Mathematical Biosciences and Engineering

http://www.aimspress.com/journal/MBE

MBE, 20(1): 1488–1504.
DOI: 10.3934/mbe.2023067
Received: 09 July 2022
Revised: 22 September 2022
Accepted: 16 October 2022
Published: 31 October 2022

Research article

Extractive text summarization model based on advantage actor-critic and graph matrix methodology

Senqi Yang[1,2], Xuliang Duan[1,2,*], Xi Wang[1,2], Dezhao Tang[1,2], Zeyan Xiao[1,2] and Yan Guo[1,2]

[1] College of Information and Engineering, Sichuan Agricultural University, Ya'an, China
[2] The Lab of Agricultural Information Engineering, Sichuan Key Laboratory, Ya'an, China

* **Correspondence:** Email: duanxuliang@sicau.edu.cn.

Abstract: The automatic text summarization task faces great challenges. The main issue in the area is to identify the most informative segments in the input text. Establishing an effective evaluation mechanism has also been identified as a major challenge in the area. Currently, the mainstream solution is to use deep learning for training. However, a serious exposure bias in training prevents them from achieving better results. Therefore, this paper introduces an extractive text summarization model based on a graph matrix and advantage actor-critic (GA2C) method. The articles were pre-processed to generate a graph matrix. Based on the states provided by the graph matrix, the decision-making network made decisions and sent the results to the evaluation network for scoring. The evaluation network got the decision results of the decision-making network and then scored them. The decision-making network modified the probability of the action based on the scores of the evaluation network. Specifically, compared with the baseline reinforcement learning-based extractive summarization (Refresh) model, experimental results on the CNN/Daily Mail dataset showed that the GA2C model led on Rouge-1, Rouge-2 and Rouge-A by 0.70, 9.01 and 2.73, respectively. Moreover, we conducted multiple ablation experiments to verify the GA2C model from different perspectives. Different activation functions and evaluation networks were used in the GA2C model to obtain the best activation function and evaluation network. Two different reward functions (Set fixed reward value for accumulation (ADD), Rouge) and two different similarity matrices (cosine, Jaccard) were combined for the experiments.

Keywords: artificial intelligence; natural language processing; automatic summarization; deep reinforcement learning; extractive summarization model

1. Introduction

In natural language processing, automatic text summarization is widely used, e.g., to help users navigate web content and assist in building question-and-answer systems. Text summarization methods can be divided into two categories: extractive summarization and abstractive summarization. Extractive summarization models form summaries by selecting and copying text snippets from the document, while abstractive methods aim to generate concise summaries with paraphrasing. The commonly used abstractive methods include graph-based methods (e.g., graph-based lexical centrality as salience in text summarization (LexRank)) [1] and centrality-based methods (e.g., the centroid method) [2]. This work is primarily concerned with extractive summarization. Though abstractive summarization models have made strides in recent years, extractive techniques are still desirable, as they are simpler and more reliably yield semantically and grammatically correct sentences. Some authors have also made this point. For example, Li et al. used an abstractive method [3] and created the distributional semantics reward (DSR) model. The final experimental results showed that the summarization generated by DSR-facing long documents could not cover all of the critical information in the original text. See et al. used a hybrid pointer generator network and coverage for abstractive summaries [4]. Therefore, here, we used the extractive method to extract summarization.

The earliest extractive summarization method was introduced by Luhn in 1958. Luhn et al. used statistical information on words to calculate the importance of sentences [5] and selected high-scoring sentences to form the summarization. In 2000, Radev et al. built the multi-document summarization processor MEAD [6]. The model can reconstruct documents with summarization sentences and minimize the reconstruction error. Around 2004, extractive methods began to shift from "statistical methods" to "graph methods". Mihalcea et al. introduced the concept of bringing order into text with TextRank [7]. The core idea of TextRank is to determine the coverage of text units on the source document to get the score of sentence units. However, the algorithm does not consider the similarity between the sentences, resulting in information redundancy in the selected sentences.

In 2016, Ma et al. introduced an unsupervised multi-document summarization framework based on neural documents model (DocRebuild) to be able to refactor documents using summarization sentences [8] and selected summarization sentences to minimize refactoring errors. Cheng et al. proposed NN-SE and NN-WE [9], which are data-driven approaches based on neural networks and continuous sentence features. Nallapati et al. introduced a recurrent neural network-based sequence model for the extractive summarization of documents (SummaRuNNer) [10], and the model achieved state-of-the-art status within a year. Jadhav et al. built the Sentences and Words from Alternating Pointer Networks (SWAP-NET) model [11], which can establish the relationship between keywords and salient sentences. The model not only identified salient sentences and keywords, but it also combined them to form summaries. Zhou et al. have achieved neural extractive document summarization (NEUSUM) [12]. NEUSUM's integration of "selection" and "scoring" together solved the "sentence scoring" and "sentence selection" fragmentation in the extractive method. It achieved state-of-the-art results for the CNN/Daily Mail in that year. In 2020, Wang et al. created a model to dynamically update the weights of nodes by using a graph attention network iteratively [13] to extract the three highest-scoring sentences to form the summarization. Zhong et al. described the extractive summarization task as a semantic text matching task [14].

Although many scholars have recently achieved remarkable results in extracting summarization by using deep learning, it still has some limitations, particularly, the existence of exposure bias in neural network models [15]. Ranzato et al. introduced the exposure bias of neural networks [16] and pointed out that the cross-entropy is challenging to solve. He used reinforcement learning (RL) to improve exposure bias. Bahdanau et al. posted the exposure bias of neural networks [17] and alleviated this problem by using the actor-critic (AC) algorithm. Paulus et al. introduced the idea that models trained by supervised learning tend to exhibit exposure bias [18]. He used RL to circumvent this problem by focusing separately on the input and the continuously generated output. Narayan et al. pointed out that neural network training is based on cross-entropy [19], and that cross-entropy tends to produce redundant information when performing extractive summaries. Meanwhile, they suggested using RL to sequence sentences to overcome this difficulty. Mao et al. introduced the concept of multi-document summarization with maximal marginal relevance-guided RL [20] to address the redundancy of multi-document summarization in a unified manner.

To contribute to this field, we propose the graph matrix and advantage AC (GA2C) model to address the exposure bias of deep learning methods. GA2C translates "how to do extractive summarization" into "how to make action decisions" by uniting the graphs matrix with RL. Our contributions in this work could be summarized as follows: i) this is a novel application of advantage AC (A2C) on sentence selection for extractive summarization; ii) the analysis and empirical results show that GA2C alleviates the exposure bias problem; iii) the results disclosed that GA2C is better than previous models on CNN/Daily Mail data.

2. Related work

GA2C's graph matrix generator uses the idea of TextRank. And, the similarity between nodes is calculated using Jaccard similarity and cosine (COS) similarity. In addition, the TextCNN model is used in the implementation of the A2C algorithm.

2.1. TextRank

TextRank, proposed by Mihalcea et al. in 2004, is a graph-based keyword extractive method. In addition, this method refers to the idea of PageRank [21]. Indeed, TextRank is a widely used keyword extraction method [7]. The method splits the text into many keywords and uses the keywords to form nodes and the similarities between the nodes to form edges, thus forming a graph. The score is higher when a keyword is more similar to other keywords.

In general, the overall algorithmic process is divided into the following four steps.

i) The text is processed and added to the graph to form nodes;

ii) The similarity between nodes is calculated and added to the graph to form edges;

iii) The algorithm is iterated until convergence;

iv) The nodes are ranked based on their scores at final convergence.

The weight iteration algorithm for the node Vi is shown in Eq (1).

$$WS(Vi) = (1-d) + d * \sum_{Vj\in\in(Vi)} \frac{w\text{ji}}{\sum_{Vk\in Out(Vj)} w\text{jk}} WS(Vj) \tag{1}$$

where $WS(Vi)$ represents the weight of sentence i. Wji denotes the weight of the edge between the nodes Vj and Vi and d is the damping factor.

2.2. Jaccard similarity

Jaccard similarity is used to establish the relationship between sentences to form the similarity matrix. The algorithm is mainly used to calculate the similarity between samples of a symbolic measure or a Boolean measure [22]. The Jaccard similarity is concerned with the characteristics that are common between samples. When the characteristic attributes between samples are identified by symbols and Boolean values, the magnitude of the specific weight of the difference cannot be measured, as one can only obtain a result such as "whether it is the same". In two samples, the Jaccard is equal to the concurrent set minus the intersections sets, and then divided by the concurrent sets as shown in Eq (2).

$$J\ (S1, S2) = \frac{|S1 \cup S2| - |S1 \cap S2|}{|S1 \cup S2|} \tag{2}$$

where *S1* and *S2* denote the vector representation of the two sentences.

2.3. Cosine similarity

Cosine similarity is used to establish the relationship between sentences to form the similarity matrix. The algorithm measures the similarity between two vectors by calculating the cosine of the angle between them [23]. The value of cosine similarity is between 0 and 1. When two vectors have the same pointing, the cosine similarity is 1, which means that the similarity is 100%. When the angle of two vectors is 90°, the cosine similarity is 0, which implies that the similarity is 0. The cosine between two vectors can be found using the Euclidean dot product formula, as shown in Eq (3).

$$cos(S1, S2) = \frac{\sum_{i=1}^{n}(S1_i * S2_i)}{\sqrt{\sum_{i=1}^{n}(S1i)^2}\sqrt{\sum_{i=1}^{n}(S2i)^2}} \tag{3}$$

where *S1* and *S2* are the vector representations of the two sentences, and *n* is the dimension of the vector.

2.4. TextCNN

The evaluation network and decision-making network of GA2C were implemented in this study by using TextCNN. Both of them share the parameters of TextCNN. The summarization is selected by TextCNN. And, the selected summarization is evaluated by TextCNN. In 2014, Kim applied a convolutional neural network (CNN) model to the text classification tasks [24] and proposed TextCNN. The method is a variant of a CNN [25] and it aims to extract different local features by defining different filter convolutional kernel sizes. The model is mainly used in the field of text classification. Extractive summarization can be viewed as a text classification task divided into two categories.

As shown in Figure 1, the input was a 6 × 5 sentence matrix with a dimension of 5 for each word. The sentence matrix is passed through six convolutional kernels of size (3, 4, 5). The (3, 4, 5) convolutional kernels correspond to the extracted features size of (5, 4, 3). The extracted features are pooled and combined to get the representation after convolution.

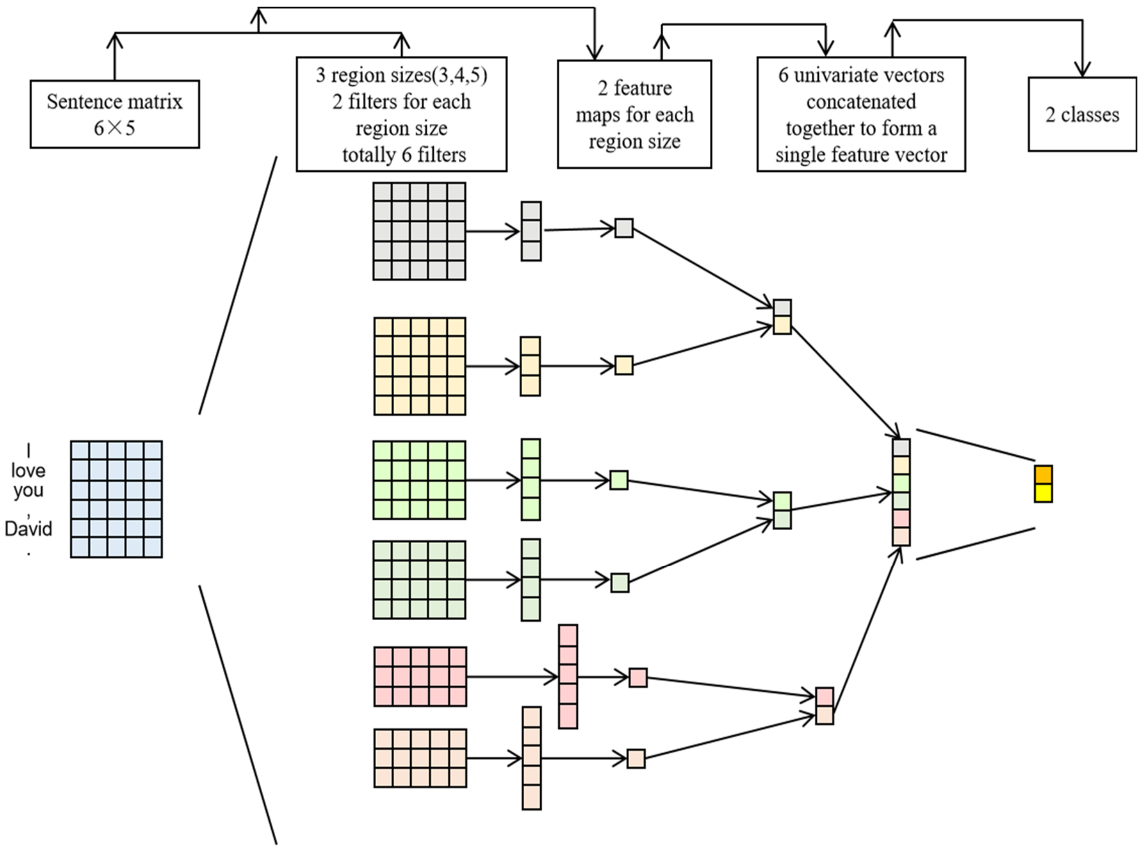


Figure 1. Illustration of TextCNN architecture.

The convolutional layer is the core of TextCNN. The three filters in this work have convolutional kernel sizes of 3, 4 and 5, which leads to the extraction of different features on the three levels. Its calculation formula is shown in Eq (4).

$$h_i = f\left(\sum_{x=1}^{3}\sum_{y=1}^{3} w_{i(x,y)} \times C_{(x,y)} + b_i\right) \tag{4}$$

f represents the activation function, commonly used as the rectified linear unit (ReLU). $w_{i(x,y)}$ denotes the weight of the i-th node in the output matrix corresponding to the filter input node (x, y). $C_{(x,y)}$ is the value of the node (x, y) in the filter. b_i denotes the bias term corresponding to the i-th node. Local feature extraction is achieved by setting three filters with convolutional kernels of (3, 4, 5). h_i is the result of the convolution layer.

3. Extractive summarization model based on graph matrix and A2C methodology

GA2C, which combines the A2C algorithm with graph structure composition, can reduce the impact of exposure bias. The methods used are mainly composed of A2C, TextRank and TextCNN models. The GA2C model mainly consists of a graph matrix generator, an evaluation network and a decision-making algorithm. The text is input to the graph matrix generator, after which the evaluation network and decision network are used for training. The framework structure diagram is shown in Figure 2.

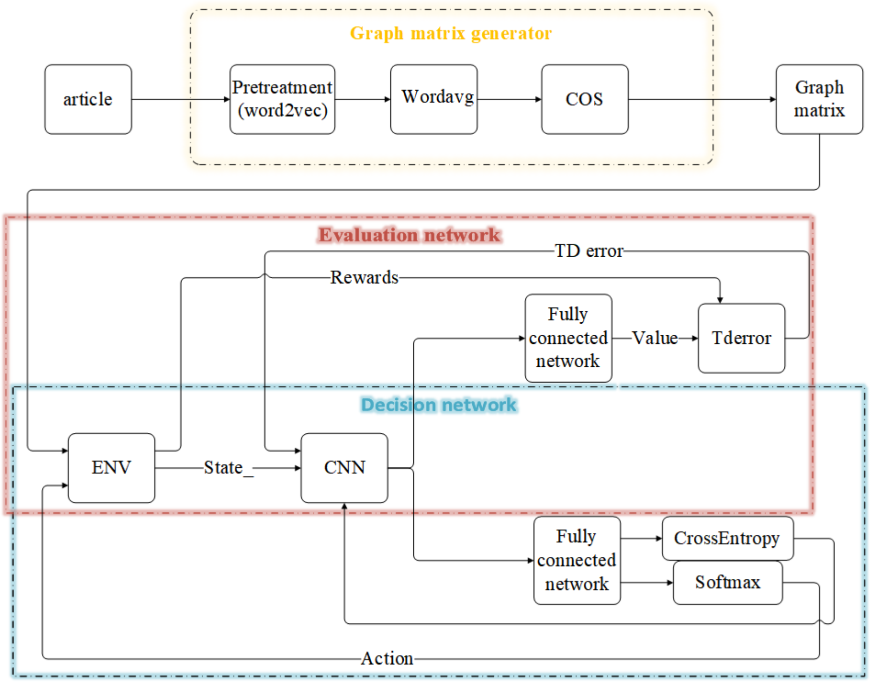


Figure 2. GA2C's overall framework diagram.

The articles are fed into the graph matrix generator to generate the graph matrix, and the evaluation network and the decision-making network use the graph matrix to get Rewards, the next state and the Action. The evaluation network evaluates the decision-making network based on the Action. The decision-making network then improves the decision process based on the evaluation of the evaluation network.

3.1. Graph matrix generator

TextRank provides a reference for the graph matrix generator. The process of using a graph matrix generator to generate a graph matrix is shown in Figure 3. The articles are input to the graph matrix generator, and after segmenting the articles into word vectors, WORDAVG is used to represent the sentence vectors and calculate the similarity between the sentence vectors to get the similarity matrix of the articles. The last graph is constructed using the similarity matrix and sentences node.

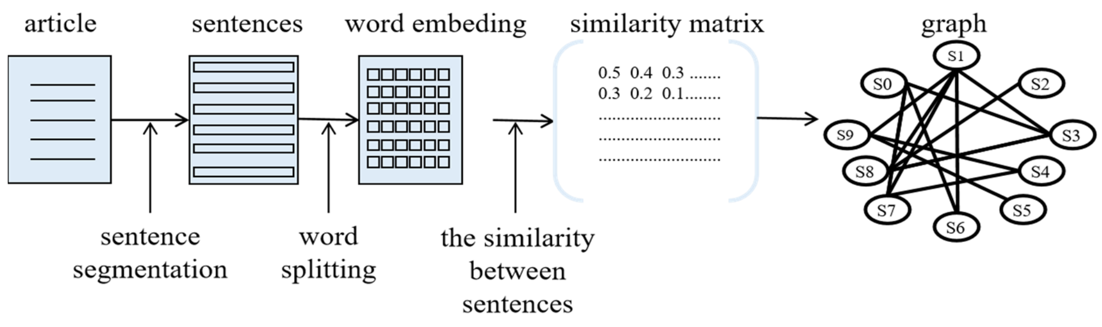


Figure 3. Graph matrix generation process.

3.2. Evaluation network

In this model, the A2C algorithm is used. The evaluation network corresponds to the critic of A2C. The network mainly calculates the actual values of the nodes. The optional actions are fed to the network and then provided to the hidden layer for full connectivity to obtain the value of the current state. The schematic diagram of the evaluation network is shown in Figure 4.

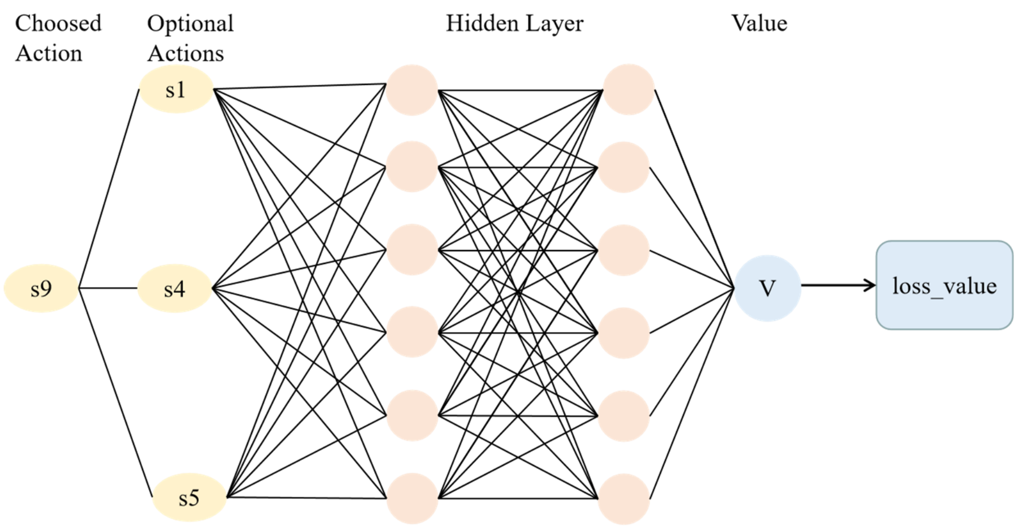


Figure 4. Evaluation network.

The figure starts from the node s9. Using the graph matrix, the action nodes connected to s9 can be derived as s1, s4 and s5. These three nodes are sent to the TextCNN and fully connected through the hidden layer, and the actual values are obtained. The actual values are calculated using the predicted values to obtain the loss of the evaluated network.

At each action step, the current state of the decision-making network is the set D of moves connected by the selected actions. The decision-making network determines the next sentence *di* from Set D. The evaluation network compares the selected *di* with the reference summarization *bi* to calculate the reward. In this paper, two reward calculation methods are provided.

1) ROUGE-1, ROUGE-2 and ROUGE-L as a reward:

$$Rewardi = \frac{\sum_{S\in\{label\}}\sum_{gram-1\in S} Countmatch\ (gram-1)}{\sum_{S\in\{label\}}\sum_{gram-1\in S} Count\ (gram-1)}$$

$$\frac{+\sum_{S\in\{label\}}\sum_{gram-2\in S} Countmatch\ (gram-2)}{\sum_{S\in\{label\}}\sum_{gram-2\in S} Count\ (gram-2)} + \frac{(1+b^2)\,R_{LCS}P_{LCS}}{R_{LCS}+b^2R_{LCS}} \tag{5}$$

which is equal to

$$Rewardi = R1 + R2 + Rl \tag{6}$$

The label is the standard summarization of this method. $Countmatch(gram-1)$ is the number of gram-1 matches. $Count(gram-1)$ is the total number of gram-1. ROUGE-2 is the same. $PLCS$ is the accuracy rate of longest common subsequence (LCS). $RLCS$ is the recall rate of LCS, and b=P_{LCS}/R_{LCS}.

2) Another method is to use simple accumulation for reward calculation. A positive reward is given for a correct decision, and there is no reward for an error.

$$Reward_i = \sum_0^k judge(di, bi) \tag{7}$$

where JUDGE is a reward calculation function that compares *di* with *bi*. If it is a summarization, JUDGE returns a positive reward. If it is not a summarization, there is no reward. Evaluate the network to get the total reward by discounting and adding up.

$$Reward_{all} = \sum_{i=0}^{k} \gamma Reward_i \tag{8}$$

where k is the number of times the selection action is executed. γ is the discount factor, and it is a constant.

In this method, we use the A2C algorithm. The advantage function uses TD-error.

$$TD - error_t = \left(r_t + V_{s(t+1)} - V_{st}\right) \tag{9}$$

This leads to the loss function of the evaluation network.

$$loss_{value} = -E \sum_{t=0}^{k} TD - error_t \tag{10}$$

3.3. Decision-making network

The decision-making network performs the selection of sentences. The sentences are fed to the decision network, and the probability of each current action is output after the hidden layer and SOFTMAX, and the action with the highest probability is selected as a result this time. The most basic TextCNN was used for action selection in this study to prove the framework's effectiveness.

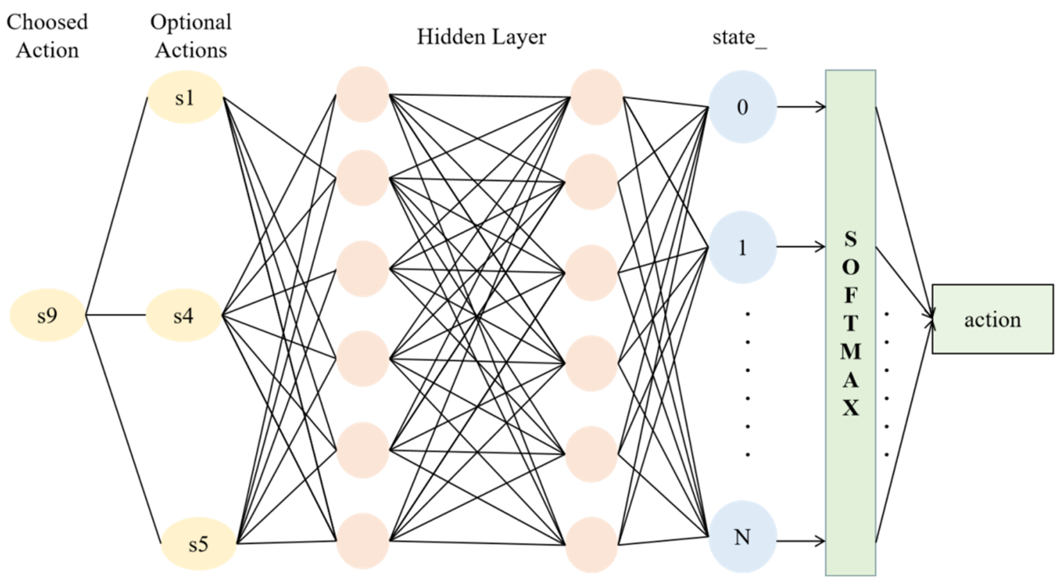


Figure 5. Decision-making network.

The figure starts from the node s9. Using the graph matrix, the action nodes connected to s9 can be derived as s1, s4 and s5. These three nodes are fed to the TextCNN and passed through the hidden layer. The probabilities of the three actions are obtained, and the results are selected after SOFTMAX.

The decision-making network is the policy function πθ (s, a). TD-error records the error between the actual and predicted values of the current action. The decision-making network can use TD-error to calculate the update gradient. The formula is shown in Eq (11).

$$\nabla_\theta G\ (\theta) = E_{\pi_\theta}[TD - \text{error}\nabla_\theta log\pi_\theta(a_t|s_t)] \tag{11}$$

where the update of θ is $\theta \leftarrow \theta + \alpha\nabla_\theta G\ (\theta)$.

The loss function of the decision-making network is $loss_{actor}$.

$$loss_{actor} = -E\sum_{t=0}^{k} log\pi\theta\ (at \vee st) \tag{12}$$

4. Results and discussion

4.1. Dataset

CNN and Daily Mail were used to evaluate GA2C [26]. This dataset contains news documents and their corresponding summarizations from CNN and Daily Mail websites. It contains 287,227 documents for training, 13,368 documents for validation and 11,490 documents for testing. We followed [13] to preprocess the dataset.

4.2. Evaluation metrics

There are more evaluation methods for text summarization tasks. In this study, we used recall-oriented understudy for gisting evaluation (ROUGE), which Lin and Och constructed in 2004 [26]. It is a frequently used evaluation metric in various text summarization models [27–29]. ROUGE is used to calculate the F1 scores of ROUGE-1, ROUGE-2 and ROUGE-L between the sentences selected by GA2C and the standard sentences for evaluation.

In the formula for ROUGE-N, N-gram refers to a contiguous substring of length N in the text. The denominator is the number of N-grams of all reference summaries. The numerator is the number of N-grams of both the model-predicted and the reference summarizations. In this study, *N* was taken as 1 and 2, which means that the evaluation methods are ROUGE-1 and ROUGE-2.

$$ROUGE - N = \frac{\sum_{S\in\{RS\}}\sum_{gram_n\in S} Count_{match}\ (gram_n)}{\sum_{S\in\{RS\}}\sum_{gram_n\in S} Count\ (gram_n)} \tag{13}$$

where *RS* denotes the reference summarization.

The formula for ROUGE-L is shown in Eqs (14)–(16).

$$Rlcs = \frac{LCS(X,Y)}{m} \tag{14}$$

$$Plcs = \frac{LCS(X,Y)}{n} \tag{15}$$

The LCS function is the length of the longest common substring of X and Y. *m* and *n* means the size of the reference summarization and the GA2C summarization, respectively. R_{lcs} and P_{lcs} respectively represent the recall and accuracy.

$$F_{lcs} = \frac{(1+\beta^2)R_{lcs}P_{lcs}}{R_{lcs}+\beta^2 P_{lcs}} \quad (16)$$

F_{lcs} indicates the F1 score for ROUGE-L. When $\beta = 1$, precision and recall are weighted equally. In some cases, if precision is more critical, β should be adjusted to less than 1. If recall is more important, β should be adjusted to be greater than 1. In this study, β was taken as 1.

4.3. Implementation details

For these experiments, the parameter settings were mainly divided into a graph matrix generator, an evaluation network and a decision-making network.

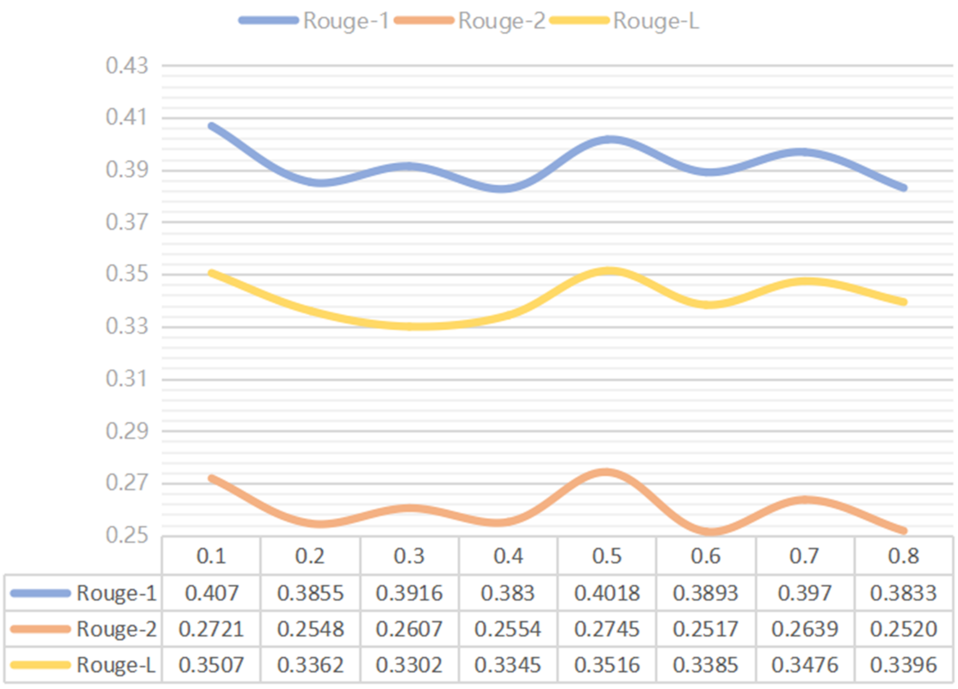
1) Graph matrix generator

The sentences were cropped, and the average length of the sentences was set to 22 after calculation. The similarity between sentences and sentences was set to 0.05 to have the correct action in each candidate action. When the similarity is more significant than 0.05, the connection is created in the graph.

2) Evaluation network

When fixed-value rewards are used, this paper sets the ADD reward value to 0.1.

In this study, eight sets of experiments were conducted using the values of fixed rewards, as shown in Figure 6.



	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
Rouge-1	0.407	0.3855	0.3916	0.383	0.4018	0.3893	0.397	0.3833
Rouge-2	0.2721	0.2548	0.2607	0.2554	0.2745	0.2517	0.2639	0.2520
Rouge-L	0.3507	0.3362	0.3302	0.3345	0.3516	0.3385	0.3476	0.3396

Figure 6. Experimenting with fixed rewards.

Figure 6 shows that, for the fixed-reward setting, the highest value is reached when the reward value is 0.1. The leads were 0.52 and 1.67 for ROUGE-1 and ROUGE-2, respectively.

3) Common parameters of the evaluation and decision-making networks

Where the discount factor of information entropy is 0.001, the discount factor of the reward was set to 0.99. The lexicon was 32341. The convolutional kernel size was (22,2). The learning rate was 0.01.

4.4. Update of decision-making network and evaluation network

GA2C's loss function consists of three parts, i.e., $loss_{value}$, $loss_{actor}$ and entropy.

$$entropy = \sum_{t=0}^{k} \left(log\pi\theta(at \vee st) \middle| \quad \middle| \frac{exp(ft)}{\sum_{c=1}^{C} exp(fc)} \right) \tag{17}$$

The total loss is $loss_{Total}$:

$$loss_{Total} = loss_{value} + \beta * loss_{actor} + \eta * entropy \tag{18}$$

where β and η are discount factors. The $loss_{Total}$ function used in this study is consistent with many A2C applications. For example, Li et al. used A2C to diversify recommendation results [30]. The method uses a loss calculation that also adds up the actor and critic parts to perform the update. In this study, the selection summary problem was considered as an action decision problem, which means that the intelligence is trained by putting them into a graph composed of sentences, somewhat like a path-planning problem. For this reason, we applied the loss function used in the A3C Super Mario implementation and added additional $entropy$ [31]. To demonstrate the effect of information entropy on the final effect, a set of comparative experiments was conducted. The values of ROUGE-1, ROUGE-2 and ROUGE-L for the case without $\eta * entropy$ were tested when the reward value was 0.1.

Table 1. Entropy impact on results.

Model	ROUGE-1	ROUGE-2	ROUGE-L
No added entropy	40.25	27.14	35.40
With entropy	**40.70**	**27.21**	**35.07**

The bold in the table represents the best result. As shown in Table 1, GA2C leads to 0.45 and 0.07 for Rouge-1 and Rouge-2, respectively, for the case of added $\eta * entropy$. This result proves that adding $\eta * entropy$ is more helpful for the convergence of training.

4.5. Experimental results

To verify the validity of the models in this paper, GA2C was compared with some typical methods on CNN/Daily Mail data, including Lead-3, BANDITSUM [15], Refresh [19], SummaRuNNer [10], Pointer-generator + coverage [32] and deep Q-network (DQN) [33]. Refresh, which combines maximum-likelihood cross-entropy loss with RL, was chosen as the baseline of this study. The model abstracted the single-document summarization task into a sentence ranking task, and global optimization ROUGE was used for extractive summarization. Meanwhile, several classical models were used as comparison models. SummaRuNNer is an extractive model based on a recurrent neural network. The model uses a novel training mechanism that allows the introduction of labels without the training phase. Lead-3 simply uses the first three sentences of the document as a summarization, but it achieved great results. Pointer-generator + coverage incorporates the seq2seq model, a pointer-generator network, a coverage mechanism, mitigating out-of-vocabulary words and information redundancy. The DQN is an RL algorithm based on the DQN method for extractive summarization. The method learns strategies to maximize the ROUGE score in Q-value approximation. BANDITSUM uses the optimized ROUGE score to treat the extractive summarization as a contextual bandit problem.

In addition, we conducted ablation experiments from many angles, including using four activation functions, using different evaluation network AC methods and uniting different reward functions with different mapping methods.

Table 2. GA2C with comparisons of typical models.

Model	ROUGE-1	ROUGE-2	ROUGE-L	ROUGE-AVG
SummaRuNNer	39.60	16.20	35.30	30.37
Pointer-generator + coverage	39.53	17.28	36.38	31.06
Lead-3	40.34	21.00	36.57	32.64
Refresh (baseline)	40.00	18.20	36.60	31.60
DQN	39.4	16.10	35.6	30.37
GA2C	**40.70**	**27.21**	35.07	**34.33**

The bold in the table represents the best result. GA2C was compared with several classical models, as shown in Table 2, with the best results indicated by bolded numbers. Compared to Refresh, GA2C led on ROUGE-1, ROUGE-2 and ROUGE-AVG by 0.70, 9.01 and 2.73, respectively. Compared with Refresh, GA2C utilized article-building graphs to better establish the connection between individual sentences and speed up the training. And, GA2C incorporates the A2C algorithm, which is superior to Refresh that only draws on the RL framework. GA2C not only avoids the problems caused by maximum likelihood similarity, it also minimizes the reliance on real tags. It can be seen that the GA2C performance achieved some improvement. This indicates that the critical sentence content extracted by the model in this study was more accurate and fluent.

Moreover, four activation functions were used to test the GA2C model. The default here was to use the cumulative reward method and COS calculation to build the graph. The four activation functions included ReLU [34], Leaky ReLU [35], parametric ReLU (PReLU) [36] and randomized ReLU (RReLU) [37]. The four activation functions are shown in Figure 7.

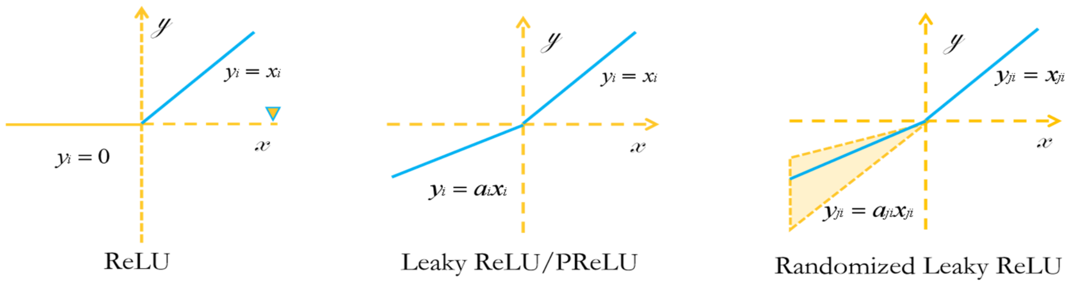


Figure 7. Comparisons of four activation functions.

ReLU: The ReLU activation function was a simple calculation. If the input was greater than 0, it directly returned the value provided as input; if the information was 0 or less, it returned the value 0.

Leaky ReLU: Leaky ReLU was used to assign a non-zero slope to all negative values, and the formula is shown in Eq (19).

$$yi = \{ \begin{matrix} x_i if x_i \geq 0 \\ \frac{x_i}{a_i} if x_i < 0 \end{matrix} \tag{19}$$

where a_i denotes a fixed parameter in a $(1, +\infty)$ interval.

PReLU: PReLU can be considered as a variant of Leaky ReLU. In PReLU, the slope of the negative part is based on the data and not predefined.

RReLU: RReLU is also a variant of Leaky ReLU. In RReLU, the slope of the negative value is obtained randomly in training and becomes fixed in subsequent tests.

$$yji = \{ \begin{matrix} x_{ji} if x_{ji} \geq 0 \\ a_{ji} x_{ji} if x_{ji} < 0 \end{matrix} \tag{20}$$

where a_{ji} is a randomly drawn value from a uniform distribution $U(I, u)$ and $I < u$; u takes values in the range of [0,1).

Table 3. Comparisons of four activation functions on GA2C.

Activation function	ROUGE-1	ROUGE-2	ROUGE-L	ROUGE-AVG
RELU	**40.46**	**26.85**	**34.77**	**34.02**
RRELU	40.30	26.71	34.60	33.87
PRELU	40.26	26.63	34.59	33.83
Leaky RELU	40.35	26.82	**34.77**	33.98

The bold in the table represents the best result. By comparison, RELU achieved the best results against the other four activation functions. Compared to Leaky RELU, GA2C led on ROUGE-1, ROUGE-2 and ROUGE-AVG by 0.70, 9.01 and 2.73, respectively. RELU was a simple way to achieve good results on GA2C. In this study, we finally decided to use the RELU activation function as the default function.

Meanwhile, to compare the different evaluation network algorithms (AC and A2C) [38], experiments were conducted in this study using AC combined with graph matrices. The experimental results are shown in Table 4.

Table 4. Using AC and A2C algorithms on GA2C.

Model	ROUGE-1	ROUGE-2	ROUGE-L	ROUGE-AVG
GAC	40.41	**26.86**	34.71	33.99
GA2C	**40.46**	26.85	**34.77**	**34.02**

The bold in the table represents the best result. The AC algorithm is slightly worse than the A2C algorithm in terms of the combined effect. Compared to AC, A2C led on ROUGE-1, ROUGE-L and Rouge-AVG by 0.70, 9.01 and 2.73, respectively. The AC algorithm has no baseline, and the feedback was positive, which can lead to some actions not converging. However, the A2C algorithm includes a baseline, making the feedback positive and negative. This made it easier to converge and train [39]. The A2C was more suitable for the model in this study.

In addition, to find the best pairing of a reward function and graph building method, we provided two ways of reward calculation: 1) cumulative way to calculate the reward of action, and 2) use of ROUGE to calculate the reward of action. And, two ways of text similarity when building graphs have been provided: (i) Jaccard similarity to calculate sentence similarity and (ii) cosine similarity to calculate sentence similarity. In this study, to try out the best combination, these two were combined separately for experiments.

Table 5. Comparisons of different reward and similarity algorithms.

Model	ROUGE-1	ROUGE-2	ROUGE-L	ROUGE-AVG
GA2C-ADD-COS	40.46	26.85	34.77	34.02
GA2C-ADD-JACCARD	34.86	20.49	29.03	33.88
GA2C-ROUGE-COS	**40.70**	**27.21**	**35.07**	**34.33**
GA2C-ROUGE-JACCARD	34.76	20.50	28.98	28.08

The bold in the table represents the best result. The GA2C model used ROUGE to calculate rewards and COS to calculate sentence similarity to build graphs, which can yield the best results. The combination of ROUGE and COS performed the best. Compared to ADD and COS, it led on ROUGE-1, ROUGE-2, ROUGE-L and ROUGE-AVG by 0.24, 0.36, 0.30 and 0.31, respectively. The Jaccard was not satisfactory in combination with ROUGE or ADD, which was 6% worse than the combination with COS. This is mainly because Jaccard compared the intersection and concurrence of two sentences, which does not reflect the similar relationships between texts well. The results of the COS experiments were all higher than those of the Jaccard experiments, where ADD was 0.24, 0.36, 0.40 and 0.31 lower than ROUGE for the four metrics when the COS constructed graphs were applied, indicating that ROUGE returned a more accurate reward.

5. Conclusions

In order to solve the problems of exposure bias and low precision in extractive summarization. This paper introduced the extractive summarization model GA2C. The algorithm transforms the data into a graph matrix and trains the model using an evaluation and a decision-making network. GA2C uses an RL network for feature extraction and achieves the selection of actions by using a decision-making network. The actions selected by the decision-making network are evaluated using the evaluation network and a reward is returned. GA2C utilizes RL to reduce exposure bias greatly. The experimental results showed that the summarization content extracted by GA2C was more accurate, and that the language was more fluent. Compared to Refresh, GA2C led on ROUGE-1, ROUGE-2 and Rouge-AVG by 0.70, 9.01 and 2.73, respectively.

The model in this paper could be further improved, as follows: i) the word vector representation can be trained using Bert, Word2vec, etc.; ii) several more types of connections between sentences can be tried, such as ROUGE or other methods of establishing relationships; and iii) GA2C can be applied to datasets in different domains, such as meeting minutes, emails, etc. In addition, GA2C training is time-consuming. The training time can be reduced in the future by learning from some techniques from graph CNN training. The reward design of GA2C can also be further optimized to improve the accuracy of summary extraction.

Conflict of interest

The authors declare that there are no conflicts of interest.

References

1. G. Erkan, D. R. Radev, Lexrank, Graph-based lexical centrality as salience in text summarization, *J. Artif. Intell. Res.*, **22** (2004), 457–479. https://doi.org/10.1613/jair.1523
2. D. R. Radev, H. Jing, M. Styś, D. Tam, Centroid-based summarization of multiple documents, *Inf. Process. Manage.*, **40** (2004), 919–938. https://doi.org/10.1016/j.ipm.2003.10.006
3. S. Li, D. Lei, P. Qin, W. Y. Wang, Deep reinforcement learning with distributional semantic rewards for abstractive summarization, preprint, arXiv:1909.00141. https://doi.org/10.48550/arXiv.1909.00141
4. A. See, P. J. Liu, C. D. Manning, Get to the point: summarization with pointer-generator networks, preprint, arXiv:1704.04368. https://doi.org/10.48550/arXiv.1704.04368
5. H. P. Luhn, The automatic creation of literature abstracts, *IBM J. Res. Dev.*, **2** (1958), 159–165. https://doi.org/10.1147/rd.22.0159
6. D. Radev, T. Allison, S. Blair-Goldensohn, J. Blitzer, Z. Zhang, MEAD—a platform for multidocument multilingual text summarization, in *4th International Conference on Language Resources and Evaluation*, (2004), 699–702.
7. R. Mihalcea, P. Tarau, E. Figa, PageRank on semantic networks, with application to word sense disambiguation, in *COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics*, (2004), 1126–1132.
8. S. Ma, Z. H. Deng, Y. Yang, An unsupervised multi-document summarization framework based on neural document model, in *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, (2016), 1514–1523.
9. J. Cheng, L. Dong, M. