ITC 3/50
Information Technology and Control
Vol. 50 / No. 3 / 2021
pp. 458-469
DOI 10.5755/j01.itc.50.3.28047

Automatic Text Summarization Using Deep Reinforcement Learning and Beyond

Received 2020/11/20 | Accepted after revision 2021/06/17

http://dx.doi.org/10.5755/j01.itc.50.3.28047

HOW TO CITE: Sun, G., Wang, Z., Zhao, J. (2021). Automatic Text Summarization Using Deep Reinforcement Learning and Beyond. *Information Technology and Control*, 50(3), 458-469. https://doi.org/10.5755/j01.itc.50.3.28047

Automatic Text Summarization Using Deep Reinforcement Learning and Beyond

Gang Sun, Zhongxin Wang
School of Computer and Information Engineering, Fuyang Normal University, Fuyang 236037, China

Jia Zhao
School of Computer and Information Engineering, Fuyang Normal University, Fuyang 236037, China; Computer Science and Information Engineering, Hefei University of Technology, Hefei 230009, China

Corresponding author: wzxfync@ 163.com

In the era of big data, information overload problems are becoming increasingly prominent. It is challenging for machines to understand, compress and filter massive text information through the use of artificial intelligence technology. The emergence of automatic text summarization mainly aims at solving the problem of information overload, and it can be divided into two types: extractive and abstractive. The former finds some key sentences or phrases from the original text and combines them into a summarization; the latter needs a computer to understand the content of the original text and then uses the readable language for the human to summarize the key information of the original text. This paper presents a two-stage optimization method for automatic text summarization that combines abstractive summarization and extractive summarization. First, a sequence-to-sequence model with the attention mechanism is trained as a baseline model to generate initial summarization. Second, it is updated and optimized directly on the ROUGE metric by using deep reinforcement learning (DRL). Experimental results show that compared with the baseline model, Rouge-1, Rouge-2, and Rouge-L have been increased on the LCSTS dataset and CNN/DailyMail dataset.

KEYWORDS: AI, DRL, ROUGE metric, text summarization, LCSTS dataset, CNN/DailyMail dataset.

1. Introduction

Artificial intelligence technology is in a period of rapid development, and its application in various industries is becoming increasingly common. From medical diagnosis to social networks, and from intelligent education to news media, specific application cases of artificial intelligence can be seen everywhere. In people's daily lives, we are facing information overload, and how to deal with massive data information in limited time has become a problem. Using computers to understand natural language can filter useless and redundant information and only retain key information feedback to users. This specific application is called automatic text summarization. It uses a computer to summarize the whole text, helping users understand the core semantics of the original text directly by reading the abstract. Therefore, the machine learning model that automatically extracts summaries can quickly extract key information from massive texts, saving users valuable time. The emergence of automatic text summarization not only reduces information overload but also saves the high cost of manual text summarization.

Automatic text summarization is mainly divided into two types of methods. The extractive summarization technique extracts several key sentences from the original text and then forms a single abstract. The abstractive summarization technique understands the semantics of the original text and summarizes and induces the subject matter using a human-readable language. At present, extractive summarization is relatively easy to implement, so it is more widely used. The abstractive summarization requires the ability of the computer to understand the original text, so it has higher technical requirements.

Traditional extractive summarization techniques have two types: graph-based sorting methods and artificial feature-based methods. Based on the graph sorting method, each sentence in the document is used as a node of the graph, and calculate the similarity between sentences, and the value of the similarity is used as the weight of the edge to construct the graph model. Then, the PageRank algorithm [36] is used to solve each sentence score; finally, the highest scores are output as a summarization. The representative algorithms are TextRank [30] and LexRank [11]. The artificial feature-based method usually builds a model based on the length of each sentence, whether the sentence contains the title words. The representative algorithm is TextTeaser. With the rise of deep learning and its powerful feature representation ability, an increasing number of extractive summarization techniques are proposed, and excellent results have been obtained. Extractive summarization techniques do not need to consider the extracted summarization syntax and semantic problems. However, since the extracted summarization is only the combination of the original sentences, there are often problems such as inconsistency and information redundancy. Comparatively speaking, abstractive summarization is more convenient for humans to understand. The emergence of the deep learning-based sequence generation model [2] makes abstractive summarization a research hotspot. The seq2seq model [5] is applied as a benchmark model in the abstractive summarization task, and many deep learning models are emerging. The best results have been achieved on the corresponding datasets.

Aiming at the insufficient utilization of contextual semantic information, the insufficient semantic understanding of the attention mechanism, and the low accuracy of text summarization in the previous text summarization algorithms, this paper proposes an automatic summarization optimization algorithm, which first uses the seq2seq model with an attention mechanism [45] as the basic model for initial summarization generation and then uses depth enhancement learning to optimize the initial summarization directly through the ROUGE evaluation criteria [23]. The abstract generated by the base model is an abstractive summarization, and the optimization algorithm proposed selects the word output distribution of the base model in the decoding stage and the words of the top-k highest probability distributions that constitute the attention distribution of the original word as action spaces for enhanced learning. The initial summarization is optimized by the enhanced learning technique.

The main contributions of this paper are as follows:

1 This paper presents a two-stage optimization method for automatic text summarization that combines abstractive summarization and extractive summarization for the first time.

2 First, a sequence-to-sequence model with the attention mechanism is trained as a baseline model to generate initial summarization. Second, it is updated and optimized directly on the ROUGE metric by using deep reinforcement learning (DRL).

3 Compared with the basic model, Rouge-1, Rouge-2, and Rouge-L have been increased on the LCSTS dataset and CNN/DailyMail dataset. Therefore, the effect of the optimized method improved.

2. Related Work

2.1. Seq2seq Model

The seq2seq model is widely used in natural language processing [6, 34, 17]. The seq2seq model consists mainly of an encoder and a decoder. The encoder uses a cyclic neural network (RNN) such as the long short-term memory network [18] (LSTM) to encode the input sequence into a vector of fixed dimensions, and the decoder then uses RNN to decode the vector to produce an output sequence. Applying the attention mechanism [2] to the seq2seq model, it is possible to assign different weights to different parts of the input sequence during sequence generation. In natural language tasks, the seq2seq model typically uses a fixed vocabulary of input and output, which results in a poor representation of words that appear outside the vocabulary. The method of pointing to some unusual words or subsequences in an input sequence through a decoder network and then copying them directly into the output sequence [47, 24] can largely solve this problem. Gulcehre [13] and Merity [29] applied this pointing mechanism to the decoding process; then, the model can not only generate vocabularies in the vocabulary but also output uncommon words.

2.2. Reinforcement Learning and Sequence Generation

The emergence of Alpha Go has created considerable interest in artificial intelligence. Reinforcement learning is the most important technology in Alpha Go, which is a learning control strategy framework through computer algorithms. Given the agent and interaction environment [42], the agent can be trained to learn a strategy through reinforcement learning so that it can obtain the maximum reward. Compared with the traditional supervised learning method, reinforcement learning can be used to solve the problem when the agent has to perform discrete actions or when the optimization process is not defined. The process for optimizing sequence generation problems directly through metrics such as BELU, ROUGE, and METEOR is not divisible, so reinforcement learning can be applied to sequence generation tasks.

To achieve direct optimization of task evaluation criteria, Ranzato [38] used the REINFORCE algorithm [48] to train a cyclic neural network-based model for sequence generation tasks compared to traditional supervised learning. The results of the enhanced learning training method have been significantly improved. Bahdanau [1] proposed an evaluation-decision method to train neural network generation sequences, use a decision network to predict output actions, and then use an evaluation network to evaluate the value of the decision-making network to generate actions while also making the training process more stable. Rennie [39] proposed a self-assessment sequence generation training method without an additional decision network, and the evaluation results in the picture title generation task were significantly improved. Guo [15] proposed an iterative decoding output sequence based on the depth Q network [32] (DQN) training the sequence-to-sequence learning task. Guo's method [15] stimulates the automatic text summarization optimization method, generates an initial summarization and candidate action space through a seq2seq model with an attention mechanism, and directly optimizes the initial summarization evaluation standard (ROUGE) using DQN.

2.3. Automatic Text Summarization

Automated abstract research focuses on two areas: text [9, 35, 27] and speech [50, 51]. Although the abstractive summarization study has made some progress, most of the outstanding performance summarization models are still based on extractive methods. Traditional extractive summarization methods are mostly based on a greedy search [3] and graph model methods [11]. Kageback [21] implemented document summarization generation by deploying a recursive autoencoder [43]. Yin [49] used a convolutional neural network to minimize the objective function based on diversity and importance, and select sentences to generate summaries. Nallapati [34] adjusted a question and answer dataset of DeepMind [17] to a summarization dataset, the CNN/DailyMail dataset, and proposed the first benchmark model of the abstractive summarization on this dataset. On this dataset, Cheng and Lapata [5] proposed an attention-oriented encoder-decoder framework for abstract extraction. Nallapati [33] also proposed an model, which constructs a hierarchical cyclic neural network to select and extract original sentences.

Although the extractive summarization method is simpler and has some errors, there are also problems such as inconsistent semantics of the abstractive summarization context and unclear references. The generalized approach is freer and more in line with human writing and thinking patterns and can generate new and diverse sentences. With the advent of neural network-based text generation models [36], abstractive summarization technology is becoming a research hotspot. Rush [40] proposed an attention model with a convolutional encoder. On the CNN/DailyMail dataset, Chen[4] proposed a novel attention mechanism and applied it to the summarization generation model. On the CNN/DailyMail dataset, Nallapati [34] constructed a hierarchical network structure model using a hierarchical attention mechanism and pointer functions. On the same dataset, See [41] proposed a pointer network and additionally used a loss term for the attention-coverage mechanism in the loss function of its model. Patel [37] studied on abstractive and extractive content rundown strategies. Kejun [22] proposed an improved word vector generation technique and an abstractive automatic summarization model. Minaee [31] discussed more than 150 deep learning based models for text classification.

This paper proposed the automatic text summarization optimization method. First, the initial summarization is generated and the candidate action space required for reinforcement learning by deploying a seq2seq model. The action candidate space is divided into two parts. One part is generated by the decoder of the seq2seq model. This process can be regarded as an abstractive summarization method. The other part is generated by the attention mechanism of the seq2seq model, which can be seen as an extractive summarization method. Second, DQN [32] is used to learn a strategy to directly optimize the initial summarization to obtain the maximum reward (ROUGE score).

3. Modeling

This section mainly introduces the automatic text summarization optimization model, which is based on DQN. DQN is deployed through a cyclic neural network (GRU-RNN) with a gated recurrent unit [7] (GRU) in an encoder-decoder architecture. First, the pretraining phase is actually a parameter training of the seq2seq model (the lower half of Figure 1) with the attention mechanism through maximum likelihood estimation (MLE) and generates the initial summarization after reaching the convergence state (i.e., y_1^0, y_i^0, y_N^0 in the figure). In addition, the candidate action space of the DQN (i.e., DQN is the action apace in Figure 1, generated by the attention mechanism and decoder). Second, in the DQN iterative decoding stage, the parameters of the DQN are initialized using pretrained parameter values. This phase is an iterative decoding process. DQN learns a certain strategy, selects actions from the action space, and implements iterative optimization of the initial summarization to obtain the maximum reward. The reward is the ROUGE score obtained by the similarity between the summarization generated by DQN and the real summarization in the iterative process.

