xueqi Cheng
CAS Key Lab of Network Data Science and Technology, Institute of Computing Technology,
Chinese Academy of Sciences, Beijing, China
University of Chinese Academy of Sciences, Beijing, China
{zhangruqing,guojiafeng,fanyixing,lanyanyan,cxq}@ict.ac.cn

Abstract

Headline generation is an important problem in natural language processing, which aims to describe a document by a compact and informative headline. Some recent successes on this task have been achieved by advanced graph-based neural models, which marry the representational power of deep neural networks with the structural modeling ability of the relational sentence graphs. The advantages of graph-based neural models over traditional Seq2Seq models lie in that they can encode long-distance relationship between sentences beyond the surface linear structure. However, since documents are typically weakly-structured data, modern graph-based neural models usually rely on manually designed rules or some heuristics to construct the sentence graph a prior. This may largely limit the power and increase the cost of the graph-based methods. In this paper, therefore, we propose to incorporate structure learning into the graph-based neural models for headline generation. That is, we want to automatically learn the sentence graph using a data-driven way, so that we can unveil the document structure flexibly without prior heuristics or rules. To achieve this goal, we employ a deep & wide network to encode rich relational information between sentences for the sentence graph learning. For the deep component, we leverage neural matching models, either representation-focused or interaction-focused model, to learn semantic similarity between sentences. For the wide component, we encode a variety of discourse relations between sentences. A Graph Convolutional Network (GCN) is then applied over the sentence graph to generate high-level relational representations for headline generation. The whole model could be optimized end-to-end so that the structure and representation could be learned jointly. Empirical studies show that our model can significantly outperform the state-of-the-art headline generation models.

1 Introduction

Headline generation is a task to generate a fluent and condensed headline for a document, with the constraint that only a short sequence of words is allowed to generate. Previous efforts on headline generation can be categorized to extractive and abstractive methods. Extractive methods directly extract sentences from the original document and then exploit sentence compression techniques to produce the headline. They have the advantage of producing fluent sentence and preserving the meaning of original documents, but also inevitably suffer from generating less informative headlines. Recently, thanks to the popularity of neural network models and their ability to learn continuous representations without preprocessing tools, abstractive methods have become popular in headline generation, which aim to generate the headline based on understanding the document.

Without loss of generality, a good understanding of the document should consist of not only the language modeling (Rush, Chopra, and Weston 2015), but also the structure modeling, i.e., capturing the intrinsic relationships between sentences. The headline generation task is generally formulated as a sequence-to-sequence (Seq2Seq) learning problem and a wide range of neural Seq2Seq models have been applied to solve it. Specifically, Seq2Seq models only capture the surface linear structure of the document, which can not model the long-distance relationship between sentences. To address this issue, some recent studies (Yasunaga et al. 2017; Fernandes, Allamanis, and Brockschmidt 2019) begin to focus on the graph-based neural models, which are inspired by modeling highly-structured objects (e.g., entity relationships and molecules) using graphs (Kipf and Welling 2017). These methods exploit the representational power of deep neural networks and the structural modeling ability of the relational sentence graphs, which can encode long-distance relationship between sentences. However, since documents are typically weakly-structured data, these graph-based neural models usually rely on manually designed rules (e.g., discourse relations) or some heuristics (e.g., tf-idf cosine similarity) to construct the sentence graph a prior. This may result in the limitation of flexibility in modeling different relationships between sentences. Furthermore, it could increase the cost of designing the sentence graph.

In this work, we propose a Structure Learning based Generation model, named SLGen, which incorporates sentence graph learning into the graph-based neural models for headline generation. We represent document sentences as nodes in a graph with undirected edges as links between sentences. To learn the sentence graph, we employ a deep & wide network including deep component and wide com-

ponent, to encode rich relational information between sentences in a principled way. Specifically, for the deep component, we introduce two different neural matching models, i.e., representation-focused model or interaction-focused model, to learn the semantic similarity between sentences. Representation-focused model aims to conduct matching based on the sentence representations by a relatively simple matching function. Interaction-focused model first builds local interactions between two sentences, and then uses deep neural networks to learn hierarchical interaction patterns for matching. For the wide component, we encode various discourse relations between sentences. In this way, our model can learn the intrinsic sentence graph of the input document, thus incorporating well-established insights from earlier work.

Given the sentence graph, our SLGen model applies a Graph Convolutional Network (GCN) (Kipf and Welling 2017), which takes in sentence representations induced from the deep & wide framework as input node features. Through multiple layer-wise propagation, the GCN generates high-level relational representations for sentences that capture high order neighborhoods information. To have a global view of the entire document, we obtain the document representation by computing the weighted sums of sentence relational representations, where the weight of each sentence is measured by computing its degree centrality. Finally, we employ the unmodified decoder with the attention mechanism (Bahdanau, Cho, and Bengio 2014) to focus on sentence relational representations for better generation. To the best of our knowledge, this is the first study to automatically learn the underlying document structure and apply a GCN architecture over the sentence graph in a data-driven way for headline generation.

We evaluate our model on a public benchmark collection, i.e., the New York Times (NYT) Annotated corpus. For evaluation, we compare with several state-of-the-art methods to verify the effectiveness of our model. Empirical results demonstrate that our model can well learn the sentence graph and outperform all the baselines significantly.

2 Related Work

In this section, we briefly review two lines of related work, i.e., headline generation and graph neural networks.

2.1 Headline Generation

Existing methods on headline generation can be generally categorized into extractive methods and abstractive methods. Extractive methods try to extract the most important sentences in the document and rearranging them into a new headline. Early works mainly define surface features for unsupervised learning (Luhn 1958; Edmundson 1964). Later, graph-based methods are applied broadly to rank sentences (Erkan and Radev 2004; Mihalcea and Tarau 2004). Recently, neural Seq2Seq models have also been investigated for the extractive task (Nallapati, Zhai, and Zhou 2017).

