Data Science and Engineering (2019) 4:14–23
https://doi.org/10.1007/s41019-019-0087-7

Multi-Task Learning for Abstractive and Extractive Summarization

Yangbin Chen[1] · **Yun Ma**[1] · **Xudong Mao**[2] · **Qing Li**[2]

Received: 20 February 2019 / Revised: 16 March 2019 / Accepted: 19 March 2019 / Published online: 16 April 2019

Abstract
The abstractive method and extractive method are two main approaches for automatic document summarization. In this paper, to fully integrate the relatedness and advantages of both approaches, we propose a general unified framework for abstractive summarization which incorporates extractive summarization as an auxiliary task. In particular, our framework is composed of a shared hierarchical document encoder, a hierarchical attention mechanism-based decoder, and an extractor. We adopt multi-task learning method to train these two tasks jointly, which enables the shared encoder to better capture the semantics of the document. Moreover, as our main task is abstractive summarization, we constrain the attention learned in the abstractive task with the labels of the extractive task to strengthen the consistency between the two tasks. Experiments on the CNN/DailyMail dataset demonstrate that both the auxiliary task and the attention constraint contribute to improve the performance significantly, and our model is comparable to the state-of-the-art abstractive models. In addition, we cut half number of labels of the extractive task, pretrain the extractor, and jointly train the two tasks using the estimated sentence salience of the extractive task to constrain the attention of the abstractive task. The results do not decrease much compared with using full-labeled data of the auxiliary task.

Keywords Automatic document summarization · Multi-task learning · Attention mechanism

1 Introduction

Automatic document summarization has been studied for decades. The target of it is to generate a shorter passage from the document in a grammatically and logically coherent way, meanwhile preserving the important information. There are two main approaches for document summarization: extractive summarization and abstractive summarization. The extractive method first extracts salient sentences from the source document and then groups them to produce a summary, without changing the source text. Graph-based ranking model [15, 30] and feature-based classification model [4, 45] are two typical models. However, summaries generated from extractive method unavoidably include secondary or redundant information, which are far from those generated by human beings [44].

The abstractive method, in contrast, produces generalized summaries, conveying information in a concise way, and eliminating the limitations to the original words and sentences of the document. This task is more challenging since it needs advanced language compression and generation techniques. Syntax [8, 17] and semantics [16, 25] are usually used for generating abstractive summaries.

Recently, Recurrent Neural Network (RNN) and its derivatives based sequence-to-sequence model with attention mechanism has been applied to abstractive summarization, due to its great success in machine translation [1, 29, 41]. However, there are some task-specific challenges. First, the RNN-based models have difficulties in capturing long-term dependencies, making summarization for long document much tougher. Second, different from machine translation which aims to cover as more information as possible from the source to the target, an abstractive summary corresponds

✉ Yangbin Chen
robinchen2-c@my.cityu.edu.hk

Yun Ma
mayun371@gmail.com

Xudong Mao
xudong.xdmao@gmail.com

Qing Li
csqli@comp.polyu.edu.hk

[1] City University of Hong Kong, Kowloon Tong, Hong Kong

[2] The Hong Kong Polytechnic University, Hung Hom, Hong Kong

to only a small part of the source document, adding to the difficulties of the learning process.

For the first challenge, we adopt a hierarchical approach to address the long-term dependency problem. Hierarchical approaches have been used in several text classification tasks [23, 43], whereas few of them have been applied to the abstractive summarization task. In particular, we encode the input document in a hierarchical way from word-level to sentence-level. There are two advantages of using a hierarchical approach. First, it is able to capture both the local and global semantics of a document, resulting in better feature learning. Second, it can improve the training efficiency as the computational complexity of the RNN-based model can be reduced by dividing the long document into short sentences.

Multi-task learning method has been successfully applied in a wide range of tasks across computer vision [18], speech recognition [12] and natural language processing [11]. It improves generalization by leveraging the domain-specific information contained in the training signals of related tasks [5]. Incorporating auxiliary tasks which are related to the main task is one of most used mechanisms of multi-task learning. We choose extractive summarization as the auxiliary task for abstractive summarization because of their high correlation. We use the hard parameter sharing approach through sharing the hierarchical encoder to better capture the semantics of the document.

The attention mechanism is widely used in sequence-to-sequence tasks [10, 29]. However, for abstractive summarization, it is difficult to learn the attention since only a small part of the source document contributes to the summary, which is the second challenge mentioned above. In this paper, we propose two methods to learn a better attention distribution. First, we use a hierarchical attention mechanism, which means that we calculate both the word-level and sentence-level attention. Similar to the hierarchical approach for the encoder, the advantage of using hierarchical attention is to capture both the locally and globally important information related to the target summary. Second, we use the salience scores of the auxiliary task (i.e., the extractive summarization) to constrain the sentence-level attention because of their same meaning and indication. The extractive summarization is good at selecting important sentences so that we use them to guide the learning process of the sentence-level attention.

In this paper, we present a novel multi-task learning-based technique for abstractive summarization which incorporates extractive summarization as an auxiliary task. In general, our framework consists of three parts: a shared document encoder, a hierarchical attention mechanism-based decoder and an extractor. As Fig. 1 shows, we encode the document in a hierarchical way [Fig. 1(1), (2)] in order to address the long-term dependency problem. Then, the learned document representations are shared by the extractor [Fig. 1(3)] and the decoder [Fig. 1(5)]. The extractor and the decoder are jointly trained which can capture better semantics of the document. Furthermore, as both the sentence salience scores in the extractor and the sentence-level attention in the decoder indicate the importance of source sentences, we constrain the learned sentence-level attention [Fig. 1(4)] with the sentence salience scores of the extractor in order to strengthen their consistency.