3.1. Automatics Summarization Modeling

Let the vocabulary be Ω, and its vocabulary size be $|\Omega| = V$. $\mathbf{x}$ is set to represent an input sentence containing a sequence of m words, that is, $\mathbf{x} = [x_1, ..., x_\mathrm{m}]$. Among them, each word $x_i \in \Omega$. The task of the automatic summarization is to generate a target sequence of length n words $y = [y_1,..., y_\mathrm{N}]$, such that $y = \arg\max_\mathrm{y} P(y|x)$ where $N < m$.

The automatic summarization model can be represented as a function $P(\mathrm{y}|\mathrm{x}) = P(\mathrm{y}|\mathrm{x}; \theta)$ with parameter θ, where θ is trained by maximizing the conditional probability of the {sentence-abstract} pairs in the training set. More specifically, given the last generated word, the training model generates the next word of the abstract.

$$P(y|\mathrm{x};\theta) = \prod_{t=0}^{N} p(y_t \mid \{y_0, ..., y_{t-1}\}, x; \theta). \tag{1}$$

Since (1) is difficult to solve, in practice, we instantiate this target in a serialized way, turning the original objective function into

$$L(\theta) = \max_{\theta} \sum_{t=0}^{N} \log p(y_t | \{y_0, ..., y_{t-1}\}, x; \theta). \quad (2)$$

This paper uses an encoder-decoder framework to model the conditional probability of (1).

3.2. Seq2seq Model

The encoder encodes the observed sample sequence X into the variable Z, and the decoder decodes the hidden variable Z into an output target sequence Y. The traditional seq2seq model encodes regardless of how long the sequence of observation samples is into a fixed-dimensional hidden variable Z. Obviously, such an operation limits the ability of the seq2seq model. Therefore, Bahdanau [2] proposed a recurrent neural network search model. On the decoder side, using a hidden layer forward network, an adaptive method is used to calculate the weight of each word in the observation sequence X and the output target sequence Y.

This paper introduces the encoder-decoder framework to the automatic summarization problem takes the pure data-driven approach and trains the automatic summarization end-to-end model. As the basic model of this paper, this model generates the initial summarization and candidate action space of the DQN model.

Encoder: The network gated unit can better express the long-term and short-term dependence. In this paper, a cyclic neural network with a gated recurrent unit (GRU [8]) is used as the basic module for constructing a document encoder. A GRU consists of u and r:

$$u_t = \sigma(W_{ux}x_t + W_{uh}h_{t-1} + b_u) \quad (3)$$

$$r_t = \sigma(W_{rx}x_t + W_{rh}h_{t-1} + b_r) \quad (4)$$

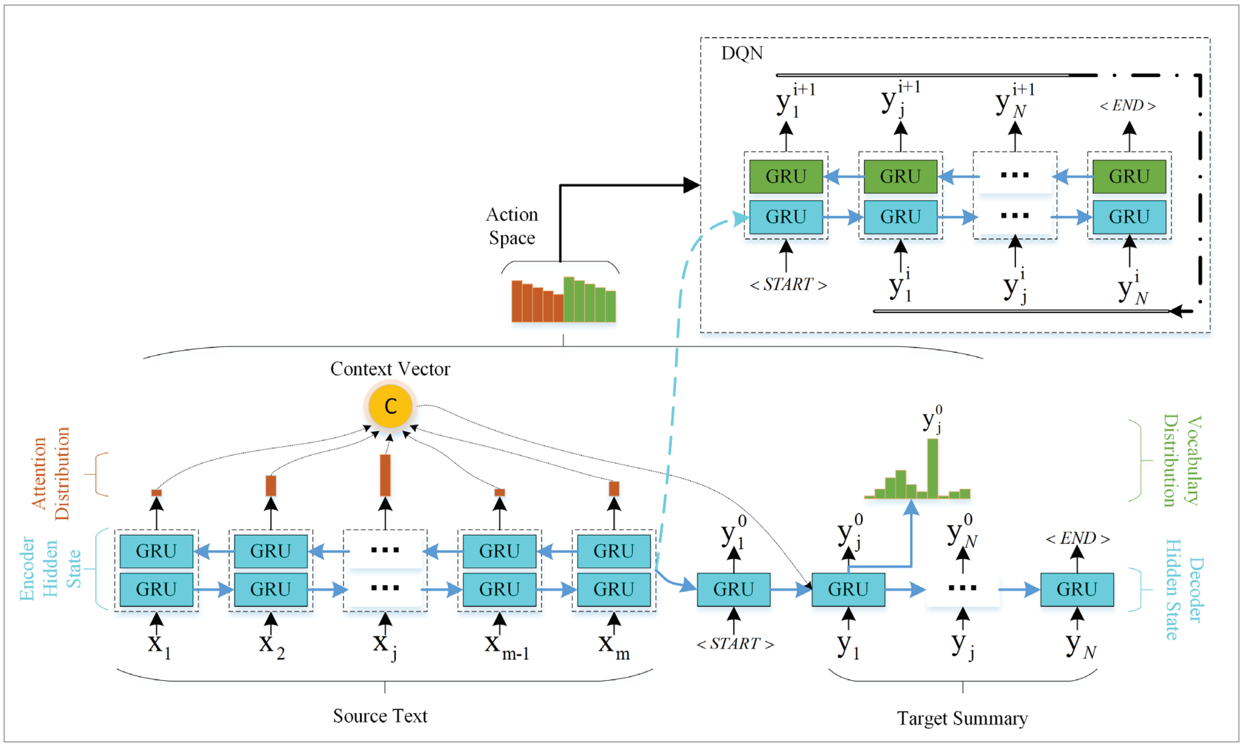
$$h'_t = \tanh(W_{hx}x_t + W_{hh}(r_t \odot h_{t-1}) + b_h) \quad (5)$$

$$h_t = (1 - u_t) \odot h'_t + u_t \odot h_{t-1}, \quad (6)$$

where W and b represent hyperparameters of the GRU; x_t represents the input word vector at time t; h_t denotes the hidden layer vector $\sigma(\cdot)$; $\tanh(\cdot)$ is the activation function at the corresponding time; and $\odot$ is the bitwise multiplication operation.