Abstractive methods, on the other hand, aim to generate more novel headline based on understanding the document. Banko, Mittal, and Witbrock (2000) viewed the task as a problem analogous to statistical machine translation for content selection and surface realization. Woodsend, Feng, and Lapata (2010) proposed a quasi-synchronous grammar approach to produce clear headline. Later, the task is formulated as a Seq2Seq learning problem and neural models have been adopted to solve it. For example, Rush, Chopra, and Weston (2015) trained an attention-based encoder-decoder framework in a data-driven way. Chopra, Auli, and Rush (2016) extended this work with an attentive RNN framework, and incorporated the position information of words. See, Liu, and Manning (2017) used the pointer-generator network that can copy words from the document to solve the out-of-vocabulary words. Moreover, recent studies focus on using reinforcement learning to combine extractive and abstractive methods. (Hsu et al. 2018; Chen and Bansal 2018). However, these abstractive models still capture the surface linear structure of the input document.

2.2 Graph Neural Networks

Early studies about graph neural networks learned the representation of a node by propagating neighbor information via recurrent neural networks in an iterative manner until a fixed point is reached (Micheli 2009; Scarselli et al. 2008). However, this process is computationally expensive. Inspired by the huge success of convolutional networks in the computer vision area, many works related to Graph Convolutional Networks (GCN) have been rapidly developed (Kipf and Welling 2017; Hamilton, Ying, and Leskovec 2017; Niepert, Ahmed, and Kutzkov 2016) to improve efficiency, which directly perform the convolution in the graph.

The GCN has been introduced in several NLP tasks, e.g., text classification (Yao, Mao, and Luo 2019), machine translation (Bastings et al. 2017), reading comprehension (De Cao, Aziz, and Titov 2019) and semantic role labeling (Marcheggiani and Titov 2017), where GCN is used to encode the syntactic structure of sentences. Specifically, headline generation can be viewed as a text summarization problem (Tan, Wan, and Xiao 2017), with the constraint that only a short sequence of words is allowed to generate to preserve the essential topics of a document. Some recent studies also explored GCN for text summarization. Yasunaga et al. (2017) presented a novel multi-document summarization system that incorporates pre-designed sentence relation graphs. Fernandes, Allamanis, and Brockschmidt (2019) presented a framework for extending sequence encoders with a graph component that can leverage rich additional structure. Different from these previous efforts, our model automatically learns the sentence graph without prior rules for better generation.

3 Our Approach

In this section, we introduce the SLGen model, a novel structure learning based generation method designed for the headline generation task.

3.1 Model Overview

Formally, given an input document $D = \{s_1, \ldots, s_L\}$ with L sentences, where each sentence $s_i \in D$ contains a se-

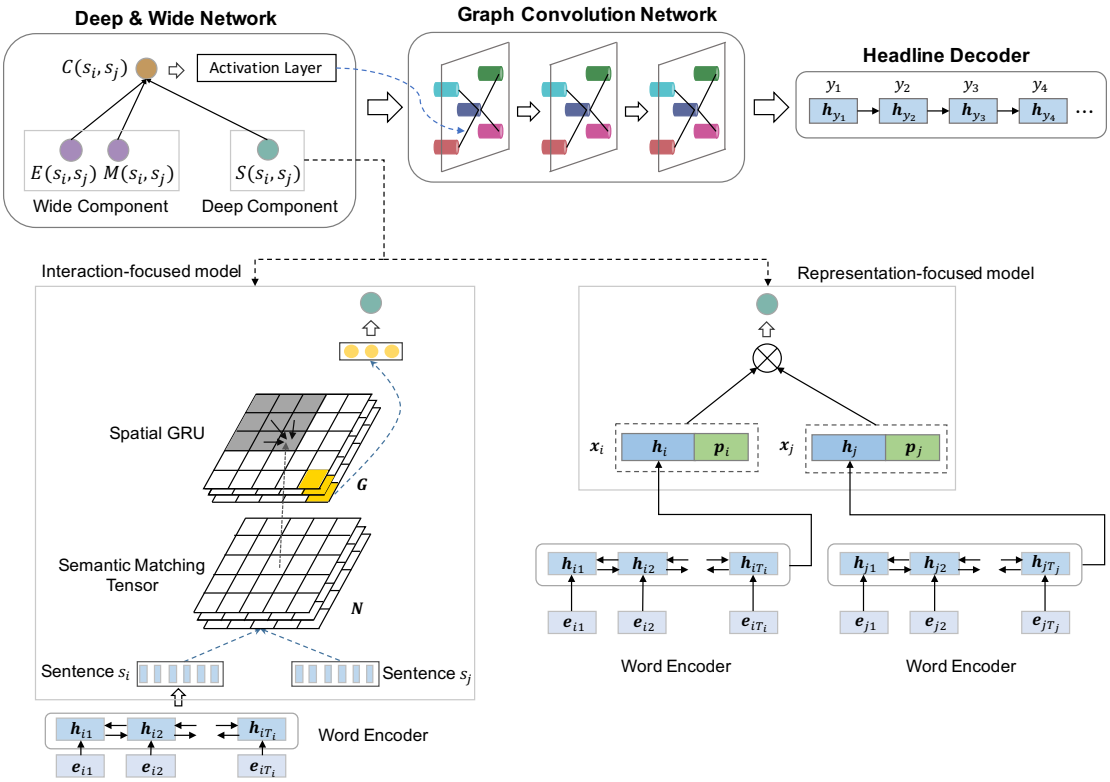


Figure 1: The overall architecture of SLGen model.

quence of T_i words w_{it} ($t \in [1, T_i]$), SLGen aims to generate a headline Y for the document D.