We conduct experiments on a news corpus which is the CNN/DailyMail dataset [6]. First is the ablation experiments. From the results, we find that adding the auxiliary

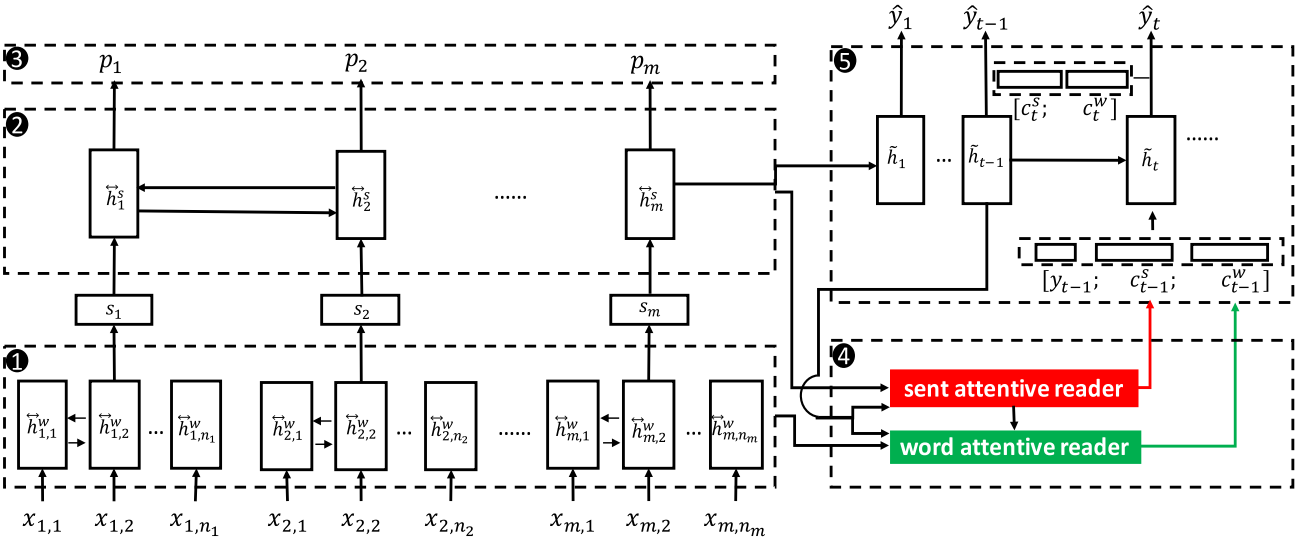

Fig. 1 General framework of our proposed model with 5 components: (1) Word-level encoder encodes the sentences independently word-by-word, (2) Sentence-level encoder encodes the document sentence-by-sentence, (3) Sentence extractor does binary classification for each sentence, (4) Hierarchical attention assists to calculate the word-level and sentence-level context vectors for the decoding steps, (5) Decoder decodes the output sequential word sequence with a beam search algorithm

extractive task and constraining the attention are both useful to improve the performance of the abstractive task. Second is the comparison with baseline models. From the results, we draw a conclusion that our proposed joint model is comparable to the state-of-the-art abstractive models. Third is a few-labeled experiment. From the results, we find that even if we label only half number of samples of the extractive task, the performance of abstractive task does not decrease much.

The rest of the paper is organized as follows. In Sect. 2, we present our neural summarization model. In Sect. 3, we describe the experimental setup. The results and discussions are shown in Sect. 4. The related work is introduced in Sect. 5. Section 6 is the conclusion.

2 Neural Summarization Model

In this section, we describe the detailed components of the framework of our proposed model. As illustrated in Fig. 1, the hierarchical document encoder includes both the word-level and sentence-level encoders. The word-level encoder reads sentences independently word-by-word and gets an embedding for each sentence. The sentence-level encoder reads the document sentence-by-sentence and generates document representations. The representations are shared by two tasks. For the extractive summarization task, they are fed into the sentence extractor which is a sequence labeling model to calculate salience scores. For the abstractive summarization task, they are used as the input of the GRU-based sequence-to-sequence model to generate abstractive summaries. The sequence-to-sequence model takes advantage of the hierarchical attention which includes the sentence-level and word-level attention. Finally, the two tasks are jointly trained as a multi-task learning problem, with the sentence-level attention constrained by the salience scores.

2.1 Shared Hierarchical Document Encoder

We encode the document in a hierarchical way. In particular, the word sequences of the sentences are first encoded by a bidirectional GRU network parallelly. And a sequence of sentence-level vector representations called sentence embeddings are generated by a nonlinear transformation. Then, the sentence embeddings are fed into another bidirectional GRU network and get the document representations.

Formally, let $\mathbf{V}$ denote the vocabulary which contains D tokens, and each token is embedded as a d-dimension vector. Given an input document $\mathbf{X}$ containing m sentences $\{\mathbf{X}_i, i \in 1, \ldots, m\}$, let n_i denote the number of words in $\mathbf{X}_i$.

Word-level encoder reads a sentence word-by-word until the end, using a bidirectional GRU network as depicted by the following equations:

$$\vec{\boldsymbol{h}}^{\mathrm{w}}_{i,j} = \begin{bmatrix} \vec{\boldsymbol{h}}^{\mathrm{w}}_{i,j} \\ \overleftarrow{\boldsymbol{h}}^{\mathrm{w}}_{i,j} \end{bmatrix} \tag{1}$$

$$\vec{\boldsymbol{h}}^{\mathrm{w}}_{i,j} = \mathrm{GRU}\left(\boldsymbol{x}_{i,j}, \vec{\boldsymbol{h}}^{\mathrm{w}}_{i,j-1}\right) \tag{2}$$

$$\overleftarrow{\boldsymbol{h}}^{\mathrm{w}}_{i,j} = \mathrm{GRU}\left(\boldsymbol{x}_{i,j}, \overleftarrow{\boldsymbol{h}}^{\mathrm{w}}_{i,j+1}\right) \tag{3}$$

where $\boldsymbol{x}_{i,j}$ represents the embedding vector of the jth word in th ith sentence. The superscript w means word-level. $\vec{\boldsymbol{h}}^{\mathrm{w}}_{i,j} \in \mathbb{R}^H$ is the forward hidden state vector corresponded to the word $\boldsymbol{x}_{i,j}$ and $\overleftarrow{\boldsymbol{h}}^{\mathrm{w}}_{i,j} \in \mathbb{R}^H$ is the backward hidden state vector corresponded to the word $\boldsymbol{x}_{i,j}$. $\overleftrightarrow{\boldsymbol{h}}^{\mathrm{w}}_{i,j} \in \mathbb{R}^{2H}$ is a concatenated vector of the two column vectors. H is the size of the hidden state.