Figure 1

The optimization model for automatic text summarization



The forward GRU generates a hidden layer vector representation of each corresponding word vector of the forward sequence, while the backward GRU generates the hidden layer vector of each corresponding word vector of the reverse sequence.

Forward GRU generates forward sequences, and the hidden layer vector of each corresponding word vector represents h_t^f, while backward GRU generates the hidden layer vector h_t^b of each corresponding word vector of the reverse sequence.

$$h_t=[h_t^f, h_t^b]. \tag{7}$$

The input sequence of the original text is used as the input of the encoder, and the hidden layer representation of each position is generated by the encoder. The semantic representation c of the input sequence can be directly assigned by the last hidden layer of the encoder, i.e., $c = h_m$, or it can can be the linear representation of the last hidden layer.

Decoder and Attention mechanism: The semantic feature representation of the input sequence generated by the encoder is decoded as the initial input state of the decoder out of the target text sequence, i.e., the summarization. The decoder is essentially a language model. This article also uses a one-way GRU, as shown in the lower right part of Figure 1. c is decoded by the decoder so it must contain all the information in the original sequence. Additionally, in the process of generating a text sequence, each word is generated using the same semantic vector. Obviously, this method is too simple.

To solve the abovementioned problems, a feasible solution is to introduce an attention mechanism. The attention mechanism gives different attention weights to different input words at each decoding time and generating each word. The semantic representation c at each time in the decoding process adaptively selects the most appropriate context information for the target y_i to be output at the current time. Specifically, we use a_{ij} to measure the correlation of the encoder's hidden layer representation h_j at time j and decoding at time i. Finally, the context information ci input by the decoder at time i is equal to the hidden layer of the encoder at all times, which represents the weighted sum of h_j and a_{ij}.

$$s_i = f(s_{i-1}, y_i, c_i) \tag{8}$$

where,

$$c_i=\sum_{j=1}^{M} a_{ij} h_j \tag{9}$$

$$a_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{M} \exp(e_{ik})} \tag{10}$$

$$e_{ij}=v_a^T \tanh(W_a s_{i-1} + U_a h_j). \tag{11}$$

Using the decoder with the attention mechanism, the model of equation (1) can be expressed as

$$p(y_i|\{y_0,\dots,y_{i-1}\},x;\theta) = g(s_i), \tag{12}$$

where g(·) nction that outputs the probability of the target vocabulary y_i. A softmax function is usually used to map a hidden layer vector into a probability distribution of V categories (i.e., vocabulary) of the vocabulary:

$$p(y_i = z|\theta) = g(s_i)=\frac{\exp(w_z^T s_i)}{\sum_{z=1}^{V} \exp(w_z^T s_i)}, \tag{13}$$

where w_z represents the weight matrix.

3.3. DQN Decoding

The four components of the reinforcement learning model are described in detail: states, actions, strategies, and rewards. Automatically generate semantic features of the input sequence by deploying a recurrent neural network with a coder-decoder architecture with a gated unit. DQN directly uses these features to learn the Q-value function to estimate future rewards and to maximize future long-term rewards by optimizing model parameters.

Given the abstract text sequence $<x_1,\dots, x_j \dots, x_m>$ and the target sequence $<y_1,\dots, y_j \dots, y_N>$ (source text and target summarization in Figure 1), the Bi-GRU encoder encodes the entire input sequence into a fixed-dimensional semantic vector that is decoded as the initial input vector of the decoder to generate the actual initial summarization sequence $< \hat{y}_1^0,\dots,\hat{y}_j^0\dots,\hat{y}_T^0 >$. In the decoding process, the decoder sequentially generates a hidden layer state representation $<s_1,\dots, s_j,\dots, s_N$.

Status: At each iteration step i of DQN decoding, the summarization text input sequence $<x_1,\dots,x_j\dots,x_m>$ and the actual summarization sequence $< \hat{y}_1^i,\dots,\hat{y}_j^i\dots,\hat{y}_T^i >$

are used as DQN internal states. When i=0, the initial summarization generated by the decoder consists of the vocabulary of the maximum output probability of each time decoder.

Action: In the decoding stage of the decoder, the hidden layer vector is mapped to the probability distribution of V vocabularies in the vocabulary by a softmax function. At the same time, it also produces the attention weight distribution of the summarization text sequence at each time. The vocabulary that generates the vocabulary probability distribution and the first k maximum probabilities of the attention distribution are selected as the candidate action space (size 2k) for each time DQN, as shown in Figure 1. The action candidate set generated by the seq2seq model encoding-decoding is an "abstractive summarization" generation process, and the action candidate set generated at each time based on the summarization original attention distribution can be regarded as an extractive summarization process.

Strategy: Given the current state, i.e., the current iteration step summarization sequence $< \hat{y}_1^i,..., \hat{y}_j^i ..., \hat{y}_T^i >$, and the action space at each time, the DQN learns and estimates the value function (Q-value) at each time of the current state and selects the action of the maximum Q-value at each time as the output. More specifically, in the i-th iteration step at time t, the DQN selects the action $\tilde{y}_t^i$ to replace the state $\tilde{y}_t^i$ corresponding to the time s. Therefore, this process will result in an update of the state, i.e., a new state $< \hat{y}_1^{i+1},..., \hat{y}_j^{i+1} ..., \hat{y}_T^{i+1} >$.