Basically, our SLGen model could be decomposed into three dependent components: 1) Deep & Wide Network: to automatically construct the sentence graph; 2) Graph Convolution Network: to generate high-level relational representations for sentences; 3) Headline Decoder: to generate the headline for the input document. The overall architecture of SLGen is depicted in Figure 1, and we will detail our model as follows.

3.2 Deep & Wide Network

This component is to automatically learn the sentence graph for unveiling the document structure. Formally, given the input document D, we aim to construct the sentence graph $\mathcal{G}$ where the nodes $\mathcal{V}$ are the set of L sentences and the undirected edges $\mathcal{E}$ denote the link between two nodes.

Here, we employ a deep & wide network (Cheng et al. 2016) including deep (semantic) component and wide (discourse) component, where the deep component is used to generalize semantic similarity through low-dimensional dense embeddings and the wide component is used to memorize the discourse relations. We may further incorporate some other rules or heuristics into the wide component, and we leave this as our future work.

Deep (Semantic) Component The deep component is a feed-forward neural network and aims to produce the semantic matching score $S(s_i, s_j)$ for each sentence pair $< s_i, s_j >$, which could be decomposed into two steps: 1) Word encoder: to achieve the sentence representations by encoding the words; 2) Semantic matching: to learn semantic similarity based on sentence representations.

- **Word encoder**. Given a sentence s_i, each word $w_{it} \in s_i$ is firstly represented by its distributed representation $\mathbf{e}_{it}$. We then use a bi-directional GRU as the word encoder. The forward GRU reads the words in the i-th sentence s_i in the left-to-right direction, resulting in a sequence of hidden states $(\overrightarrow{\mathbf{h}_{i1}}, \ldots, \overrightarrow{\mathbf{h}_{iT_i}})$. The backward GRU reads s_i in the reversed direction and outputs $(\overleftarrow{\mathbf{h}_{i1}}, \ldots, \overleftarrow{\mathbf{h}_{iT_i}})$. We obtain the hidden state for the word w_{it} by concatenating the forward and backward hidden states, i.e., $\mathbf{h}_{it} = [\overrightarrow{\mathbf{h}_{it}} || \overleftarrow{\mathbf{h}_{it}}]$. Then we concatenate the last hidden states of the forward and backward passes as the semantic representation of the sentence s_i, denoted as $\mathbf{h}_i = [\overrightarrow{\mathbf{h}_{iT_i}} || \overleftarrow{\mathbf{h}_{i1}}]$. Specifically, the surface linear structure in the input document is still rich in meaning (Fernandes, Allamanis, and Brockschmidt 2019). We assume that each sentence s_i, beyond its semantic representation $\mathbf{h}_i$, also has a position representation $\mathbf{p}_i$. For each sentence s_i, the final sentence representation $\mathbf{x}_i$ is constructed by concatenating the semantic and position representations, i.e., $\mathbf{x}_i = [\mathbf{h}_i || \mathbf{p}_i]$.
- **Semantic Matching**. As shown in Figure 1, based on sentence representations, we introduce two neural matching models to learn semantic similarity between sentences. One is the representation-focused model, which tries to

directly compare two sentence representations by a simple matching function. The other is the interaction-focused model, which learn from a matching matrix between two sentences (Guo et al. 2016).

– **Representation-focused model**. Given the sentence pair $< s_i, s_j >$, the representation-focused model measures the degree of matching as a score by a scoring function based on the sentence representations $\mathbf{x}_i$ and $\mathbf{x}_j$. Here, we employ Bilinear to compute the semantic matching score:

$$S(s_i, s_j) = \mathbf{x}_i^{\mathsf{T}} \mathbf{W} \mathbf{x}_j + \mathbf{b}, \quad (1)$$

where $\mathbf{W}$ is the transformation matrix to learn.

– **Interaction-focused model**. Formally, each sentence s_i is represented as a sequence of word hidden states denoted by $\{\mathbf{h}_{i1}, \ldots, \mathbf{h}_{iT_i}\}$. The interaction-focused model produces the word-level interaction matrix for final matching score. Here, we compute the semantic matching matrix $\mathbf{M}$ based on the word hidden states from the sentence pair $< s_i, s_j >$, defined as follows:

$$\mathbf{M}_{pq} = \frac{\mathbf{h}_{ip}^{\mathsf{T}} \mathbf{h}_{jq}}{||\mathbf{h}_{ip}|| \cdot ||\mathbf{h}_{jq}||}, \quad (2)$$

where $\mathbf{h}_{ip}$ and $\mathbf{h}_{jq}$ stand for the p and q-th word hidden states for two sentences s_i and s_j, respectively. To further incorporate word importance, we extend each element of $\mathbf{M}_{pq}$ to a three-dimensional vector $\mathbf{N}_{pq} = [\mathbf{w}_p, \mathbf{w}_q, \mathbf{M}_{pq}]$ by concatenating two corresponding compressed word hidden states as in (Fan et al. 2018), where $\mathbf{w}_p = \mathbf{h}_{ip} * \mathbf{W}_c$ and $\mathbf{w}_q = \mathbf{h}_{jq} * \mathbf{W}_c$, here, $\mathbf{W}_c$ is the learnable transformation parameter.