Furthermore, the ith sentence is represented by a nonlinear transformation of its word-level hidden states as follows:

$$\boldsymbol{s}_i = \tanh\left(\mathbf{W}^{\mathrm{T}} \frac{1}{n_i} \sum_{j=1}^{n_i} \overleftrightarrow{\boldsymbol{h}}^{\mathrm{w}}_{i,j} + \boldsymbol{b}\right) \tag{4}$$

where $\boldsymbol{s}_i \in \mathbb{R}^H$ is the sentence embedding and $\mathbf{W}$, $\boldsymbol{b}$ are learnable parameters.

Sentence-level encoder reads a document sentence-by-sentence until the end, using another bidirectional GRU network as depicted by the following equations:

$$\vec{\boldsymbol{h}}^{\mathrm{s}}_{i} = \begin{bmatrix} \vec{\boldsymbol{h}}^{\mathrm{s}}_{i} \\ \overleftarrow{\boldsymbol{h}}^{\mathrm{s}}_{i} \end{bmatrix} \tag{5}$$

$$\vec{\boldsymbol{h}}^{\mathrm{s}}_{i} = \mathrm{GRU}\left(\boldsymbol{s}_i, \vec{\boldsymbol{h}}^{\mathrm{s}}_{i-1}\right) \tag{6}$$

$$\overleftarrow{\boldsymbol{h}}^{\mathrm{s}}_{i} = \mathrm{GRU}\left(\boldsymbol{s}_i, \overleftarrow{\boldsymbol{h}}^{\mathrm{s}}_{i+1}\right) \tag{7}$$

where $\overleftrightarrow{\boldsymbol{h}}^{\mathrm{s}}_{i} \in \mathbb{R}^{2H}$ is a concatenated vector of the forward hidden state $\vec{\boldsymbol{h}}^{\mathrm{s}}_{i} \in \mathbb{R}^H$ and the backward hidden state $\overleftarrow{\boldsymbol{h}}^{\mathrm{s}}_{i} \in \mathbb{R}^H$ like the word-level encoder does. Here, the superscript s means sentence-level. The concatenated vectors $\overleftrightarrow{\boldsymbol{h}}^{\mathrm{s}}_{i}$ are document representations shared by the two tasks which will be introduced next.

Such a hierarchical architecture has two advantages. First, it can reduce the negative effects during the training process caused by the long-term dependency problem so that the document can be represented from both local and global granularity. Second, it helps improve the training efficiency. Considering a document with m sentences and each sentence with n words, a basic encoder's computational complexity is $O(mn\mathrm{d}H)$, while a hierarchical encoder is $O((m+n)\mathrm{d}H)$.

The efficiency improvement is especially significant when the document is large.

2.2 Sentence Extractor

The sentence extractor can be viewed as a sequential binary classifier. We use a logistic function to calculate a score between 1 and 0, which is an indicator of whether or not to keep the sentence in the final extractive summary. The score can also be considered as the salience of a sentence in the document. Let p_i denote the score and $q_i \in \{0, 1\}$ denote the result of whether or not to keep the sentence. In particular, p_i is calculated as follows:

$$\begin{aligned} p_i &= P\left(q_i = 1|\vec{\boldsymbol{h}}_i^{\mathrm{s}}\right) \\ &= \sigma\left(\mathbf{W}^{\mathrm{extr}^{\mathrm{T}}}\vec{\boldsymbol{h}}_i^{\mathrm{s}} + b^{\mathrm{extr}}\right) \end{aligned} \tag{8}$$

where $\mathbf{W}^{\mathrm{extr}}$ is the weight and b^{extr} is the bias which can be learned.

The sentence extractor generates a sequence of probabilities indicating the importance of the sentences. As a result, the extractive summary is created by selecting sentences with a probability larger than a given threshold τ. We set $\tau = 0.5$ in our experiment. We use the cross entropy loss as the extractive loss, i.e.,

$$E_{\mathrm{se}} = -\frac{1}{m}\sum_{i=1}^{m} q_i \log p_i + \left(1 - q_i\right) \log\left(1 - p_i\right) \tag{9}$$

2.3 Decoder

Our decoder is a unidirectional GRU network with hierarchical attention. We use the attention to calculate the context vectors which are weighted sums of the hidden states of the hierarchical encoders. The equations are given as below:

$$\boldsymbol{c}_t^{\mathrm{s}} = \sum_{i=1}^{m} \boldsymbol{\alpha}_{t,i} \cdot \vec{\boldsymbol{h}}_i^{\mathrm{s}} \tag{10}$$

$$\boldsymbol{c}_t^{\mathrm{w}} = \sum_{i=1}^{m}\sum_{j=1}^{n_i} \boldsymbol{\beta}_{t,i,j} \cdot \vec{\boldsymbol{h}}_{i,j}^{\mathrm{w}} \tag{11}$$

where $\mathbf{c}_t^{\mathrm{s}}$ is the sentence-level context vector and $\mathbf{c}_t^{\mathrm{w}}$ is the word-level context vector at decoding time step t. Here, the superscript s means sentence-level and the superscript w means word-level. Specifically, $\boldsymbol{\alpha}_{t,i}$ denotes the attention value at tth decoding step over the ith sentence and $\boldsymbol{\beta}_{t,i,j}$ denotes the attention value at tth decoding step over the jth word of the ith sentence.

