applied sciences

MDPI

Article

Abstract Text Summarization with a Convolutional Seq2seq Model

Yong Zhang *, Dan Li *, Yuheng Wang, Yang Fang and Weidong Xiao

Science and Technology on Information Systems Engineering Laboratory, National University of Defense Technology, Changsha 410073, China; wangyuheng12@nudt.edu.cn (Y.W.); fangyang12@nudt.edu.cn (Y.F.); wdxiao@nudt.edu.cn (W.X.)

* Correspondence: zhangyong_nudt@sina.com (Y.Z.); 13317310232@163.com (D.L.); Tel.: +86-(0)731-8700-6407 (Y.Z.)

Received: 25 March 2019; Accepted: 18 April 2019; Published: 23 April 2019

Abstract: Abstract text summarization aims to offer a highly condensed and valuable information that expresses the main ideas of the text. Most previous researches focus on extractive models. In this work, we put forward a new generative model based on convolutional seq2seq architecture. A hierarchical CNN framework is much more efficient than the conventional RNN seq2seq models. We also equip our model with a copying mechanism to deal with the rare or unseen words. Additionally, we incorporate a hierarchical attention mechanism to model the keywords and key sentences simultaneously. Finally we verify our model on two real-life datasets, GigaWord and DUC corpus. The experiment results verify the effectiveness of our model as it outperforms state-of-the-art alternatives consistently and statistical significantly.

Keywords: abstract text summarization; convolutional neural network; Seq2seq model

1. Introduction

Automatic abstract text summarization plays an essential part in natural language processing (NLP) and information retrieval (IR) tasks. Abstract text summarization is to generate a short summary which covers the core information of a passage or an article, consisting of a few keywords and sentences. It can be harnessed for other analyses or applications, e.g., text categorization [1], opinion mining [2] and so on. In this task, we do not simply extract the existing sentences or words from the original text, instead we generate a highly compressed paraphrasing of the core information of the text, even leveraging vocabulary absent from the source if needed.

Due to the fast development of deep learning, many sequence to sequence (in the following, we use seq2seq for simplicity) models are proposed to project an input sequence into another output sequence. They have been successful in many applications or tasks like speech recognition [3], video captioning [4] and machine translation [5]. Unlike the above tasks, in text summarization, the output sequence is much shorter than the input source sequence (document).

To realize the context summarization, [6] proposes an attentional seq2seq model based on RNNs which was originally developed for machine translation. It projects the the original document into low-dimensional embeddings. However, RNNs tend to have low-efficiency problems as they rely on the previous steps when training. Therefore, though not commonly used in sequence model, we propose a seq2seq model based on CNN to create the representations for the source texts. However, traditional CNNs can only encode fixed size contexts. To address this issue, we propose to enlarge the effective text by stacking CNN layers over each other. By doing so, the length of the sequence being dealt with can be easily controlled, and each element in the sequence could be computed parallelly.

In the contrast, RNN has to preserve the hidden state of the whole past, thus not able to perform parallelization operations.

Multi-layer CNNs create hierarchical representations on the input, i.e., close elements in a sequence interact at lower layers whereas distant elements interact at higher layers [7]. Unlike the chain structure of RNN, our multi-layer CNN offers a shortcut to parallelly express long-range sequences. For instance, to represent a sequence with n words, CNNs only need $O(\frac{n}{k})$ operations, and k is the kernel width, while RNNs need $O(n)$ operations. In addition, kernels and non-linearities numbers are fixed in CNN, whereas in RNN, the first word will need n operations and non-linearities, and the last word will also need a single set of operations [8]. We also add gated linear units (GLU) [9] and residual connections [10] in this convolutional model.

In the source document, the sentences could be very long, hence a summary could be drawn from not only the keywords but also the key sentences. So we firstly apply two CNNs on the source text, at word level and sentence level, respectively. Then to capture such hierarchical document structure, we propose a hierarchical attention mechanism applied on both levels simultaneously. We first calculate the word-level attention, then it will be re-weighted by its corresponding sentence-level attention.

In summarization task, it is common that the named-entities and keywords in the test text are essential to the summary, but in training data they may actually be rare or even unseen, and we call these words as out-of-vocabulary (OOV) words. To solve the issue, we introduce a copying mechanism [11] which enables the seq2seq model to extract the OOV words. It can distinguish the keywords based on their position or syntactic information from original text, even not knowing their exact meanings.

In this work we have three major contributions:

- We apply a convolutional seq2seq model to realize the text summarization which could help improve the model efficiency. And we also equip the CNN model with GLU and residual connections;
- We adopt a hierarchical attention mechanism to generate the keywords and the key sentences simultaneously. And we also introduce a copying mechanism to extract OOV words from source text.
- We evaluate our model on real-life datasets with other existing models. The experiment results on text summarization prove that our model has better performance than other advanced alternatives consistently and statistically significantly.

In the rest of the paper, we first present the related work in Section 2, and then introduce the mechanism of our model in detail in Section 3. Then we conduct extensive experiments on real-life datasets in Section 4. In the end, we present the conclusion of our findings in Section 5.

2. Related Work

2.1. Automatic Text Summarization

Lots of previous work in summarization focuses on extractive methods, that is, directly extract key words or sentences from the original document and then reproduce them as summary [12–15].

In contrast, humans tend to summarize the source text by using paraphrase in their own words and seldom reproduce the original sentences from the document. Inspired by this intuition, many researches used the DUC-2003 and DUC-2004 competitions http://duc.nist.gov/ to standardize such generative summarization tasks. In generative tasks, the datsets are composed of stories of news with diverse topics, and each story contains several reference summaries generated by humans. TOPIARY [16] utilized a combination of compression techniques motivated by linguistic, and an unsupervised topic detection method which appends keywords extracted from the story to the compressed output.Reference [17] used a conventional phrase-table based machine translation model to realize text summarization. Reference [18] leveraged weighted tree-transformation rules to compress the source text. Reference [19] utilized quasi-synchronous grammar methods to generate summaries.

Deep learning has been developing fast recently, and it can handle many NLP tasks [20]. So researchers have began to consider such framework as an effective, completely data-driven alternative for text summarization. Reference [21] used convolutional modes to encode the original text, and an attentional feed-forward neural network to generate summaries. But it did not use hierarchical CNNs, thus not that effective. Reference [22] is an extension to [23], and it replaced the decoder with an RNN. Reference [24] introduced a large Chinese dataset for short text summarization and it used a RNN based seq2seq model. [25] utilized a similar seq2seq model with RNN encoder and decoder, but it contained English corpora, reporting advanced results on DUC and Gigaword datasets. Reference [26] utilized the generative adversarial networks to generate the abstract text summarization, meanwhile equipped with a time-decay attention mechanism. Reference [27] was based on BERT to take advantage of the pre-trained language model in the seq2seq framework, and designed a two-stage decoder process to consider both sides' context information of every word in a summary. We denoted the above two methods as GAN and BERT respectively, and we will compare our model with them in experiments in detail. Reference [28] proposed a two-stage sentences selection model based on clustering and optimization techniques. Reference [29] proposed a text summarization model based on a LSTM-CNN framework that can construct summaries by exploring semantic phrases. Reference [30] presented an automatic process for text abstraction which was based on fuzzy rules on a variety of extracted features to find the most important information from the source text.

2.2. Seq2seq Model

Seq2seq modeling has been the synonym of encoder-decoder structures based on RNN [31]. Firstly, the encoder RNN handles an input sequence $\mathbf{x} = x_1, ..., x_m$, m denotes the number of elements and then obtains the corresponding hidden state $\mathbf{z} = (z_1, ..., z_m)$. The decoder RNN uses $\mathbf{z}$ as an input and generates the output $\mathbf{t} = (t_1, .., t_n)$ one by one from left to right. When generating the output t_{i+1}, decoder will produce a new hidden state h_{i+1} via the previous state h_i, along with a representation vector g_i of the previous target language word y_i, and a conditional input c_i based on the decoder input $\mathbf{z}$. Based on the above generic formulation, many seq2seq models try to have novelties on RNN type or conditional input.

If not considering adding the attention mechanism, models may merely utilize the final encoder state z_m by setting the conditional input c_i equal to z_m for each i [32] or initializing the input to the first decoder as z_m [31]. Those models [5,33] with attention mechanism calculated c_i as a sum of $(z_1, ..., z_m)$ with different weights at each time step. These weights of the sum are the so-called attention scores, which enable the decoder to focus on the different parts of the input sequence when generating the outputs. To compute attention scores, it needs to compare each encoder's output state z_j with a combination of the last prediction y_i and previous decoder state h_i. In the end, the results will be normalized to a distribution over the input sequence.

Many seq2seq model choose LSTM [34] and GRU [32] as RNN models. Both networks extend Elman RNNs with a gating mechanism which enables the memorization of information of previous time steps so as to process long-term dependencies. Recently, bi-directional encoders are proposed to capture both future and past contexts [5]. In practice, models with many layers are often equipped with shortcut or residual connection [10].

3. Model Description

In this section, we elaborate the mathematical mechanism of our abstract text summarization model in detail. After defining the text summarization task, we describe the CNN structure of our proposed seq2seq model. Afterwards we introduce the copying mechanism and hierarchical attention mechanism.

3.1. Problem Definition

Suppose a dataset with **N** data components, a component $(\mathbf{x^{(i)}}, \mathbf{t^{(i)}})$ is composed of a source text represented as $\mathbf{x^{(i)}}$ and M_i target keywords or key sentences denoted as $\mathbf{t^{(i)}} = (\mathbf{t^{(i,1)}}, \mathbf{t^{(i,2)}}, \ldots, \mathbf{t^{(i,M_i)}})$.

Both source document and target summaries consist of word or sentence sequences:

$$\mathbf{x^{(i)}} = x_1^{(i)}, x_2^{(i)}, \ldots, x_{L_{\mathbf{x^{(i)}}}}^{(i)},$$

$$\mathbf{t^{(i,j)}} = t_1^{(i,j)}, t_2^{(i,j)}, \ldots, t_{L_{\mathbf{t^{(i,j)}}}}^{(i,j)},$$

$L_{\mathbf{x^{(i)}}}$ and $L_{\mathbf{t^{(i,j)}}}$ denotes the length of $\mathbf{x^{(i)}}$ and $\mathbf{t^{(i,j)}}$, respectively.

Note that in each data component, one source text has several corresponding target keywords or key sentences. Then we transform the data component into pairs, that is, one source sequence only have one corresponding target sequence, so as to fit the proposed convolutional seq2seq model. Specifically, we split the data component into M_i pairs: $(\mathbf{x^{(i)}}, \mathbf{t^{(i,1)}}), (\mathbf{x^{(i)}}, \mathbf{t^{(i,2)}}), \ldots, (\mathbf{x^{(i)}}, \mathbf{t^{(i,M_i)}}))$.

3.2. Position Embeddings

For the input sequence $\mathbf{x} = (x_1, \ldots, x_m)$, we first represent it as a low dimensional vector $\mathbf{u} = (u_1, \ldots, u_m)$, where $u_j \in \mathbb{R}^d$. As for the position embeddings, we firstly obtain a one-hot vector to record the absolute position of a element in a sequence. Then we use an embedding layer to transform this sparse and discrete representation into a continuous embeddings as $\mathbf{p} = (p_1, \ldots, p_m)$ where $p_j \in \mathbb{R}^d$, thus enabling our model to sense which part of the sequence are being processed. d is the dimension of the position embeddings and input sequence element embeddings. We use the combination of both embeddings to form the input element embeddings $\mathbf{e} = (u_1 + p_1, \ldots, u_m + p_m)$. Similarly we also add the position embeddings to the output elements embeddings of decoder, and then feed them back to decoder.

3.3. Convolutional Seq2seq Model

We share the convolutional layer architecture on both encoder and decoder and they will calculate intermediate states via the input elements. We represent the output of l^{th} layer as $\mathbf{z^l} = (z_1^l, \ldots, z_m^l)$ for encoder and $\mathbf{h^l} = (h_1^l, \ldots, h_n^l)$ for decoder. Each layer consists of a one dimensional convolution and a non-linearity. If a decoder has one layer with kernel width being k, then its output h_i^1 will compress the information of k input elements. To enlarge the length of input elements, we stack blocks over each other, for example, stacking 6 blocks with $k = 5$ could represent 25 input elements. If needed, non-linearities enable our model to deal with the entire input sequence or only a few elements. The computational advantage of our model is that it conducts a parallel computation which is much more efficient than the traditional RNN model computed one element by one element. As mentioned in Section 1, to represent a sequence with n words, CNNs only need $O(\frac{n}{k})$ operations, while RNNs need $O(n)$ operations. However, our hierarchical CNN model is not that efficient than traditional CNN model as ours has to stack CNN layers to represent a sequence more expressively, while a traditional CNN model only needs one layer to explore a whole sequence.

In each convolution kernel, parameters are $W \in \mathbb{R}^{2d \times od}$, $b_w \in \mathbb{R}^{2d}$. The input is represented as $X \in \mathbb{R}^{o \times d}$, which is a matrix having o input elements with the dimension being d. Then the input is mapped by the layer to get the output being a single element $Y \in \mathbb{R}^{2d}$ with its dimension twice of that of the input. Then the o outputs elements will be fed to the subsequent layers. We leverage the gated linear units (GLU) [9] as non-linearity, which is applied on the output represented as $Y = [A\ B] \in \mathbb{R}^{2d}$:

$$v([A\ B]) = A \otimes \sigma(B), \tag{1}$$

in which the GLU takes $A, B \in \mathbb{R}^d$ as inputs, $\otimes$ denotes the element-wise multiplication. It is notable that $Y \in \mathbb{R}^{2d}$ is also twice the size of the output $v([A\ B]) \in \mathbb{R}^d$. $\sigma(B)$ is the gate that controls which inputs A of the source text are currently dealt with.

In order to realize deep CNN, we adopt the residual connections [10] which connect the input of each convolution layer with the ouput:

$$h_i^l = v(W^l[h_{i-o/2}^{l-1}, \ldots, h_{i+o/2}^{l-1}] + b_w^l) + h_i^{l-1}. \tag{2}$$

For encoder, to ensure that the output of the convolution blocks match the input elements length, we pad the input by $o-1$ elements at each layer with zero vectors on both the left and the right. Specifically, since no future information should be added to the decoder [35], o elements at the end of the convolution output will be removed.

Furthermore, we also apply linear mappings on the $2d$ convolution outputs so as to fit the embedding with its dimension being f. Such project is used to w when we feed the input embeddings to encoder. We also use it to encoder output z_j^i to compute the attention scores, along with the final layer of decoder before the softmax operation layer $\mathbf{h^L}$.

After all the operations above, the distribution on T possible next target elements t_{i+1} could be generated as follows:

$$p(t_{i+1}|t_1, \ldots, t_i, \mathbf{x}) = \text{softmax}(W_s h_i^l + b_s) \in \mathbb{R}^T, \tag{3}$$

where W_s denotes the weights in the linear layer and b_s is the bias.

3.4. Copying Mechanism

In text summarization task, to improve embedding quality and decrease the size of the softmax layer in decoder, it usually selects a relative small quantity of words with high frequency, for example, [32] adopts $30,000$ words as the vocabulary. However, it ignores those rare words, thus those keywords or key sentences containing OOV words are unable to be predicted. To solve the problem, we introduce the copying mechanism [11] that allows the seq2seq model to extract those OOV words from the original document. Copying mechanism is based on the fact that keywords or key sentences can be identified via position or syntactic information of the original text, even not knowing their exact meanings.

With copying mechanism, we can separate the the prediction probability of new word t_{i+1} into two components. One is the generative probability introduced in the above section, i.e., convolutional seq2seq model. The other is the words that are directly copied from the original document.

$$\begin{aligned} &p(t_{i+1}|t_1, \ldots, t_i, \mathbf{x}) \\ &= p_g(t_{i+1}|y_1, \ldots, t_i, \mathbf{x}) + p_c(t_{i+1}|t_1, \ldots, t_i, \mathbf{x}). \end{aligned} \tag{4}$$

Each word is assigned with a weight in original document by the copying mechanism. Such weight is able to evaluate the word's importance via positional attention score. As mentioned above, seq2seq model is limited to produce the keywords or key sentences from the vocabulary. However, with the help of copying mechanism, the model is able to not only extract those OOV words, but also extract present words since it is the truth that most keywords or key sentences exist in the original document. The detailed equation to compute the copying component is shown as

$$p_c(t_{i+1}|t_1, \ldots, t_i, \mathbf{x}) = \frac{1}{Z} \sum_{j:x_j=t_i} \exp(\Psi_c(x_j)), t \in \chi, \tag{5}$$

$$\Psi_c(x_j) = \sigma(\mathbf{h}_j^T \mathbf{W}_c)\mathbf{s}_t, \tag{6}$$

in which χ is the OOV words extracted from the original document, and σ denotes a non-linear function. $\mathbf{W}_c \in \mathbb{R}$ denotes a matrix with learnable parameters. $\mathbf{Z}$ is the summation of all scores, which denotes the normalization operation. Interested readers could see [11] for more details.

3.5. Hierarchical Attention Mechanism

In text summarization, especially for those very long documents, it is not enough to only identify the keywords, key sentences are also essential when forming the summaries. So we apply the convolutional seq2seq model on both words and sentences level. Then we adopt the hierarchical attention mechanism to simultaneously operate on both levels. After calculating the sentence-level attention, it will re-weight the corresponding word-level attention which is already computed. Then the whole attention score is renormalized:

$$Q^a(j) = \frac{Q^a_w(j)Q^a_s(s(j))}{\sum_{i=1}^{N_d} Q^a_w(i)Q^a_s(s(i))}, \tag{7}$$

where $Q^a_w(j)$ is the attention weight of word level appearing at j^{th} position, and $s(j)$ denotes the ID of the sentence at j^{th} word position $Q^a_s(s(j))$. Similarly, $Q^a_s(i)$ is the sentence-level attention weight for the l^{th} sentence in the origin. N_d denotes the number of the words in the origin text, and $Q^a(j)$ denotes the attention score after re-scaling at the j^{th} word position. Afterwards, we utilize the above attention score to compute the attention weighted context embeddings which is further fed into the decoder hidden state. The structure above is able to model key sentences along with the keywords within those sentences. Figure 1 describes the framework of this mechanism.

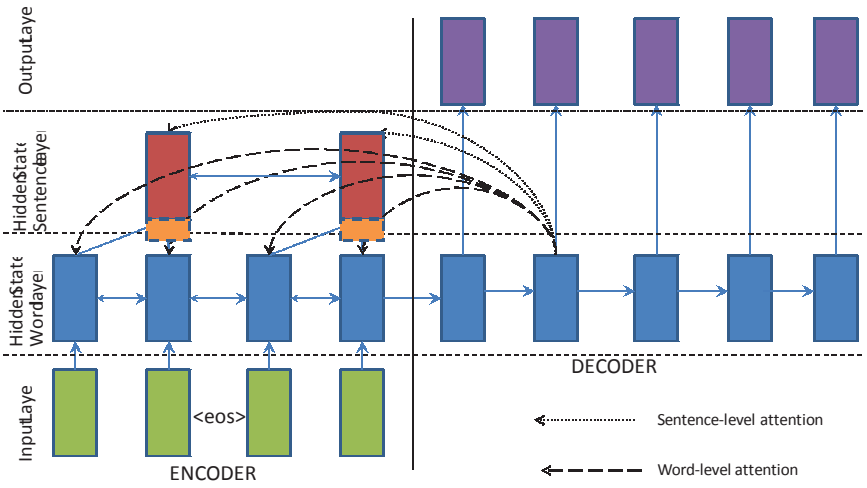

Figure 1. Model framework of hierarchal attention mechanism

4. Experiments and Results

We first introduce the datasets used in our experiments in detail, along with the experiment setup and the evaluation metric. Then we introduce the variants of our model and analyze the experiment results in detail. At last, we compare the computation costs among the models to demonstrate the efficiency of our model.