Reward: In iteration step i, the ROUGE score of the current summarization sequence $< \hat{y}_1^i,..., \hat{y}_j^i ..., \hat{y}_T^i >$ and the reference summarization $<y_1,..., y_j ..., y_N>$ is set to r_c. The DQN selects actions based on the current policy. This process generates a new state $< \hat{y}_1^{i+1},..., \hat{y}_j^{i+1} ..., \hat{y}_T^{i+1} >$, which is the optimized summarization. The ROUGE score of the abstract versus the real abstract is r, so the reward for the DQN's current action is

$$\text{reward}=r_n - r_c. \tag{14}$$

Parameter training: The DQN parameter training is a process of minimizing the loss function. In the actual training process, Two networks are used to improve the stability [21]: A Q-value estimate with the parameter θ original network and a target network with parameter $\theta\theta$ generates the target q-value during the Q-value learning update process.

$$L^i(\theta^i) = E_{s,a}[(q^i - Q(s,a;\theta^i))^2], \tag{15}$$

where $q^i = E_{s,a}[r^i + \lambda \max_{a'} \hat{Q}(s^{i+1}, a'; \hat{\theta}) \mid s, a]$ is the target q-value, generated by the target network with the parameter θ according to the next state s^{i+1}. This process can be seen as training the DQN to predict future rewards.

In the training process, in every C update training, the parameter θ of the original network is assigned to the parameters of the target network, i.e., $\hat{\theta}=\theta$. Additionally, experience pool technology[25] is used in training to store the experience of each iteration step. DQN selects the action of maximum Q to maximize future expectations. In the initial stage of training, DQN randomly selects actions with probability ε [20] to ensure sufficient exploration of the state space. Its specific parameter training process is referenced in algorithm 1.

Algorithm 1. Automatic text summarization optimization algorithm.

```
1.  Random parameters initialize the encoder and decoder
    of the seq2seq model;
2.  Pretrain the seq2seq model given the training set, i.e.the
    original and real summarization sequence pairs;
3.  for epoch = [1, U] do
4.    for each x ∈ X and y∈Y.(The length of l) do
5.    The original sequence x is input to the pretrained
      seq2seq model to generate an initial summarization
      sequence $\hat{y}^0$ and an action candidate space $A$;
6.      for iteration $i$ = 1, 2$l$ do
7.        if the value is less than $\varepsilon$
8.          Randomly select an action $a^i$ (i.e., word) from the
            action space $A$ at time t;
9.        else
10.         Calculate the Q-value function $Q(s, a; \theta)$, let action
            $a^i = \arg\max_a Q(s, a; \theta)$;
11.       end if
12.     Using the action $a^i$ instead of generating the element
        (word) corresponding to the time of the summarization
        sequence $\hat{y}^i$ a new summarization sequence is
        generated;
13.     Calculate the similarity between the newly
        summarization sequence $\hat{y}^{i+1}$ and the real
        summarization sequence y and return the reward $r^i$;
14.     Store state experience $[s^i, a^i, r^i, s^{i+1}]$ to experience pool D,
        where $s^i$=<x, $\hat{y}^i$ >
```

```
15.  Randomly sample batch state experience $[s^i, a^i, r^i, s^{i+1}]$
     from experience pool D;
16.      if $r^i > \sigma$ $r^i > \sigma$
17.          $q^i = r^i$; This round of iterative update ends;
18.      else
19.          $q^i = r^i + \lambda \max_{a'} \hat{Q}(s^{i+1}, a'; \hat{\theta})$
20.      end if
21.  Perform the optimization on the objective function
     $(q^i - Q(s^i, a^i; \theta))^2$ with respect to the parameter $\theta$;
22.  Update the parameters every C iterations of the
     iteration, let $\hat{\theta} = \theta$
23.  end for
24. end for
25. end for
```

4. Experiments

This section details the summarization of the evaluation indicators, datasets and automatic summarization generation-related comparison algorithms and tests our methods on two datasets (LCSTS and CNN/DailyMail).

4.1. Evaluation

In this paper, the ROUGE [23] evaluation system is used to automatically evaluate the summarization. ROUGE evaluates the quality of the summarization between the reference summarization and the summarization generated by the abstract system. ROUGE includes a series of evaluation methods. The summarization task usually uses ROUGE-N and ROUGE-L.

$$R = \frac{\sum_{S \in \{GenSummaries\}} \sum_{gram_n \in S} Count_{match}(gram_n)}{\sum_{S \in \{RefSummaries\}} \sum_{gram_n \in S} Count(gram_n)} \tag{16}$$

$$P = \frac{\sum_{S \in \{GenSummaries\}} \sum_{gram_n \in S} Count_{match}(gram_n)}{\sum_{S \in \{GenSummaries\}} \sum_{gram_n \in S} Count(gram_n)} \tag{17}$$

$$F = \frac{2 \times R \times P}{R + P}, \tag{18}$$

where n is the length of the n-gram, which takes 1 and 2 in this evaluation.

The formula for calculating accuracy P, recall rate R and F of ROUGE-L is as follows:

$$R = \frac{LCS(X, Y)}{m} \tag{19}$$

$$P = \frac{LCS(X, Y)}{n} \tag{20}$$

$$F = \frac{(1+\beta^2) \times R \times P}{R + \beta^2 \times P}, \tag{21}$$

where X is a reference summarization and has a length of m, Y is summarization generated by the system, the length is n, $LCS(\cdot)$ is used to measure the longest common subsequence and β is a hyperparameter.

4.2. Dataset

LCSTS dataset[19]: LCSTS is a Chinese short text summarization dataset. Each piece of data was collected from Sina Weibo in the form of a {short news, summarization} pair. The dataset includes PARTI, PARTII, and PARTIII. The specific statistical information is shown in Table 1. In this paper, using the same settings as in [14].

Table 1
Statistics of the LCSTS Dataset

LCSTS	PART I	PART II	PART III
Quantity	2400059	10666	1106
Quantity (≥3′)	-	8685	725

CNN/DailyMail dataset: The CNN/DailyMail dataset is an artificially generated summarization dataset constructed by Herman [18] based on news articles. This dataset can be obtained directly from GitHub. This paper uses the Stanford CoreNLP tool [28] to preprocess the data. We set the maximum length of the abstract text and the corresponding summarization to 400 and 100.

4.3. Experiment Setup

The encoder is built using a single-layer bidirectional Bi-GRU as the base module. In addition, a backward GRU and a forward GRU of the decoder are used together to form our DQN. The number of GRU hidden layers is set to 256. Model network parameters are initialized using a uniform distribution of intervals [-0.1, 0.1]. The learning rate and batch size are set to 0.05 and 32, respectively. We first pretrain the base model on a given dataset until it reaches a convergence state and then train the DQN. In the initial stage of DQN training, the state space exploration parameter is set to 1.0, and the step is gradually reduced to 0.1 in 1,000 steps. Given that the length of the output sequence is l, its DQN iteration number is set to 2l [15]. The iteration termination threshold is set to 0.9. The DQN experience pool size is set to 200,000, the reward discount factor is set to 0.9, and the action space size is set to 20; that is, the 10 most likely words from the vocabulary output distribution and the attention distribution are chosen.

In the LCSTS dataset, the first 50,000 words in the training set were selected to form a vocabulary. Other words were uniformly represented by the <UNK> tag, and the word vector dimension was set to 200.

On the CNN/DailyMail dataset, the first 25,000 most frequently appeared words were selected to form the vocabulary, and other words were uniformly represented by the <UNK> tag. The word vector dimension is set to 128.

4.4. Experimental Results and Analysis

1 Evaluation results on the LCSTS dataset:

Except for the “Bi-GRU + Distraction” method, the method and comparison method use the basic unit after the word segmentation as the word vector to input to the model. The experimental results are shown in Table 2, where “Bi-GRU” represents the basic model of this paper. Additionally, an attention mechanism is used in the decoding process to generate the final abstract by using a beam search and decoding with a width of 10. “DQN” represents the optimization model; that is, the results of the basic model are optimized by the depth Q network. On this dataset, the comparison methods we use are as follows.

RNN: the benchmark model first proposed by Hu [19] on the LCSTS dataset only uses a recurrent neural network as the implementation summarization of the encoder and decoder.

RNN context: A reinforcement model proposed by Hu [19]. The difference is that in the decoding process, all hidden layer states of the encoder are input to the decoder as context.

Table 2
Result on the LCSTS Dataset

Methods	Rouge-1	Rouge-2	Rouge-L
RNN	17.7	8.5	15.8
RNN context	26.8	16.1	24.1
COPYNET	35.0	22.3	32.0
Bi-GRU + Distraction	35.2	22.1	32.5
Bi-GRU	28.4	19.2	28.5
DQN	35.7	22.6	32.8

COPYNET: A replication mechanism was proposed by Gu [14] and incorporated into sequence-to-sequence learning. COPYNET can combine the replication mechanism with the sequence generation process of the traditional decoder so that it can directly select the corresponding subsequence.

Bi-GRU + Distraction: A novel attention mechanism method proposed by Chen [4], which differs from the above method in that the test result is based on characters as a basic unit, that is, a Chinese character.

The basic model “Bi-GRU” used in this paper and the “RNN context” method used by Hu [19] are based on GRU construction. Unlike the latter, our basic model constructs an encoder and also adds attention mechanisms in the decoding process. Therefore, our base model performs better than the “RNN context” method. The “COPYNET” method and the “Bi-GRU + Distraction” method can be seen as introducing a replication mechanism and a new attention mechanism on top of our basic model. Our optimization model “DQN” introduces the optimization process of reinforcement learning to achieve the initial summarization of the basic model. From the experimental results, we can conclude that our optimization model achieves the best results.

2 Evaluation results on the CNN/DailyMail dataset:

As seen in Table 3, the comparison method we used includes some methods used on LCSTS data, including TextRank [30], LexRank [11], Luhn[26], Edmundson [10], LSA [44], Sum-basic [15] and KL-sum [16]. The experimental results of these methods can be achieved through the open source tool SUMY.

Table 3
Result on the CNN/DailyMail Dataset

Methods	Rouge-1	Rouge-2	Rouge-L
Luhn	23.2	7.2	15.5
Edmundson	24.5	8.2	16.7
LSA	21.2	6.2	14.0
LexRank	26.1	7.9	17.7
TextRank	23.3	7.7	15.8
Sum-basic	22.9	5.5	14.8
KL-sum	20.7	5.9	13.7
Bi-GRU+ Distraction	27.1	8.2	18.3
Bi-GRU	19.3	5.3	14.8
DQN	27.2	9.6	18.7

From the experimental results, we can conclude that our optimization model achieves the best results.

In the two datasets, the optimization effect in the ROUGE-1 evaluation was significantly higher than that of ROUGE-2 and ROUGE-L. Since our DQN model is more likely to be rewarded for selecting an individual action from the action candidate space each time in the iterative optimization summarization, the ROUGE-1 score is more likely to be updated.

5. Conclusions

This paper proposes a reinforcement text summarization optimization method based on deep enhanced learning. An attention mechanism-based seq2seq model is used to generate the initial summarization and the action candidate space required for reinforcement learning, and then the deep Q network is used to optimize the initial summarization on the action candidate space. The experimental results show that the effect of the optimized method obviously improved. Since the results generated by the base model limit the final performance of our optimization method, in future work, we consider applying reinforcement learning directly to optimize the parameters of the base model to obtain better results.

Acknowledgement

This work is supported in part by the National Natural Science Foundation of China under Grants 61906044 and 61672006, and in part by Big Data and Intelligent Computing Innovation Team of Fuyang Normal University under Grants XDHXTD201703, and in part by Fuyang Humanities and Social Science Research Special Project under Grants FYSK2019QD10, and in part by Natural Science Research Project of Fuyang Normal University under Grants 2020XXGN01, and in part by the key projects of natural science research in Anhui colleges and universities under Grants KJ2019A0529, KJ2019A0532, KJ2019A0542 and KJ2020ZD48.

References

1. Bahdanau, D., Brakel, P., Xu, K., et al. An Actor-Critic Algorithm for Sequence Prediction. ArXiv preprint arXiv:1607.07086, 2016.
2. Bahdanau, D., Cho, K., Bengio, Y. Neural Machine Translation by Jointly Learning to Align and Translate. ArXiv preprint arXiv:1409.0473, 2014.
3. Carbonell, J., Goldstein, J. The Use of MMR, Diversity-Based Reranking for Reordering Documents and Producing Summaries. Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. ACM, 1998, 335-336. https://doi.org/10.1145/290941.291025
4. Chen, Q., Zhu, X. D., Ling, Z. H., et al. Distraction-Based Neural Networks for Modeling Document. IJCAI, 2016, 2754-2760.
5. Cheng, J., Lapata, M. Neural Summarization by Extracting Sentences and Words. ArXiv preprint arXiv:1603.07252, 2016. https://doi.org/10.18653/v1/P16-1046
6. Chopra, S., Auli, M., Rush, A. M., et al. Abstractive Sentence Summarization with Attentive Recurrent Neural Networks. HLT-NAACL, 2016, 93-98. https://doi.org/10.18653/v1/N16-1012
7. Chung, J., Gulcehre, C., Cho, K., et al. Gated Feedback Recurrent Neural Networks. International Conference on Machine Learning, 2015, 2067-2075.
8. Chung, J., Gulcehre, C., Cho, K. H., et al. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. ArXiv preprint arXiv:1412.3555, 2014.

9. Das, D., Martins, A. F. T. A Survey on Automatic Text Summarization. Literature Survey for the Language and Statistics II course at CMU, 2007, 4, 192-195.

10. Edmundson, H. P. New Methods in Automatic Extracting. Journal of the ACM (JACM), 1969, 16(2), 264-285. https://doi.org/10.1145/321510.321519

11. Erkan G, Radev D R. Lexrank: Graph-based lexical centrality as salience in text summarization. Journal of Artificial Intelligence Research, 2004, 22: 457-479. https://doi.org/10.1613/jair.1523

12. Ganesan, K., Zhai, C. X, Han, J. Opinosis: A Graph-Based Approach to Abstractive Summarization of Highly Redundant Opinions. Proceedings of the 23rd International Conference on Computational Linguistics, Association for Computational Linguistics, 2010, 340-348.

13. Gulcehre, C., Ahn, S., Nallapati, R., et al. Pointing the Unknown Words. ArXiv preprint arXiv:1603.08148, 2016. https://doi.org/10.18653/v1/P16-1014

14. Gu, J., Lu, Z., Li, H., et al. Incorporating Copying Mechanism in Sequence-to-Sequence Learning. ArXiv preprint arXiv:1603.06393, 2016. https://doi.org/10.18653/v1/P16-1154

15. Guo, H. Generating Text with Deep Reinforcement Learning. ArXiv preprint arXiv:1510.09202, 2015.

16. Haghighi, A., Vanderwende, L. Exploring Content Models for Multi-Document Summarization. Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Association for Computational Linguistics, 2009, 362-370. https://doi.org/10.3115/1620754.1620807

17. Hermann K M, Kocisky T, Grefenstette E, et al. Teaching machines to read and comprehend. Advances in Neural Information Processing Systems. 2015: 1693-1701

18. Hochreiter, S., Schmidhuber, J. Long Short-Term Memory. Neural Computation, 1997, 9(8), 1735-1780. https://doi.org/10.1162/neco.1997.9.8.1735

19. Hu, B., Chen, Q., Zhu, F. Lcsts: A Large Scale Chinese Short Text Summarization Dataset. ArXiv preprint arXiv:1506.05865, 2015. https://doi.org/10.18653/v1/D15-1229

20. Kaelbling, L. P., Littman, M. L., Moore, A. W. An Introduction to Reinforcement Learning. The Biology and Technology of Intelligent Autonomous Agents. Springer, Berlin, Heidelberg, 1995, 90-127. https://doi.org/10.1007/978-3-642-79629-6_5

21. Kågebäck, M., Mogren, O., Tahmasebi, N., et al. Extractive Summarization Using Continuous Vector Space Models. Proceedings of the 2nd Workshop on Continuous Vector Space Models and their Compositionality (CVSC), 2014, 31-39. https://doi.org/10.3115/v1/W14-1504

22. Kejun, Z., Weinan, L. I., Rong, Q., et al. Automatic Text Summarization Scheme Based on Deep Learning. Journal of Computer Applications, 2019, 311-315.

23. Lin, C. Y. Rouge: A Package for Automatic Evaluation of Summaries. Text summarization branches out: Proceedings of the ACL-04 workshop, 2004, 8.

24. Ling, W., Grefenstette, E., Hermann, K. M., et al. Latent Predictor Networks for Code Generation. ArXiv preprint arXiv:1603.06744, 2016. https://doi.org/10.18653/v1/P16-1057

25. Lin, L. J. Reinforcement Learning for Robots Using Neural Networks. Carnegie-Mellon University Pittsburgh PA School of Computer Science, 1993.

26. Luhn, H. P. The Automatic Creation of Literature Abstracts. IBM Journal of Research and Development, 1958, 2(2), 159-165. https://doi.org/10.1147/rd.22.0159

27. Mani, I. Automatic Summarization. John Benjamins Publishing, 2001. https://doi.org/10.1075/nlp.3

28. Manning, C. D., Surdeanu, M., Bauer, J., et al. The Stanford Corenlp Natural Language Processing Toolkit. ACL (System Demonstrations), 2014, 55-60. https://doi.org/10.3115/v1/P14-5010

29. Merity, S., Xiong, C., Bradbury, J., et al. Pointer Sentinel Mixture Models. ArXiv preprint arXiv:1609.07843, 2016.

30. Mihalcea, R., Tarau, P. TextRank: Bringing Order into Tex. EMNLP, 2004, 4, 404-411.

31. Minaee, S., Kalchbrenner, N., Cambria, E. Deep Learning Based Text Classification: A Comprehensive Review [ArXiv preprint arXiv:2004.03705, 2020.

32. Mnih, V., Kavukcuoglu, K., Silver, D., et al. Human-Level Control Through Deep Reinforcement Learning. Nature, 2015, 518(7540), 529-533. https://doi.org/10.1038/nature14236

33. Nallapati, R., Zhai, F., Zhou, B. SummaRuNNer: A Recurrent Neural Network Based Sequence Model for Extractive Summarization of Documents. Thirty-First AAAI Conference on Artificial Intelligence.

34. Nallapati, R., Zhou, B., Gulcehre, C., et al. Abstractive Text Summarization Using Sequence-to-Sequence

Rnns and Beyond. ArXiv preprint arXiv:1602.06023, 2016. https://doi.org/10.18653/v1/K16-1028

35. Nenkova, A., McKeown, K. A Survey of Text Summarization Techniques. Mining Text Data, 2012, 43-76. https://doi.org/10.1007/978-1-4614-3223-4_3

36. Page, L., Brin, S., Motwani, R., et al. The PageRank Citation Ranking: Bringing Order to the Web. Stanford InfoLab, 1999.

37. Patel, R., Thakkar, A., Makwana, K. Comprehensive and Evolution Study Focusing on Comparative Analysis of Automatic Text Summarization. International Conference on Information and Communication Technology for Intelligent Systems, 2018, 383-389. https://doi.org/10.1007/978-3-319-63645-0_43

38. Ranzato, M. A., Chopra, S., Auli, M., et al. Sequence Level Training with Recurrent Neural Networks. ArXiv preprint arXiv:1511.06732, 2015.

39. Rennie, S. J., Marcheret, E., Mroueh, Y., et al. Self-Critical Sequence Training for Image Captioning. ArXiv preprint arXiv:1612.00563, 2016. https://doi.org/10.1109/CVPR.2017.131

40. Rush, A. M., Chopra, S., Weston, J. A Neural Attention Model for Abstractive Sentence Summarization. ArXiv preprint arXiv:1509.00685, 2015. https://doi.org/10.18653/v1/D15-1044

41. See, A., Liu, P. J., Manning, C. D. Get to the Point: Summarization with Pointer-Generator Networks. ArXiv preprint arXiv:1704.04368, 2017. https://doi.org/10.18653/v1/P17-1099

42. Silver, D., Sutton, R. S., Müller, M. Reinforcement Learning of Local Shape in the Game of Go. IJCAI, 2007, 7, 1053-1058.

43. Socher, R., Huang, E. H., Pennin, J, et al. Dynamic Pooling and Unfolding Recursive Autoencoders for Paraphrase Detection. Advances in Neural Information Processing Systems, 2011, 801-809.

44. Steinberger, J., Jezek, K. Using Latent Semantic Analysis in Text Summarization and Summarization Evaluation. Proceedings of ISIM'04, 2004, 93-100.

45. Sutskever, I., Vinyals, O., Le, Q. V. Sequence to sequence Learning with Neural Networks. Advances in Neural Information Processing Systems, 2014, 3104-3112.

46. Vanderwende, L., Suzuki, H., Brockett, C., et al. Beyond SumBasic: Task-focused Summarization with Sentence Simplification and Lexical Expansion. Information Processing and Management, 2007, 43(6), 1606-1618. https://doi.org/10.1016/j.ipm.2007.01.023

47. Vinyals, O., Fortunato, M., Jaitly, N. Pointer Networks. Advances in Neural Information Processing Systems, 2015, 2692-2700.

48. Williams, R. J. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. Machine Learning, 1992, 8(3-4), 229-256. https://doi.org/10.1007/BF00992696

49. Yin, W., Pei, Y. Optimizing Sentence Modeling and Selection for Document Summarization. IJCAI, 2015, 1383-1389.

50. Zhu, X., Penn, G. Comparing the Roles of Textual, Acoustic and Spoken-Language Features on Spontaneous-Conversation Summarization. Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers, Association for Computational Linguistics, 2006, 197-200. https://doi.org/10.3115/1614049.1614099

51. Zhu, X., Penn, G., Rudzicz, F. Summarizing Multiple Spoken Documents: Finding Evidence from Untranscribed Audio. Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2, Association for Computational Linguistics, 2009, 549-557. https://doi.org/10.3115/1690219.1690223