The input of the GRU-based language model at decoding time step t contains three vectors: the word embedding of previous generated word $\hat{\boldsymbol{y}}_{t-1}$, the sentence-level context vector of previous time step $\boldsymbol{c}_{t-1}^{\mathrm{s}}$ and the word-level context vector of previous time step $\boldsymbol{c}_{t-1}^{\mathrm{w}}$. They are transformed by a linear function and fed into the language model as follows:

$$\tilde{\boldsymbol{h}}_t = \mathrm{GRU}\left(\tilde{\boldsymbol{h}}_{t-1}, f_{\mathrm{in}}\left(\hat{\boldsymbol{y}}_{t-1}, \boldsymbol{c}_{t-1}^{\mathrm{s}}, \boldsymbol{c}_{t-1}^{\mathrm{w}}\right)\right) \tag{12}$$

$$f_{\mathrm{in}}\left(\hat{\boldsymbol{y}}_{t-1}, \boldsymbol{c}_{t-1}^{\mathrm{s}}, \boldsymbol{c}_{t-1}^{\mathrm{w}}\right) = \mathbf{W}^{\mathrm{abs}^{\mathrm{T}}}\begin{bmatrix}\hat{\boldsymbol{y}}_{t-1} \\ \boldsymbol{c}_{t-1}^{\mathrm{s}} \\ \boldsymbol{c}_{t-1}^{\mathrm{w}}\end{bmatrix} + \mathbf{b}^{\mathrm{abs}} \tag{13}$$

where $\tilde{\boldsymbol{h}}_t$ is the hidden state of decoding time step t. f_{in} is a linear transformation function with $\mathbf{W}^{\mathrm{abs}}$ as the weight and $\boldsymbol{b}^{\mathrm{abs}}$ as the bias.

The hidden states of the language model are used to generate the output word sequence. The conditional probability distribution over the vocabulary in the tth time step is:

$$P\left(\hat{\boldsymbol{y}}_t|\hat{\boldsymbol{y}}_1, \ldots, \hat{\boldsymbol{y}}_{t-1}, \boldsymbol{x}\right) = g\left(f_{\mathrm{out}}\left(\tilde{\boldsymbol{h}}_t, \boldsymbol{c}_t^{\mathrm{s}}, \boldsymbol{c}_t^{\mathrm{w}}\right)\right) \tag{14}$$

$$f_{\mathrm{out}}\left(\tilde{\boldsymbol{h}}_t, \boldsymbol{c}_t^{\mathrm{s}}, \boldsymbol{c}_t^{\mathrm{w}}\right) = \mathbf{W}^{\mathrm{soft}^{\mathrm{T}}}\begin{bmatrix}\tilde{\boldsymbol{h}}_t \\ \boldsymbol{c}_t^{\mathrm{s}} \\ \boldsymbol{c}_t^{\mathrm{w}}\end{bmatrix} + \mathbf{b}^{\mathrm{soft}} \tag{15}$$

where g is the softmax function and f_{out} is a linear transformation function with $\mathbf{W}^{\mathrm{soft}}$ and $\boldsymbol{b}^{\mathrm{soft}}$ as learnable parameters.

The negative log likelihood loss is applied as the loss of the decoder, i.e.,

$$E_y = \frac{1}{T}\sum_{t=1}^{T} -\log\left(y_t\right) \tag{16}$$

where T is the length of the target summary.

2.4 Hierarchical Attention

The hierarchical attention mechanism consists of a word-level attention reader and a sentence-level attention reader so as to take full advantage of the multi-level semantics captured by the hierarchical document encoder. The sentence-level attention indicates the salience distribution over the source sentences at the tth decoding step. It is calculated as follows:

$$\boldsymbol{\alpha}_{t,i} = \frac{\boldsymbol{e}_{t,i}^{\mathrm{s}}}{\sum_{k=1}^{m} \boldsymbol{e}_{t,k}^{\mathrm{s}}} \tag{17}$$

$$\boldsymbol{e}_{t,i}^{\mathrm{s}} = \exp\left\{\mathbf{V}^{\mathrm{s}^{\mathrm{T}}}\tanh\left(\mathbf{W}_1^{\mathrm{dec}^{\mathrm{T}}}\tilde{\boldsymbol{h}}_t + \mathbf{W}_1^{\mathrm{s}^{\mathrm{T}}}\vec{\boldsymbol{h}}_i^{\mathrm{s}} + \boldsymbol{b}_1^{\mathrm{s}}\right)\right\} \tag{18}$$

where $\mathbf{V}^{\mathrm{s}}$, $\mathbf{W}_1^{\mathrm{dec}}$, $\mathbf{W}_1^{\mathrm{s}}$ and $\boldsymbol{b}_1^{\mathrm{s}}$ are learnable parameters.

The word-level attention indicates the salience distribution over the source words at the tth decoding step. As the hierarchical encoder reads the input sentences independently, our model has two distinctions. First, the word-level attention is calculated within a sentence. Second, we multiply the word-level attention by the sentence-level attention of the sentence which the word belongs to. Comparing two words with the same word-level attention value, we consider the word more important if it is in a more important sentence. In this way, we can strengthen the influence of the important words in the important sentences. The word-level attention calculation is shown below:

$$\boldsymbol{\beta}_{t,i,j} = \boldsymbol{\alpha}_{t,i} \frac{\boldsymbol{e}^{\mathrm{w}}_{t,i,j}}{\sum_{l=1}^{n_i} \boldsymbol{e}^{\mathrm{w}}_{t,i,l}} \tag{19}$$

$$\boldsymbol{e}^{\mathrm{w}}_{t,i,j} = \exp\left\{ \mathbf{V}^{\mathrm{w}\mathrm{T}} \tanh\left(\mathbf{W}_2^{\mathrm{dec}\mathrm{T}} \tilde{\boldsymbol{h}}_t + \mathbf{W}_2^{\mathrm{w}\mathrm{T}} \vec{\boldsymbol{h}}^{\mathrm{w}}_{i,j} + \boldsymbol{b}_2^{\mathrm{w}} \right) \right\} \tag{20}$$

where $\mathbf{V}^{\mathrm{w}}$, $\mathbf{W}_2^{\mathrm{dec}}$, $\mathbf{W}_2^{\mathrm{w}}$ and $\boldsymbol{b}_2^{\mathrm{w}}$ are learnable parameters for the word-level attention calculation.

The abstractive summary of a long document can be viewed as a further compression of the most salient sentences of the document so that a well-learned sentence extractor and a well-learned attention distribution should both be able to indicate the important sentences of the source document. Motivated by this, we design a constraint to the sentence-level attention which is an $l2$ loss as follows:

$$E_{\mathrm{a}} = \frac{1}{m} \sum_{i=1}^{m} \left(p_i - \frac{1}{T} \sum_{t=1}^{T} \boldsymbol{\alpha}_{t,i} \right)^2 \tag{21}$$

When we use the full-labeled data, p_i is calculated by the logistic function which is trained simultaneously with the decoder. It is not suitable to constrain the attention with the p_i. In this case, we use the labels of the extractive task to constrain the attention.