4.1. Training Datasets

We adopted two real-life training datasets, one is the annotated Gigaword corpus as depicted in [21], the other is the DUC corpus http://duc.nist.gov/duc2004/tasks.html. For Gigaword corpus,

to preprocess the data, we utilized the scripts which are made available by the authors of their work https://github.com/facebook/NAMAS, resulting in creating around 3.8 M training samples. The validation and testing datasets are produced around 400K samples, and a subset of 2000 samples are used for each validation and testing purpose. In our experiments, the performance is based on the above validation and testing datasets. To compare our moder with others more precisely, we also adopted the exact test samples utilized in [21]. For DUC corpus, it has two parts: one is the 2003 corpus composed of 624 documents and summary pairs, the other is 2004 corpus having 500 document and summary pairs. We use DUC-2003 as the training dataset and DUC-2004 as the testing dataset. However, the above DUC-2003 dataset is too small to be trained by large deep neural networks. If only training on Gigaword datasets via DNN, the model may not be able to capture the features of DUC datasets. So [21] utilizes a log-linear extractive model to train the DUC-2003 dataset. But it is not expressive enough to only use the log-linear extractive model, therefore, [21] combines it with the outputs of Gigaword datasets so that it can obtain a better testing results on DUC-2004 dataset. The model only trained on Gigaword is named as ABS, and the concatenated one is named as ABS+. In the experiments on DUC corpus, we take the same metric as these models to evaluate our model. We also include a new testing dataset which is a subset of CNN/Daily Mail dataset containing 11,490 test set samples. It is a collection of news stories and corresponding summarizations generated manually.

4.2. Experiment Setup

Before training, we firstly adopted the restricted Boltzmann machines (RBM) to help pre-process our datasets. RBM performs a transformation of the feature space based on an unsupervised learning step. Interested readers could read [36] for detailed information about RBM.

After pre-processing, to initialize the input embeddings, for all the models, we adopted 200 dimensional word2vec vectors [37] on the same corpus. These embeddings would be updated during the training. We set the dimension of all the hidden state of the encoder and decoder to be 400. As we only used the first sentence of the document as the source, the size of encoder vocabulary was 119,505 and that of decoder vocabulary was 68,885. For training, we utilized Adadelta and set the initial learning rate as 0.001. The batch size was set to 50 and we randomly shuffled the training data at every epoch. To speed up training, we sort every 10 batches according to their lengths at each epoch. We harnessed the gradient clipping instead of the dropout or regularization to prevent overfitting. On validation set, we adopted the early stopping mechanism [25] and utilized the best experimental setup based on the validation set to produce the testing results. We limited the decoder vocabulary size as 20,005 resulting a cutting down of the training time per epoch by almost three times. With the help of the selected vocabulary, in comparison with full vocabulary models, only 50–75% of the epochs are needed to converge.

When decoding, to generate the summary, we utilized beam search of size 5 and restricted the size of summary to 30 words as maximum, as we noticed that the maximum size in the sampled validation set is 30 as well. Specifically for DUC corpora, for each summary, we limit the model to generate exactly 30 words so as not to emit the end-of-summary tag. We also observed that the average summary length (7.4 to 8.1) of all our models fits the fact that the length of summaries in validation set has about 8.7 words, without tuning specifically.

In Gigaword, we directly apply our convolutional framework on the training datasets. However, the DUC dataset is too small that if training with our CNN model, it is likely to cause overfitting problem. Therefore, we use the DUC-2003 as the training datasets and apply a log-linear extractive model used in [21] to train the model and concatenate it with the training outputs of Gigaword corpus. Then the combined model is used to test the DUC-2004 corpus.

4.3. Evaluation Metrics

Similar as [22,35], to evaluate our model on Gigaword corpus, we utilize the full length F1 variant of Rouge http://www.berouge.com/Pages/default.aspx. Most published work utilized limited length recall [25], however, the defect is that different corpora have different length limits, so it is hard to compare performances among those corpora. Whereas full length recall does not have to restrict the length, but it unfairly favors longer summaries. So we propose to use Full-length F1 to solve the problem since it puts penalty on longer summaries, meanwhile with no length limit. Additionally, we present the proportion of summary tokens which also occurs in the original document, and it is named as 'src. copy rate'. However, on DUC corpus, the evaluation is based on limited-length Rouge recall as each summary is set to be 30 words.

4.4. Variants of Our Model

Here we describe the variants of our model that we conduct in our experiments.

RNN-1sent is our baseline model using attentional seq2seq model with the encoder and decoder being RNNs. As done in [21], this model is also trained only on the first sentence of the original text.

RNN-2sent is identical to the above model except that it used first two sentences for training from the source.

CNN-1sent is the model based on our convolutional seq2seq framework using the first sentence as input training datasets.

CNN-2sent is the same as the above one except that it takes first two sentences from the source as input.

CNN-2sent-copy is the above CNN-2sent model equipping copying mechanism proposed in Section 3.4.

CNN-2sent-hie is the above CNN-2sent model equipped with hierarchical attention mechanism proposed in Section 3.5.

CNN-2sent-hieco is the CNN-2sent model equipped with both copying mechanism and hierarchical attention mechanism.

CNN-2sent-hieco-RBM is the above CNN-2sent-hieco model with the RBM pre-processing of the datasets before training.

CNN-1sent-hieco is the CNN-1sent model equipping copying mechanism and hierarchical attention mechanism with taking the first sentence as input.

CNN-1sent-hieco-RBM is the above CNN-1sent-hieco model equipping with the RBM pre-processing.

4.5. Experimental Results

We first describe the experimental results on Gigaword corpus, the detailed information is illustrated in Table 1. The numbers in bold are the best figures. From which we can observe that adding one sentence in the training datasets does help improve the performance, on both RNN and CNN models. Actually in practice adding more sentences results in worse performance, and we attribute it to the fact that the latter sentences are not that related to the summary. Our hierarchical convolutional seq2seq model is more expressive than traditional model as CNN models outperform RNN models on both one sentence and two sentences conditions. We also observe that copying mechanism can aid the performance a lot with the highest src. copy rate. This means source document can help improving the summarization to a extend. The hierarchical attention mechanism can also improve the performance as it collaborates key words with key sentences simultaneously. No surprise that the CNN-2sent-hieco obtains the best results as it utilizes both copying mechanism and hierarchical attention mechanism. The * indicates the statistical significant improvement of all the other models using a paired two-tailed t test with $p < 0.05$. The src. copy rate dropped comparing the CNN-2sent-copy as hierarchical attention mechanism could improve the generative ability of the model, so it does not depend so much on the source. Actually, for ABS+, it has the highest src. copy rate whereas with worst performance,

and we attribute this to the truth that it is lacking in generative ability as a good summarization model needs both generative and extractive abilities. Our model CNN-2sent-hieco also beats the advanced model RASElman, GAN and BERT, which further proves the effectiveness of our model. Additionally, our model equipped with RBM achieves the even better results than our original CNN-2sent-hieco model, which verifies that RBM pre-processing could help improve the performance of our model to a certain extend.

Table 1. The experimental results on Gigaword Corpus. * indicates the statistical significant improvement.

Metric	Rouge-1	Rouge-2	Rouge-L	Src. Copy Rate
RNN-1sent	34.83	17.07	32.55	75.77
RNN-2sent	35.68	17.34	32.64	76.48
CNN-1sent	35.02	17.23	33.25	75.21
CNN-2sent	35.87	17.38	33.36	76.53
CNN-2sent-copy	36.32	18.03	33.87	84.12
CNN-2sent-hie	36.02	17.74	33.57	75.49
CNN-2sent-hieco	37.02	18.21	34.07	80.03
CNN-2sent-hieco-RBM	* **37.95**	* **18.64**	* **35.11**	80.74
ABS+	29.89	11.94	27.05	92.06
RASElman	36.26	17.92	33.91	83.25
GAN	36.88	17.54	34.05	76.54
BERT	36.97	17.96	33.87	74.51

As for the DUC corpus, we just ran the 1-sentence models as the source only contains one single sentence. Actually we only present the CNN-1sent-hieco and CNN-1sent-hieco-RBM results to save the space as they achieved the best performance among other variants. Table 2 illustrates the detailed experimental results. We can find that our model outperforms other models consistently with statistical significance, which proves our framework is a better alternative for the text summarization task.

Table 2. The experimental results on DUC Corpus. * indicates the statistical significant improvement.

Metric	Rouge-1	Rouge-2	Rouge-L
ABS	26.58	7.11	22.03
ABS+	28.17	8.51	23.85
RASELman	29.02	8.34	24.07
CNN-1sent-hieco	29.12	9.35	25.11
CNN-1sent-hieco-RBM	* **29.74**	* **9.85**	* **25.81**
GAN	29.05	8.27	24.69
BERT	28.97	9.21	24.89

We also tested our models on the new testing dataset CNN/Daily Mail and we directly adopted our best model CNN-2sent-hieco to compare with other state-of-the-art models. The results are presented in Table 3. From it we can observe that even changing the testing dataset, our model still consistently and statistical significantly outperforms other models, which proves that our model is suitable and scalable to other real-life datasets.

Table 3. The experimental results on CNN/Daily Mail testing datasets. * indicates the statistical significant improvement.

Metric	Rouge-1	Rouge-2	Rouge-L
ABS	35.68	16.47	34.14
ABS+	36.74	16.89	35.01
RASELman	39.65	18.25	37.24
CNN-2sent-hieco	41.87	19.64	39.35
CNN-2sent-hieco-RBM	* **42.04**	* **19.77**	* **39.42**
GAN	37.87	15.71	39.20
BERT	41.71	19.49	38.79

4.6. Computation Cost

We evaluate the computation cost among several variants of our model, along with other researches' work mentioned above. We extract a subset of Gigaword with 60 text sources. We measure the cost on GTX-1080 GPU. The results are illustrated in Table 4.

Table 4. The Computation Cost.