Based on the sentence interaction tensor, a spatial GRU (Wan et al. 2016) is applied to generate the semantic matching evidences, which scans the input tensor from top left to bottom right:

$$\overrightarrow{\mathbf{G}}_{pq} = g(\overrightarrow{\mathbf{G}}_{p-1,q}, \overrightarrow{\mathbf{G}}_{p,q-1}, \overrightarrow{\mathbf{G}}_{p-1,q-1}, \mathbf{N}_{p,q}), \quad (3)$$

where g denotes the spatial GRU unit. We take the last hidden representation $\overrightarrow{\mathbf{G}}_{T_i,T_j}$ as the matching vector. Finally, we use a MLP to output the semantic matching score:

$$S(s_i, s_j) = \mathbf{W}_s \overrightarrow{\mathbf{G}}_{T_i,T_j} + \mathbf{b}_s. \quad (4)$$

Wide (Discourse) Component The wide component is a feature transformation model and aims to encode discourse relations between sentences. Here, we follow (Liu and Lapata 2019) to compute two discourse-aware scores for each sentence pair $< s_i, s_j >$, i.e., entity score $E(s_i, s_j)$ and marker score $M(s_i, s_j)$, by utilizing co-occurrence entities and discourse markers respectively:

- **Co-occurrence entities**. For each sentence $s_i \in D$, we extract a set of entities in the sentence using the Spacy Named Entity Recognizer[1]. For each sentence pair $< s_i, s_j >$, we count the number of entities with exact match as the entity score $E(s_i, s_j)$.
- **Discourse markers**. We use explicit discourse markers (e.g., but, instead, meanwhile) to identify the relationship between two adjacent sentences. If two sentences $< s_i, s_j >$ are adjacent in the document and they are connected with one of the discourse markers, we set the marker score $M(s_i, s_j)$ as 1, and 0 otherwise.

[1]https://spacy.io/api/entityrecognizer

Combination of Deep & Wide Component The deep and wide component are combined to measure the relationship between sentences. The correlation score $C(s_i, s_j)$ is obtained by using the weighted sums of three scores:

$$C(s_i, s_j) = \lambda_E E(s_i, s_j) + \lambda_M M(s_i, s_j) + \lambda_S S(s_i, s_j), \quad (5)$$

where λ_E, λ_M and λ_S are different weights for entity, marker and semantic matching score respectively. $C(s_i, s_j)$ between all sentence pairs $< s_i, s_j >$ form the fully connected matrix, i.e., the correlation matrix $\mathbf{C} \in \mathbb{R}^{L \times L}$.

Then, an activation layer is applied as the final layer of the deep & wide framework over the correlation matrix $\mathbf{C}$ to pick out the sentence graph, i.e., to create or delete edges between the nodes, and encode the set of edges into the adjacency matrix $\mathbf{A}$. Here, we introduce two activation functions to produce the adjacency matrix $\mathbf{A}$ from $\mathbf{C}$.

- k**-Max Pooling** over a sequence of values returns the subsequence of k maximum values in the sequence. Specifically, for each row in $\mathbf{C}$, the top k values are directly returned while others are returned as 0. It means that for each sentence, we only create edges between top k correlative sentences with it.
- **Relu** returns 0 if it receives any negative input, but for any positive value it returns that value back. Specifically, for each row in $\mathbf{C}$, the positive values are directly returned while others are returned as 0. It means that we only delete edges which connect two nodes with negative correlation.

Due to the symmetry of the undirected graph, we average the edges weights in both directions to obtain the final adjacency matrix.

3.3 Graph Convolution Network

After constructing the sentence graph, we feed the graph into a simple multi-layer Graph Convolutional Network (GCN) as in (Kipf and Welling 2017). Formally, consider the sentence graph of the document $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, the GCN is to learn a function $f(\mathbf{X}, \mathbf{A})$ where the inputs are:

- $\mathbf{X} \in \mathcal{R}^{L \times M}$, the input node feature matrix, where M is the dimension of each sentence representation $\mathbf{x}_i$.
- $\mathbf{A} \in \mathcal{R}^{L \times L}$, the adjacency matrix of graph $\mathcal{G}$, where the diagonal elements of $\mathbf{A}$ are set to 1 because of self-loops in the graph.

For a one-layer GCN, the new K-dimensional node feature matrix $L^{(1)} \in \mathcal{R}^{L \times K}$ is computed as

$$L^{(1)} = \rho(\tilde{\mathbf{A}} \mathbf{X} \mathbf{W}_g^{(0)}), \quad (6)$$

where $\tilde{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}}$ is the normalized symmetric adjacency matrix and $\mathbf{D}$ is the node degree matrix with $\mathbf{D}_{ii} = \sum_j \mathbf{A}_{ij}$. $\mathbf{W}_g^{(0)} \in \mathcal{R}^{M \times K}$ is the weight parameter to learn. ρ is an activation function, e.g., a ReLU.

When multiple GCN layers are stacked, information about higher order neighborhoods are incorporated. The re-

cursive computation is as follows:

$$L^{(h+1)} = \rho(\tilde{\mathbf{A}} L^{(h)} \mathbf{W}_g^{(h)}), \tag{7}$$

where h denotes the layer number and $L^{(0)} = \mathbf{X}$.

As an example, if we have a two-layer GCN, we produce the high-level sentence relational representation $\mathbf{s}_i^g$ that incorporate the sentence relationships:

$$\mathbf{S} = f(\mathbf{X}, \mathbf{A}) = \tilde{\mathbf{A}} \rho(\tilde{\mathbf{A}} \mathbf{X} \mathbf{W}_g^{(0)}) \mathbf{W}_g^{(1)}, \tag{8}$$

where $\mathbf{S}$ is the relational representation matrix and $\mathbf{s}_i^g \in \mathbf{S}$.

Afterwards, we obtain the document representation $\mathbf{c}$ by computing the weighted sums of sentence relational representations, which is used as the initial hidden state of the summary decoder. Specifically, the weight of each sentence can be measured by simply computing its degree centrality $d(s_i)$, which is defined as:

$$d(s_i) = \sum_{j \in \{1,\ldots,i-1,i+1,\ldots,L\}} \mathbf{A}_{ij}, \tag{9}$$

where $\mathbf{A}_{ij}$ denotes the edge weight between the sentence pair $< s_i, s_j >$.

After obtaining the centrality score for each sentence, we can obtain the document representation $\mathbf{c}$ as follows:

$$\mathbf{c} = \sum_{i=1}^{L} \alpha_i \mathbf{s}_i^g, \tag{10}$$

where the weight α_i is defined as $\alpha_i = \text{softmax}(d(s_i))$.

3.4 Headline Decoder

The goal of the headline decoder is to generate a headline Y given the document representation of the input document. Similar with traditional Seq2Seq models, we employ the attention mechanism (Bahdanau, Cho, and Bengio 2014) to allow the decoder to pay different attention to different sentences of the document when generating different words. When generating a word at step t, the decoder takes all the sentence relational representations to form the context vector $\mathbf{c}_t$ for better generation. Thus, the hidden state at step time t of the decoder $\mathbf{h}_{y_t}$ can be obtained by

$$\begin{aligned} h_{y_t} &= f(\mathbf{h}_{y_{t-1}}, \mathbf{c}_t, y_{t-1}), \\ \mathbf{c}_t &= \textstyle\sum_{i=1}^{L} \beta_{ti} \mathbf{s}_i^g, \\ \mathbf{h}_{y_0} &= \mathbf{c}, \end{aligned} \tag{11}$$

where f is a GRU unit and y_{t-1} is the predicted target symbol at step $t-1$. β_{ti} indicates how much the i-th sentence s_i from the input document contributes to generating the t-th word, and is computed as $\beta_{ti} = \text{softmax}(\mathbf{s}_i^g \cdot \mathbf{h}_{y_{t-1}})$.

3.5 Model Learning

We employ maximum likelihood estimation (MLE) to learn our SLGen model over the training corpus $\mathcal{D}$. The loss function is defined as:

$$\mathcal{L}(\theta) = \sum_{(D,Y) \in \mathcal{D}} -\log P(Y|D;\theta). \tag{12}$$

We use the Adam (Kingma and Ba 2015) gradient-based optimization method to learn the model parameters θ.

4 Experiments

In this section, we conduct experiments to verify the effectiveness of our proposed model.

Table 1: Data statistics: #s denotes the number of sentences and #w denotes the number of words.

Pairs	1,855,657
Document: Vocabulary #w	6,649,762
Document: avg #s	51.6
Document: avg #w	556.9
Document sentence: avg #w	10.8
Headline: Vocabulary #w	421,199
Headline: avg #w	6.6

4.1 Dataset Description

We conduct our experiments on the public New York Times (NYT) Annotated corpus[2]. The corpus contains over 1.8 million documents written and published by the New York Times between January 1, 1987 and June 19, 2007. Table 1 shows the statistics of the dataset. We leave out the pairs whose headlines have less than 3 words or more than 15 words, and whose documents have less than 20 words or more that 2000 words, reducing the corpus to 1.58 million articles. We randomly sample 2000 pairs to form the development and test set respectively, and the left pairs are used as the training data.

4.2 Implementation Details

We implement our model in Tensorflow[3]. The dimension of word embeddings is 300, while the dimension of position embeddings is 200. We use one layer of bi-directional GRU for word encoder and another uni-directional GRU for decoder. We use three GCN hidden layers. The hidden unit size in the word encoder, word decoder and GCN is 300. The pooling parameter k is set as 12. The learning rate of Adam (Kingma and Ba 2015) is set as 0.0005. All trainable parameters are initialized in the range $[-0.1, 0.1]$. We construct two separate vocabularies for documents and headlines by using 80,000 and 10,000 most frequent words on each side in the training data. All the remaining words are replaced by the special <UNK> symbol.

For training, we use a mini-batch size of 64 and documents with similar length (in terms of the number of sentences) are organized to be a batch. Dropout with probability 0.2 is applied between vertical GRU stacks and gradient clipping is adopted by scaling gradients when the norm exceeded a threshold of 5. We run our model on a Tesla K80 GPU card, and we run the training for up to 15 epochs, which takes approximately four day. We select the model that achieves the lowest perplexity on the development set. All hyper-parameters of our model are also tuned using the development set. We report results on the test set.

By combining two semantic matching models (i.e., representation-focused and interaction-focused model) and two activation functions (i.e., k-max pooling and Relu) used in the SLGen, we obtain four types of SLGen models denoted as **SLGen$_{\text{Rep+Max}}$**, **SLGen$_{\text{Rep+Relu}}$**, **SLGen$_{\text{Int+Max}}$** and **SLGen$_{\text{Int+Relu}}$**.

[2] https://catalog.ldc.upenn.edu/LDC2008T19

[3] https:/www.tensorflow.org/

4.3 Baselines

Model Variants Here, we first employ some degraded $\text{SLGen}_{\text{Int+Relu}}$ models to investigate the effect of different components.

- **SLGen$_{\text{-D}}$** removes the deep component in the deep & wide network.
- **SLGen$_{\text{-W}}$** removes the wide component in the deep & wide network.
- **SLGen$_{\text{-DW+tfidf}}$** removes the deep & wide network and constructs the tf-idf cosine similarity graph (Yasunaga et al. 2017). Namely, we add an edge between two sentences if the tf-idf cosine similarity between them is above 0.2.
- **SLGen$_{\text{-DW+ADG}}$** replaces the tfidf similarity graph in $\text{SLGen}_{\text{-DW+tfidf}}$ as the Approximate Discourse Graph (ADG) (Christensen et al. 2013).
- **SLGen$_{\text{-DW+tfidf+ADG}}$** directly sums the adjacency matrixes of the tfidf similarity graph and the ADG to form the mixed graph.

For all variants, we use the word encoder to obtain the sentence representations as the input node features of GCN.

Extractive Models We apply extractive models to extract the sentence from the input document as the headline.

- **PREFIX** simply uses the first sentence as the headline.
- **TextRank** (Mihalcea and Tarau 2004) is a graph-based method inspired by the PageRank algorithm.
- **LexRank** (Erkan and Radev 2004) is also a graph-based method inspired by the PageRank algorithm. The difference with TextRank is to use different methods to calculate the similarity between two sentences.
- **SumBasic** (Nenkova and Vanderwende 2005) assigns a score to each sentence which reflects how many high-frequency words it contains.

Abstractive Models We also apply neural generation models to generate the headline.

- **Seq2Seq$_{\text{hie}}$** employs a hierarchical encoder structure (words sequentially form sentence, sentences sequentially form document) (Li, Luong, and Jurafsky 2015).
- **Seq2Seq$_{\text{hie+att}}$** employs the sentence-level attention mechanism (Bahdanau, Cho, and Bengio 2014) in the decoding phase over the $\text{Seq2Seq}_{\text{hie}}$.
- **ABS** combines a neural probabilistic language model with a generation algorithm which produces the accurate summary (Rush, Chopra, and Weston 2015).
- **BILSTM+GNN** extends sequence encoders with a graph component that can leverage rich prior structure (Fernandes, Allamanis, and Brockschmidt 2019).

4.4 Evaluation Methodologies

We adopt the automatic, i.e., Rouge (Lin 2004), to measure the quality of headlines generated by our model and the baselines. ROUGE is commonly employed to evaluate n-grams recall of generated sentences with gold-standard sentences as references. Rouge-1, Rouge-2 and Rouge-L recall scores measure the uni-gram, bi-gram and longest-common substring similarities, respectively.

Table 2: Model analysis of four types of our SLGen models under the automatic evaluation.

Model	Rouge-1	Rouge-2	Rouge-L
$\text{SLGen}_{\text{Rep+max}}$	34.16	17.37	33.08
$\text{SLGen}_{\text{Rep+Relu}}$	34.42	17.58	33.15
$\text{SLGen}_{\text{Int+Max}}$	35.46	18.24	33.89
$\text{SLGen}_{\text{Int+Relu}}$	**35.82**	**18.41**	**34.12**

Table 3: Ablation analysis of our SLGen model with its variants under the automatic evaluation.

Model	Rouge-1	Rouge-2	Rouge-L
$\text{SLGen}_{\text{-D}}$	34.27	17.33	33.00
$\text{SLGen}_{\text{-W}}$	35.77	18.38	34.01
$\text{SLGen}_{\text{-DW+tfidf}}$	34.30	17.41	33.06
$\text{SLGen}_{\text{-DW+ADG}}$	34.31	17.42	33.08
$\text{SLGen}_{\text{-DW+tfidf+ADG}}$	34.35	17.42	33.10
$\text{SLGen}_{\text{Int+Relu}}$	**35.82**	**18.41**	**34.12**

4.5 Evaluation Results

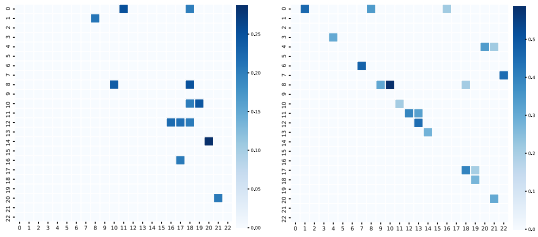
Model Analysis We first analyze the four types of *SLGen* models to investigate which combination is better for headline generation. As shown in Table 2, we can find that: (1) Our *SLGen* model based on interaction-focused model perform better than that based on representation-focused model. For example, $\textit{SLGen}_{\text{Int+Relu}}$ boosts Rouge-1 over $\textit{SLGen}_{\text{Rep+Relu}}$ by 1.4. This is mainly because interaction-focused model is capable to capture more complex semantic interaction between sentences. (2) Moreover, $\textit{SLGen}_{\text{Int+Relu}}$ can achieve better results than $\textit{SLGen}_{\text{Int+Max}}$, indicating that flexibly learning the edges between sentences is better than fixing the number of edges. (3) $\textit{SLGen}_{\text{Int+Relu}}$ achieves the best performance as evaluated by the Rouge.

Then, we conduct ablation analysis to investigate the effect of the deep & wide framework in our *SLGen* model. As shown in Table 3, we find that: (1) By removing the deep component and the wide component from $\textit{SLGen}_{\text{Int+Relu}}$ respectively, $\textit{SLGen}_{\text{-D}}$ has a more significant drop than $\textit{SLGen}_{\text{-W}}$ as compared with $\textit{SLGen}_{\text{Int+Relu}}$. The results indicate that the deep semantic matching model has much bigger impact than additional discourse relations for capturing relationships between sentences. (2) $\textit{SLGen}_{\text{-DW+tfidf}}$ and $\textit{SLGen}_{\text{-DW+ADG}}$ do not have an obvious influence on the results compared with $\textit{SLGen}_{\text{-D}}$. Moreover, $\textit{SLGen}_{\text{-DW+tfidf+ADG}}$ mixes the tf-idf similarity graph and ADG, but achieves similar results, indicating that prior rules or heuristics are not suitable for headline generation which limit the flexibility in unveiling the document structure.

Baseline Comparison The performance comparisons between our model and the baselines are shown in Table 4. We can observe that: (1) *PREFIX* performs poorly, indicating that lead sentences are not always sufficient for headline generation. (2) The extractive models performs pretty well. Specifically, *TextRank* and *LexRank* perform better than other two extractive models (i.e., *LSA* and *SumBasic*),

Table 4: Comparisons between our SLGen and the baselines under the automatic evaluation. Two-tailed t-tests demonstrate the improvements of our SLGen to all the baseline models are statistically significant (p-value < 0.01).

Model	Rouge-1	Rouge-2	Rouge-L
PREFIX	11.85	5.29	10.50
TextRank	30.67	5.42	26.18
LexRank	26.80	6.29	22.87
LSA	26.07	4.23	22.53
SumBasic	17.48	4.01	15.71
ABS	28.29	15.49	27.35
Seq2Seq$_{hie}$	32.42	16.04	31.21
Seq2Seq$_{hie+att}$	33.98	17.15	32.74
BILSTM+GNN	34.11	17.52	32.89
SLGen$_{\text{Int+Relu}}$	**35.82**	**18.41**	**34.12**



(a) tf-idf Cosine Similarity Graph (b) Sentence Graph learned by SLGen$_{\text{Int+Relu}}$

Figure 2: (a) is the heatmap of the tf-idf similarity graph; (b) is the heatmap of learned sentence graph in our SLGen$_{\text{Int+Relu}}$ Model.

by defining sentence salience by graph-based centrality scoring. (3) The abstractive models can achieve better results than the extractive methods, since these generative methods apply deep architectures to understand the document semantics. (4) The ABS improves the results a little in terms of Rouge-1, showing that the additional language modeling information is not sufficient for headline generation. (5) The results of *Seq2Seq*$_{\text{hie+att}}$ model show that introducing a hierarchical structure in the encoder and using sentence-level attention in the decoder is effective and can improve the performance significantly. (6) By extending sequence encoders with a pre-designed graph component, *BILSTM+GNN* is able to achieve the best performance among all the baseline methods. (7) The better results of *SLGen*$_{\text{Int+Relu}}$ over abstractive models demonstrate the effectiveness of automatically learning the sentence graph, which can uncover the intrinsic structure for better document understanding.

4.6 Case Study

To better understand what can be learned by our model, we conduct some case studies. We take one document from the test data as an example. As shown in Table 5, there are 23 sentences distributed over 14 paragraphs in this document, and due to the limited space, we only show some key sentences. We show the generated headline from our *SLGen*$_{\text{Int+Relu}}$ as well as that from the baseline *SLGen*$_{\text{-DW+tfidf}}$. Meanwhile, we also depict the sentence graph learned by our model and the heuristic tf-idf cosine similarity graph from *SLGen*$_{\text{-DW+tfidf}}$ in Figure 2 to help analysis. Specifically, we show the upper triangular matrix of the adjacency matrix ($\mathbf{A} \in \mathcal{R}^{23\times23}$) without self-loop. As we can see, the tf-idf cosine similarity graph lets each sentence connect with little other sentences, which can not sufficiently capture the complex semantic relationships. Also, the most informative sentences achieved by the degree centrality as defined in Equation 9 are S19 and S9, which in turn guide the decoder to pay attention to inappropriate source sentences and generate inconsistent headline with the ground-truth. The sentence graph learned by our *SLGen*$_{\text{Int+Relu}}$ model is relatively dense and the document structure can be learned flexibly. Our model finds that the most informative sentences are S9 and S1 which contribute to better understanding of the document, and then the decoder generates a much better headline which is more consistent with the ground-truth.

Table 5: An example from the test NYT data. **G** is the ground truth. **S** is the output of SLGen$_{\text{-DW+tfidf}}$. **D** is the output of our SLGen$_{\text{Int+Relu}}$. **S1** to **S23** are the sentences in the document.

S1: Italy's fragile government, under fire for making a deal to free an Italian journalist . . . standards for confronting the rising number of high-profile kidnappings in war zones. **S2** . . . **S8** . . . **S9**: Here in Italy, the government has faced criticism at home and abroad for pressuring the Afghan government to release five Taliban prisoners . . . **S10** . . . **S11** . . . **S12**: On Thursday, Massimo D'Alema, foreign minister for Italy's center-left government, strongly defended the deal to free Mr. <UNK>. **S13** . . . **S18** . . . **S19**: Center-right opposition leaders have accused the government of putting undue pressure on the Afghan government to swap the prisoners. . . **S20** . . . **S23**. . .
G: Italy proposes rules for handling abductions. **S**: Italian hostage is free with help from Afghan. **D**: Italy proposes criterions to solve kidnappings after dealing with Afghan.

4.7 Conclusion and Future Work

In this paper, we proposed to incorporate structure learning into the graph-based neural models for headline generation, which aims to automatically learn the sentence graph in a data-driven way to unveil the document structure flexibly. We employed a deep & wide network to learn the sentence graph and then applied a Graph Convolutional Network (GCN) architecture over the sentence graph for better generation. Thus, the structure and representation could be learned in an end-to-end way. Empirical results showed that our model can significantly outperform the state-of-the-art methods. In the future work, we would like to extend our model to the document summarization task by learning the intrinsic structure of multiple summary sentences.

5 Acknowledgments

This work was supported by Beijing Academy of Artificial Intelligence (BAAI), and funded by the National Natural Science Foundation of China (NSFC) under Grants No. 61425016, 61722211, 61773362, 61872338, and 61902381, the Youth Innovation Promotion Association CAS under Grants No. 20144310, and 2016102, the National Key RD Program of China under Grants No. 2016QY02D0405, and the Foundation and Frontier Research Key Program of Chongqing Science and Technology Commission (No. cstc2017jcyjBX0059).

References

[Bahdanau, Cho, and Bengio 2014] Bahdanau, D.; Cho, K.; and Bengio, Y. 2014. Neural machine translation by jointly learning to align and translate. In *ICLR*.

[Banko, Mittal, and Witbrock 2000] Banko, M.; Mittal, V. O.; and Witbrock, M. J. 2000. Headline generation based on statistical translation. In *ACL*.

[Bastings et al. 2017] Bastings, J.; Titov, I.; Aziz, W.; Marcheggiani, D.; and Sima'an, K. 2017. Graph convolutional encoders for syntax-aware neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

[Chen and Bansal 2018] Chen, Y.-C., and Bansal, M. 2018. Fast abstractive summarization with reinforce-selected sentence rewriting. *arXiv preprint arXiv:1805.11080*.

[Cheng et al. 2016] Cheng, H.-T.; Koc, L.; Harmsen, J.; Shaked, T.; Chandra, T.; Aradhye, H.; Anderson, G.; Corrado, G.; Chai, W.; Ispir, M.; et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*, 7–10. ACM.

[Chopra, Auli, and Rush 2016] Chopra, S.; Auli, M.; and Rush, A. M. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *NAACL*.

[Christensen et al. 2013] Christensen, J.; Soderland, S.; Etzioni, O.; et al. 2013. Towards coherent multi-document summarization. In *NAACL*.

[De Cao, Aziz, and Titov 2019] De Cao, N.; Aziz, W.; and Titov, I. 2019. Question answering by reasoning across documents with graph convolutional networks. In *NAACL-HLT*.

[Edmundson 1964] Edmundson, H. 1964. Problems in automatic abstracting. *Communications of the ACM*.

[Erkan and Radev 2004] Erkan, G., and Radev, D. R. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of artificial intelligence research*.

[Fan et al. 2018] Fan, Y.; Guo, J.; Lan, Y.; Xu, J.; Zhai, C.; and Cheng, X. 2018. Modeling diverse relevance patterns in ad-hoc retrieval. In *SIGIR*, 375–384. ACM.

[Fernandes, Allamanis, and Brockschmidt 2019] Fernandes, P.; Allamanis, M.; and Brockschmidt, M. 2019. Structured neural summarization. In *ICLR*.

[Guo et al. 2016] Guo, J.; Fan, Y.; Ai, Q.; and Croft, W. B. 2016. A deep relevance matching model for ad-hoc retrieval. In *CIKM*, 55–64. ACM.

[Hamilton, Ying, and Leskovec 2017] Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. In *NIPS*.

[Hsu et al. 2018] Hsu, W.-T.; Lin, C.-K.; Lee, M.-Y.; Min, K.; Tang, J.; and Sun, M. 2018. A unified model for extractive and abstractive summarization using inconsistency loss. In *ACL*.

[Kingma and Ba 2015] Kingma, D. P., and Ba, J. 2015. Adam: A method for stochastic optimization. In *ICLR*.

[Kipf and Welling 2017] Kipf, T. N., and Welling, M. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.

[Li, Luong, and Jurafsky 2015] Li, J.; Luong, M.-T.; and Jurafsky, D. 2015. A hierarchical neural autoencoder for paragraphs and documents. In *ACL*.

[Lin 2004] Lin, C.-Y. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, 74–81.

[Liu and Lapata 2019] Liu, Y., and Lapata, M. 2019. Hierarchical transformers for multi-document summarization. In *ACL*.

[Luhn 1958] Luhn, H. P. 1958. The automatic creation of literature abstracts. *IBM Journal of research and development*.

[Marcheggiani and Titov 2017] Marcheggiani, D., and Titov, I. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *EMNLP*.

[Micheli 2009] Micheli, A. 2009. Neural network for graphs: A contextual constructive approach. *IEEE Transactions on Neural Networks*.

[Mihalcea and Tarau 2004] Mihalcea, R., and Tarau, P. 2004. Textrank: Bringing order into text. In *EMNLP*, 404–411.

[Nallapati, Zhai, and Zhou 2017] Nallapati, R.; Zhai, F.; and Zhou, B. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *AAAI*.

[Nenkova and Vanderwende 2005] Nenkova, A., and Vanderwende, L. 2005. The impact of frequency on summarization.

[Niepert, Ahmed, and Kutzkov 2016] Niepert, M.; Ahmed, M.; and Kutzkov, K. 2016. Learning convolutional neural networks for graphs. In *ICML*, 2014–2023.

[Rush, Chopra, and Weston 2015] Rush, A. M.; Chopra, S.; and Weston, J. 2015. A neural attention model for abstractive sentence summarization. In *ACL*.

[Scarselli et al. 2008] Scarselli, F.; Gori, M.; Tsoi, A. C.; Hagenbuchner, M.; and Monfardini, G. 2008. The graph neural network model. *IEEE Transactions on Neural Networks*.

[See, Liu, and Manning 2017] See, A.; Liu, P. J.; and Manning, C. D. 2017. Get to the point: Summarization with pointer-generator networks. In *ACL*.

[Tan, Wan, and Xiao 2017] Tan, J.; Wan, X.; and Xiao, J. 2017. From neural sentence summarization to headline generation: A coarse-to-fine approach. In *IJCAI*.

[Wan et al. 2016] Wan, S.; Lan, Y.; Xu, J.; Guo, J.; Pang, L.; and Cheng, X. 2016. Match-srnn: Modeling the recursive matching structure with spatial rnn. In *IJCAI*.

[Woodsend, Feng, and Lapata 2010] Woodsend, K.; Feng, Y.; and Lapata, M. 2010. Generation with quasi-synchronous grammar. In *EMNLP*.

[Yao, Mao, and Luo 2019] Yao, L.; Mao, C.; and Luo, Y. 2019. Graph convolutional networks for text classification. In *AAAI*.

[Yasunaga et al. 2017] Yasunaga, M.; Zhang, R.; Meelu, K.; Pareek, A.; Srinivasan, K.; and Radev, D. 2017. Graph-based neural multi-document summarization. In *CoNLL*.