2.5 Multi-Task Learning

We combine three types of loss functions mentioned above to train our proposed model in a multi-task learning way—the negative log likelihood loss E_y for the decoder, the cross entropy loss E_{se} for the extractor, and the $l2$ loss E_{a} as the attention. Hence,

$$E = E_y + \lambda \cdot E_{\mathrm{se}} + \gamma \cdot E_{\mathrm{a}} \tag{22}$$

where λ and γ are hyperparameters to balance the three loss functions.

The parameters are trained to minimize the joint loss function. In the inference stage, we use the beam search algorithm to select the word which approximately maximizes the conditional probability [2, 19, 39]. Moreover, we use a mandatory constraint which forbids the repetition a trigram in one sequence during the decoding process.

3 Experimental Setup

3.1 Dataset

We adopt the news dataset which is collected from the websites of CNN and DailyMail. It is originally prepared for the task of machine reading by Hermann et al. [21]. Cheng and Lapata [6] added labels to the sentences for the task of extractive summarization. Table 1 lists the statistics of the dataset.

3.2 Implementation Details

In our implementation, we set the vocabulary size D to be 50K and word embedding size d as 300. The word embeddings have not been pretrained as the training corpus is large enough to train them from scratch. We cut off the documents as a maximum of 35 sentences and truncate the sentences with a maximum of 50 words. We also truncate the targeted summaries with a maximum of 100 words. The word-level encoder and the sentence-level encoder correspond to a layer of bidirectional GRU, respectively, and the decoder is a layer of unidirectional GRU. All the three networks have the hidden size H as 200. For the loss function, λ is set as 100 and γ is set as 0.5. During the training process, we use Adagrad optimizer [14] with the learning rate of 0.15 and initial accumulator value of 0.1. The mini-batch size is 16. We implement the model in Tensorflow and train it using a GTX-1080Ti GPU. The beam search size for decoding is 5.

Table 1 Statistics of the CNN/DailyMail dataset

Dataset	DailyMail	CNN
#Train	193,986	83,568
#Valid	12,147	1220
#Test	10,350	1093
S.S.N.	25.60	30.17
S.S.L.	28.84	24.02
T.S.L.	59.30	40.10

S.S.N. indicates the average number of sentences of the source documents. *S.S.L.* indicates the average length of the sentences of the source documents. *T.S.L.* indicates the average length of the sentences of the target summaries

4 Results and Discussions

4.1 Evaluation of Proposed Components

To verify the effectiveness of our proposed model, we conduct ablation study by removing the corresponding parts, i.e., the auxiliary extractive task, the attention constraint and combination of them in order to make a comparison among their effects. We use Recall-Oriented Understudy for Gist Evaluation (ROUGE) scores [24] to evaluate the summarization models. ROUGE is essentially of a set of metrics for evaluating machine translation, text summarization and so on. In our work, it compares our generated summary against a set of reference summaries produced by human. We choose the full-length Rouge-F1 score on the three test sets for evaluation. The results are shown in Tables 2, 3 and 4.

The results demonstrate that adding the auxiliary task and the attention constraint improve the performance of the abstractive task. The performance declines most when both the extractive task and the attention constraint are removed. Furthermore, as shown in the tables, the performance declines more when the extractive task is removed, which means that the auxiliary task plays a more important role in our framework. Moreover, we found that the CNN dataset receives more performance improvement, by comparing Tables 2 and 3.

4.2 Comparison with Baselines

We expect to compare our test results with as many baselines as possible. However, not all baseline models provide results on all the three test sets. Moreover, few of them publish their codes, which makes it difficult to replicate their approaches. As a result, we compare our model with the Graph-based model and the seq2seq+attn (the same as Uni-GRU) on all the three corpora. As for the words-lvt2k-hieratt and Distraction-M3 models, we compare the results on a single corpora as their original papers present.

We compare the limited length recall variants of Rouge at 75 bytes on the DailyMail test set. We use the fundamental sequence-to-sequence attentional model, the SummaRuNNer-abs model [31] and the graph-based model [40] as baselines. The results are shown in Table 5. From the table, we can see that our model outperforms all the three baselines significantly.

Next we compare our results of full-length Rouge-F1 score on the CNN test set with the three baseline models in Table 6. The Uni-GRU is a GRU-based sequence-to-sequence attentional model without any hierarchical structures. The Distraction-M3 is referred to [7]. From Table 6, we can see that our model outperforms all the baselines significantly in Rouge-L, which means that our model does well in generating long salient sentences on the CNN set. However, our model is not as good as the graph-based model in Rouge-1 and Rouge-2 for this dataset.

Finally, in Table 7, we compare the full-length Rouge-F1 score on the entire CNN/DailyMail test set. The words-lvt2k-hieratt [10] is used as a baseline. From Table 7, we can see that our model performs not better than the graph-based model in Rouge-1 and Rouge-L, and the SummaRuNNer-abs performs the best in Rouge-2.