Model	Times
ABS	3211
ABS+	3865
RASELman	4573
GAN	6587
BERT	7432
RNN-1sent	7753
CNN-1sent	1681

We can easily find that our model based on CNN is much faster than other models, in which about 4 times faster than our RNN variant. The results proves the efficiency of our convolutional seq2seq framework.

5. Conclusions

In this work, we introduce a seq2seq generative model based on convolutional framework to produce abstract text summarization. Our proposed model can represent a sequence with a hierarchical structure which promotes its efficiency a lot. A copying mechanism is also adopted to further improve the model performance as it can handle the OOV words. Moreover, our model is able to model the keywords and key sentences simultaneously with the hierarchical attention mechanism. We conduct comprehensive experiments on two real-life datasets, and prove the effectiveness and efficiency of our model for generating abstract text summarization.

Author Contributions: Y.Z. realized the conceptualization, formal analysis and put forward a methodology, he also conducted the experiments and wrote the paper; D.L. helped data curation and coding; Y.W. reviewed and edited the paper, also helped results validation; Y.F. helped with the project administration; W.X. was the supervisor and helped the funding acquisition.

Funding: This research was funded by National Nature Science Foundation of China (NSFC) grant number 61872446 and 71690233, and NSF of Hunan province under grant No. 2019JJ20024.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Zhang, Y.; Zincir-Heywood, A.N.; Milios, E.E. World Wide Web site summarization. *Web Intell. Agent Syst.* **2004**, *2*, 39–53.
2. Berend, G. Opinion Expression Mining by Exploiting Keyphrase Extraction. In Proceedings of the Fifth International Joint Conference on Natural Language Processing, IJCNLP 2011, Chiang Mai, Thailand, 8–13 November 2011; pp. 1162–1170.
3. Bahdanau, D.; Chorowski, J.; Serdyuk, D.; Brakel, P.; Bengio, Y. End-to-end attention-based large vocabulary speech recognition. In Proceedings of the 2016 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2016, Shanghai, China, 20–25 March 2016; pp. 4945–4949, doi:10.1109/ICASSP.2016.7472618. [CrossRef]
4. Venugopalan, S.; Rohrbach, M.; Donahue, J.; Mooney, R.J.; Darrell, T.; Saenko, K. Sequence to Sequence—Video to Text. In Proceedings of the 2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, 7–13 December 2015; pp. 4534–4542, doi:10.1109/ICCV.2015.515. [CrossRef]
5. Bahdanau, D.; Cho, K.; Bengio, Y. Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv* **2014**, arXiv:1409.0473.
6. Nallapati, R.; Zhou, B.; dos Santos, C.N.; Gülçehre, Ç.; Xiang, B. Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond. In Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, Berlin, Germany, 11–12 August 2016; pp. 280–290.
7. Gehring, J.; Auli, M.; Grangier, D.; Yarats, D.; Dauphin, Y.N. Convolutional Sequence to Sequence Learning. In Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, Australia, 6–11 August 2017; pp. 1243–1252.
8. Zhang, Y.; Fang, Y.; Xiao, W. Deep keyphrase generation with a convolutional sequence to sequence model. In Proceedings of the 4th International Conference on Systems and Informatics, ICSAI 2017, Hangzhou, China, 11–13 November 2017; pp. 1477–1485, doi:10.1109/ICSAI.2017.8248519. [CrossRef]
9. Dauphin, Y.N.; Fan, A.; Auli, M.; Grangier, D. Language Modeling with Gated Convolutional Networks. In Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, Australia, 6–11 August 2017; pp. 933–941.
10. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778, doi:10.1109/CVPR.2016.90. [CrossRef]
11. Gu, J.; Lu, Z.; Li, H.; Li, V.O.K. Incorporating Copying Mechanism in Sequence-to-Sequence Learning. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, Berlin, Germany, 7–12 August 2016; Volume 1, Long Papers.
12. Neto, J.L.; Freitas, A.A.; Kaestner, C.A.A. Automatic Text Summarization Using a Machine Learning Approach. In Proceedings of the 16th Brazilian Symposium on Artificial Intelligence, SBIA 2002, Porto de Galinhas/Recife, Brazil, 11–14 November 2002; pp. 205–215, doi:10.1007/3-540-36127-8_20. [CrossRef]
13. Colmenares, C.A.; Litvak, M.; Mantrach, A.; Silvestri, F. HEADS: Headline Generation as Sequence Prediction Using an Abstract Feature-Rich Space. In Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL HLT 2015, The Denver, CO, USA, 31 May–5 June 2015; pp. 133–142.
14. Riedhammer, K.; Favre, B.; Hakkani-Tür, D. Long story short - Global unsupervised models for keyphrase based meeting summarization. *Speech Commun.* **2010**, *52*, 801–815, doi:10.1016/j.specom.2010.06.002. [CrossRef]
15. Ribeiro, R.; Marujo, L.; de Matos, D.M.; Neto, J.P.; Gershman, A.; Carbonell, J.G. Self reinforcement for important passage retrieval. In Proceedings of the 36th International ACM SIGIR conference on research and development in Information Retrieval, SIGIR '13, Dublin, Ireland, 28 July–1 August 2013; pp. 845–848, doi:10.1145/2484028.2484134. [CrossRef]
16. David Zajic, B.J.D.; Schwartz, R. Bbn/umd at duc-2004: Topiary. In Proceedings of the North American Chapter of the Association for Computational Linguistics Workshop on Document Understanding, Boston, MA, USA, 2–7 May 2004; pp. 112–119.

17. Banko, M.; Mittal, V.O.; Witbrock, M.J. Headline Generation Based on Statistical Translation. In Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics, Hong Kong, China, 1–8 October 2000.
18. Cohn, T.; Lapata, M. Sentence Compression Beyond Word Deletion. In Proceedings of the 22nd International Conference on Computational Linguistics, Proceedings of the Conference, COLING 2008, Manchester, UK, 18–22 August 2008; pp. 137–144.
19. Woodsend, K.; Feng, Y.; Lapata, M. Title Generation with Quasi-Synchronous Grammar. In Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, EMNLP 2010, Cambridge, MA, USA, 9–11 October 2010; pp. 513–523.
20. Collobert, R.; Weston, J.; Bottou, L.; Karlen, M.; Kavukcuoglu, K.; Kuksa, P.P. Natural Language Processing (Almost) from Scratch. *J. Mach. Learn. Res.* **2011**, *12*, 2493–2537.
21. Rush, A.M.; Chopra, S.; Weston, J. A Neural Attention Model for Abstractive Sentence Summarization. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, 17–21 September 2015; pp. 379–389.
22. Chopra, S.; Auli, M.; Rush, A.M. Abstractive Sentence Summarization with Attentive Recurrent Neural Networks. In Proceedings of the The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL HLT 2016, San Diego, CA, USA, 12–17 June 2016; pp. 93–98.
23. Sankaran, B.; Mi, H.; Al-Onaizan, Y.; Ittycheriah, A. Temporal Attention Model for Neural Machine Translation. *arXiv* **2016**, arXiv:1608.02927.
24. Hu, B.; Chen, Q.; Zhu, F. LCSTS: A Large Scale Chinese Short Text Summarization Dataset. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, 17–21 September 2015; pp. 1967–1972.
25. Nallapati, R.; Xiang, B.; Zhou, B. Sequence-to-Sequence RNNs for Text Summarization. *arXiv* **2016**, arXiv:1602.06023.
26. Rekabdar, B.; Mousas, C.; Gupta, B. Generative Adversarial Network with Policy Gradient for Text Summarization. In Proceedings of the 13th IEEE International Conference on Semantic Computing, ICSC 2019, Newport Beach, CA, USA, 30 January–1 February 2019; pp. 204–207, doi:10.1109/ICOSC.2019.8665583. [CrossRef]
27. Zhang, H.; Gong, Y.; Yan, Y.; Duan, N.; Xu, J.; Wang, J.; Gong, M.; Zhou, M. Pretraining-Based Natural Language Generation for Text Summarization. *arXiv* **2019**, arXiv:1902.09243.
28. Alguliyev, R.M.; Aliguliyev, R.M.; Isazade, N.R.; Abdi, A.; Idris, N. COSUM: Text summarization based on clustering and optimization. *Expert Syst.* **2019**, *36*, e12340, doi:10.1111/exsy.12340. [CrossRef]
29. Song, S.; Huang, H.; Ruan, T. Abstractive text summarization using LSTM-CNN based deep learning. *Multimed. Tools Appl.* **2019**, *78*, 857–875, doi:10.1007/s11042-018-5749-3. [CrossRef]
30. Goularte, F.B.; Nassar, S.M.; Fileto, R.; Saggion, H. A text summarization method based on fuzzy rules and applicable to automated assessment. *Expert Syst. Appl.* **2019**, *115*, 264–275, doi:10.1016/j.eswa.2018.07.047. [CrossRef]
31. Sutskever, I.; Vinyals, O.; Le, Q.V. Sequence to Sequence Learning with Neural Networks. In Proceedings of the Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, Montreal, QC, Canada, 8–13 December 2014; pp. 3104–3112.
32. Cho, K.; van Merrienboer, B.; Gülçehre, Ç.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, Doha, Qatar, 25–29 October 2014; pp. 1724–1734.
33. Luong, T.; Pham, H.; Manning, C.D. Effective Approaches to Attention-based Neural Machine Translation. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, 17–21 September 2015; pp. 1412–1421.
34. Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* **1997**, *9*, 1735–1780, doi:10.1162/neco.1997.9.8.1735. [CrossRef] [PubMed]
35. van den Oord, A.; Kalchbrenner, N.; Kavukcuoglu, K. Pixel Recurrent Neural Networks. In Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, 19–24 June 2016; pp. 1747–1756.

36. Mousas, C.; Anagnostopoulos, C.N. Learning Motion Features for Example-Based Finger Motion Estimation for Virtual Characters. *3d Res.* **2017**, *8*, 25. [CrossRef]
37. Pascanu, R.; Mikolov, T.; Bengio, Y. On the difficulty of training recurrent neural networks. In Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16–21 June 2013; pp. 1310–1318.