Table 2 Performance comparison of removing the components of our proposed model on the DailyMail test set using full-length F1 variants of Rouge

Method	Rouge-1	Rouge-2	Rouge-L
Our method	**36.7**	**14.1**	**34.0**
W/o extr	36.2	13.8	33.5
W/o attn	36.7	14.0	34.0
W/o extr+attn	36.2	13.7	33.4

Best results are in bold

Table 3 Performance comparison of removing the components of our proposed model on the CNN test set using full-length F1 variants of Rouge

Method	Rouge-1	Rouge-2	Rouge-L
Our method	**28.0**	**8.5**	**25.4**
W/o extr	26.8	7.8	24.3
W/o attn	26.9	8.2	24.6
W/o extr+attn	26.2	7.8	23.6

Best results are in bold

Table 4 Performance comparison of removing the components of our proposed model on the entire CNN/DailyMail test set using full-length F1 variants of Rouge

Method	Rouge-1	Rouge-2	Rouge-L
Our method	**35.8**	**13.6**	**33.4**
W/o extr	34.3	12.6	31.6
W/o attn	34.7	12.8	32.2
W/o extr+attn	34.2	12.5	31.6

Best results are in bold

Table 5 Performance comparison of various abstractive models on the DailyMail test set using limited length recall variants of Rouge at 75 bytes

Method	Rouge-1	Rouge-2	Rouge-L
seq2seq+attn	28.3	11.5	17.2
SummaRuNNer-abs	23.8	9.6	13.3
Graph-based	27.4	11.3	15.1
Our method	**28.9**	**12.0**	**17.7**

Best results are in bold

Table 6 Performance comparison of various abstractive models on the CNN test set using full-length F1 variants of Rouge

Method	Rouge-1	Rouge-2	Rouge-L
Uni-GRU	18.4	4.8	14.3
Distraction-M3	27.1	8.2	18.7
Graph-based	**30.3**	**9.8**	20.0
Our method	28.0	8.5	**25.4**

Best results are in bold

Table 7 Performance comparison of various abstractive models on the entire CNN/DailyMail test set using full-length F1 variants of Rouge

Method	Rouge-1	Rouge-2	Rouge-L
seq2seq+attn	33.6	12.3	31.0
words-lvt2k-hieratt	35.4	13.3	32.6
SummaRuNNer-abs	37.5	**14.5**	33.4
Graph-based	**38.1**	13.9	**34.0**
Our method	35.8	13.6	33.4

Best results are in bold

We also try an experiment which reduces half number of labels of the extractive task. We first only pretrain the extractive model and use the model to get the estimated salience scores of the remaining half training data. Then, we jointly train the abstractive and extractive task, with the attention constraint. We find that the rough results are not bad. In the future, we would like to try a semi-supervised way to train a good model using fewer labels.

4.3 Discussions

The above results show that there is no single model which performs the best on all the three test sets using Rouge variants as evaluation metrics. The full-length Rouge-L F1 scores demonstrate that our model performs well in generating the long salient subsequences. Moreover, the performance of limited length recall at 75 bytes shows that our model can also generate prescribed short summaries well.

However, the full-length Rouge-1 and Rouge-2 F1 score of our model is not as good as some baselines. There are three reasons for this phenomenon. The first reason is that both the auxiliary extractive task and the attention constraint encourage the model to cover more salient sentences, resulting in longer summaries generated by our model. Take the summaries in Fig. 2 as an example, our generated summaries are longer than the baselines, which increases the recall but may decrease the precision.

Another reason is that our model is more effective to summarize documents with more sentences, as more sentences lead to more complex semantics and long-term relationships which make it difficult for the normal abstractive method to learn good attention. For the datasets, the CNN dataset contains an average of 30.17 sentences, which is more than the DailyMail's 25.6 sentences. It leads to an improvement of the full-length Rouge variants on the CNN set more than on the DailyMail set. However, the DailyMail test set contains about 10 K pieces of news, while the CNN test set contains only about 1 K, so the model does not achieve fancy results on the entire test set in full-length Rouge variants.

The third reason is that when decoding the output sequence, the graph-based model compares the candidate sequences with the source words and sentences, so as to choose the most similar one. Our model does not include this mechanism.

Our model has the advantages from three aspects. First, summaries generated by our model contain as much important information and perform well grammatically. In practice, it depends on users' preference between the information coverage and condensibility to make a suitable balance. Compared to the low recall abstractive methods, our model is able to cover more information. And compared to the extractive methods, the generated summaries are more logically coherent. Second, the computational complexity of our approach is much less than the baselines due to the hierarchical structures, which improves the training speed. Third, as our key contribution is to improve the performance of the main task by incorporating an auxiliary task, in this experiment we just use normal GRU-based encoder and decoder for simplicity. However, as our model is orthogonal to various novel extractive and abstractive models, interested researchers can integrate their models to our proposed framework.

4.4 Case Study

We list some examples of the generated summaries of a source document(news) in Fig. 2. The source document contains 24 sentences with totally 660 words. Figure 2 presents five summaries: a golden summary which is the news highlight written by the reporter, three generated summaries by our proposed model, and the sequence-to-sequence attentional model.

From the figure, we can see that all system-generated summaries are copied words from the source document, because the highlights written by reporters used for training are usually partly copied from the source. However, different models have different characteristics. As illustrated in Fig. 2, all the four summaries are able to catch several key sentences from the document. The fundamental seq2seq+attn model misses some words like pronouns, which leads to grammatical mistakes in the generated summary. Our model without the auxiliary extractive task is able to detect more salient content, but the concatenated sentences have some grammatical mistakes and redundant words. Our model without the attention constraint generates fluent sentences which are very similar to the source sentences, but it focuses on just a small part of the

Source Document
a flaw in robots designed to perform surgery has been found that lets them be easily hacked, according to researchers. the experts were able to take control of a so-called telerobot during surgery by exploiting a simple programming trick. this enabled them to change the speed of the arms of the robot and change their orientation, making it impossible for the machines to carry out a procedure as directed. researchers at the @entity19 studied the @entity20 (shown) they found that robots designed for surgery could be 'easily' hacked in to. this is because they are operated over public networks which allowed the researchers to access them and stop them working the research was carried out by scientists ……
Golden Summary
researchers at the @entity19 studied so-called telerobots. they found robots designed for surgery could be hacked and manipulated. this is because robots being tested were operated over public networks. it allowed the researchers to access them and stop them working.
Our Method
researchers at the @entity19 studied the @entity20. they found that robots designed for surgery could be 'easily' hacked in to. it enabled them to change the speed of the arms of the robot and change their orientation, making it impossible for the machines to carry out a procedure as directed. this is because they are operated over public networks which allowed the researchers to access them and stop them working.
Our Method w/o extr
researchers at the @entity19 studied the @entity20 (shown) at @entity19. they found that robots designed for surgery could be 'easily' used to help them develop the technology. this is because they are operated over public networks which allowed the researchers to access them and stop them working the research was carried out.
Our Method w/o attn
researchers at the @entity19 studied the @entity20. they found that robots designed for surgery could be 'easily hacked in to'.
Seq2seq+attn
researchers at the @entity19 studied the @entity20 (shown) robots designed for surgery could be 'easily' hacked in to. this is because they are operated over public networks which allowed the researchers to access them and stop them working.

Fig. 2 An example of summaries toward a piece of news. From top to down, the first is the Source Document which is the raw news content. The second is the Golden Summary which is used as the reference summaries. The third is the summary generated by our proposed model. The fourth is the summary generated by our model without the auxiliary task. The fifth is the summary generated by our model without the attention constraint. The last is the summary generated by the vanilla sequence-to-sequence attentional model

source document. The summary generated by our proposed full model is most similar to the golden summary: It covers as much information and keeps correct grammar. It does not just copy sentences but use segmentation. Moreover, it changes the order of the source sentences while keeping the logical coherence.

5 Related Work

The neural attentional abstractive summarization model was first applied in sentence compression [36], where the input sequence is encoded by a convolutional network and the output sequence is decoded by a standard feedforward Neural Network Language Model (NNLM). Chopra et al. [10] and Lopyrev [27] switched to RNN-type model as the encoder, and did experiments on various values of hyper parameters. To address the out-of-vocabulary problem, Gu et al. [20], Cao et al. [3] and See et al. [37] presented the copy mechanism which adds a selection operation between the hidden state and the output layer at each decoding time step so as to decide whether to generate a new word from the vocabulary or copy the word directly from the source sentence.

The sequence-to-sequence model with attention mechanism [10] achieves competitive performance for sentence

compression, but is still a challenge for document summarization. Some researchers use hierarchical encoder to address the long-term dependency problem, yet most of the works are for extractive summarization tasks. Nallapati et al. [31] fed the input word embedding extended with new features to the word-level bidirectional GRU network and generated sequential labels from the sentence-level representations. Cheng and Lapata [6] presented a sentence extraction and word extraction model, encoding the sentences independently using Convolutional Neural Networks and decoding a binary sequence for sentence extraction as well as a word sequence for word extraction. Nallapati et al. [32] proposed a hierarchical attention with a hierarchical encoder, in which the word-level attention represents a probability distribution over the entire document.

There are recent summarization works focusing on reinforcement learning [34] and generative adversarial learning [26]. The performances are quite good. However, there remains difficulties about how to guarantee the training stability and how to improve the quality of generated texts.

Most previous works consider the extractive summarization and abstractive summarization as two independent tasks. The extractive task has the advantage of preserving the original information, and the abstractive task has the advantage of generating coherent sentences. It is thus reasonable and feasible to combine these two tasks. Tan et al. [40] as the first attempt to combine the two, tries to use the extracted sentence scores to calculate the attention for the abstractive decoder. But their proposed model using unsupervised graph-based model to rank the sentences is of high computation cost, and incurs long time to train.

Multi-task learning becomes more and more popular in NLP tasks. Some recent studies try to group related NLP tasks and do joint training, which aims to make the tasks benefit to each other. In machine translation, Zoph and Knight [46] jointly train the encoders. Dong et al. [13] jointly train the decoders. Johnson et al. [22] jointly train both encoders and decodes. What is more interesting is that parsing and image captioning [28], POS tagging and NER [33], and dependency tree structure [42] can also perform as auxiliary tasks to machine translation. Other NLP tasks such as semantic parsing [35], question answering [9] and chunking [38] have also been shown to benefit from multi-task learning. This is the reason driving us to use multi-task learning for document summarization.

6 Conclusion

In this work, we have presented a sequence-to-sequence model with hierarchical document encoder and hierarchical attention for abstractive summarization, and incorporated extractive summarization as an auxiliary task. To our best knowledge, it is the first attempt in summarization to use multi-task learning method to jointly train the two tasks by sharing the same document encoder. The auxiliary task and the attention constraint contribute to improve the performance of the main task. Experiments on the CNN/DailyMail datasets show that our proposed framework is comparable to the state-of-the-art abstractive models. This paper presents an initial unified framework for summarization. In the future, we will try to adopt more advanced encoder and decoder to make further improvement. We will also explore the multi-task learning as a data augmentation approach to use the high-resource tasks to benefit the low-resource tasks in a semi-supervised way.

Acknowledgements This article was financially supported by Innovation and Technology Commission - Hong Kong (Grant No. GHP/036/17SZ) and City University of Hong Kong (Grant No. 9220089).

References

1. Bahdanau D, Cho K, Bengio Y (2014) Neural machine translation by jointly learning to align and translate. arXiv:1409.0473
2. Boulanger-Lewandowski N, Bengio Y, Vincent P (2013) Audio chord recognition with recurrent neural networks. In: ISMIR, pp 335–340
3. Cao Z, Luo C, Li W, Li S (2017) Joint copying and restricted generation for paraphrase. In: AAAI, pp 3152–3158
4. Cao Z, Wei F, Li S, Li W, Zhou M, Wang H (2015) Learning summary prior representation for extractive summarization. In: ACL, vol 2, pp 829–833
5. Caruana R (1997) Multitask learning. Mach Learn 28(1):41–75
6. Cheng J, Lapata M (2016) Neural summarization by extracting sentences and words. arXiv:1603.07252
7. Chen Q, Zhu X, Ling Z, Wei S, Jiang H (2016) Distraction-based neural networks for document summarization. arXiv:1610.08462
8. Cheung JCK, Penn G (2014) Unsupervised sentence enhancement for automatic summarization. In: EMNLP, pp 775–786
9. Choi E, Hewlett D, Uszkoreit J, Polosukhin I, Lacoste A, Berant J (2017) Coarse-to-fine question answering for long documents. In: Proceedings of the 55th annual meeting of the association for computational linguistics (volume 1: long papers), vol 1, pp 209–220
10. Chopra S, Auli M, Rush AM (2016) Abstractive sentence summarization with attentive recurrent neural networks. In: Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies, pp 93–98
11. Collobert R, Weston J (2008) A unified architecture for natural language processing: deep neural networks with multitask

learning. In: Proceedings of the 25th international conference on Machine learning, ACM, pp 160–167
12. Deng L, Hinton G, Kingsbury B (2013) New types of deep neural network learning for speech recognition and related applications: an overview. In: 2013 IEEE international conference on acoustics, speech and signal processing, IEEE, pp 8599–8603
13. Dong D, Wu H, He W, Yu D, Wang H (2015) Multi-task learning for multiple language translation. In: Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing (volume 1: long papers), vol 1, pp 1723–1732
14. Duchi J, Hazan E, Singer Y (2011) Adaptive subgradient methods for online learning and stochastic optimization. J Mach Learn Res 12:2121–2159
15. Erkan G, Radev DR (2004) Lexrank: graph-based lexical centrality as salience in text summarization. J Artif Intell Res 22:457–479
16. Fang Y, Zhu H, Muszynska E, Kuhnle A, Teufel S (2016) A proposition-based abstractive summarizer
17. Gerani S, Mehdad Y, Carenini G, Ng RT, Nejat B (2014) Abstractive summarization of product reviews using discourse structure. In: EMNLP, vol 14, pp 1602–1613
18. Girshick R (2015) Fast R-CNN. In: Proceedings of the IEEE international conference on computer vision, pp 1440–1448
19. Graves A (2012) Sequence transduction with recurrent neural networks. arXiv:1211.3711
20. Gu J, Lu Z, Li H, Li VO (2016) Incorporating copying mechanism in sequence-to-sequence learning. arXiv:1603.06393
21. Hermann KM, Kocisky T, Grefenstette E, Espeholt L, Kay W, Suleyman M, Blunsom P (2015) Teaching machines to read and comprehend. In: Advances in neural information processing systems, pp 1693–1701
22. Johnson M, Schuster M, Le QV, Krikun M, Wu Y, Chen Z, Thorat N, Viégas F, Wattenberg M, Corrado G et al (2017) Google's multilingual neural machine translation system: enabling zero-shot translation. Trans Assoc Comput Linguist 5:339–351
23. Li J, Luong MT, Jurafsky D (2015) A hierarchical neural autoencoder for paragraphs and documents. arXiv:1506.01057
24. Lin CY (2004) Rouge: a package for automatic evaluation of summaries. In: Text summarization branches out: proceedings of the ACL-04 workshop, Barcelona, Spain, vol 8
25. Liu F, Flanigan J, Thomson S, Sadeh N, Smith NA (2015) Toward abstractive summarization using semantic representations
26. Liu L, Lu Y, Yang M, Qu Q, Zhu J, Li H (2018) Generative adversarial network for abstractive text summarization. In: Thirty-second AAAI conference on artificial intelligence
27. Lopyrev K (2015) Generating news headlines with recurrent neural networks. arXiv:1512.01712
28. Luong MT, Le QV, Sutskever I, Vinyals O, Kaiser L (2015) Multi-task sequence to sequence learning. arXiv:1511.06114
29. Luong MT, Pham H, Manning CD (2015) Effective approaches to attention-based neural machine translation. arXiv:1508.04025
30. Mihalcea R, Tarau P (2004) Textrank: Bringing order into texts. In: Lin D, Wu D (eds) Proceedings of EMNLP 2004, Association for Computational Linguistics, Barcelona, Spain, pp 404–411
31. Nallapati R, Zhai F, Zhou B (2017) Summarunner: a recurrent neural network based sequence model for extractive summarization of documents. hiP (yi= 1—hi, si, d) 1:1
32. Nallapati R, Zhou B, Gulcehre C, Xiang B et al (2016) Abstractive text summarization using sequence-to-sequence RNNS and beyond. arXiv:1602.06023
33. Niehues J, Cho E (2017) Exploiting linguistic resources for neural machine translation using multi-task learning. arXiv:1708.00993
34. Paulus R, Xiong C, Socher R (2017) A deep reinforced model for abstractive summarization. arXiv:1705.04304
35. Peng H, Thomson S, Smith NA (2017) Deep multitask learning for semantic dependency parsing. arXiv:1704.06855
36. Rush AM, Chopra S, Weston J (2015) A neural attention model for abstractive sentence summarization. arXiv:1509.00685
37. See A, Liu PJ, Manning CD (2017) Get to the point: summarization with pointer-generator networks. arXiv:1704.04368
38. Søgaard A, Goldberg Y (2016) Deep multi-task learning with low level tasks supervised at lower layers. In: Proceedings of the 54th annual meeting of the association for computational linguistics (volume 2: short papers), vol 2, pp 231–235
39. Sutskever I, Vinyals O, Le QV (2014) Sequence to sequence learning with neural networks. In: Advances in neural information processing systems, pp 3104–3112
40. Tan J, Wan X, Xiao J (2017) Abstractive document summarization with a graph-based attentional neural model. In: Proceedings of the 55th annual meeting of the association for computational linguistics (volume 1: long papers), vol 1, pp 1171–1181
41. Wu Y, Schuster M, Chen Z, Le QV, Norouzi M, Macherey W, Krikun M, Cao Y, Gao Q, Macherey K et al (2016) Google's neural machine translation system: bridging the gap between human and machine translation. arXiv:1609.08144
42. Wu S, Zhang D, Yang N, Li M, Zhou M (2017) Sequence-to-dependency neural machine translation. In: Proceedings of the 55th annual meeting of the association for computational linguistics (volume 1: long papers), vol 1, pp 698–707
43. Yang Z, Yang D, Dyer C, He X, Smola AJ, Hovy EH (2016) Hierarchical attention networks for document classification. In: HLT-NAACL, pp 1480–1489
44. Yao J, Wan X, Xiao J (2017) Recent advances in document summarization. Knowl Inf Syst 53(2):297–336
45. Zhang J, Yao J, Wan X (2016) Towards constructing sports news from live text commentary. In: Proceedings of the 54th annual meeting of the association for computational linguistics (volume 1: long papers), Association for Computational Linguistics, Berlin, Germany, pp 1361–1371. http://www.aclweb.org/anthology/P16-1129
46. Zoph B, Knight K (2016) Multi-source neural translation. arXiv:1601.00710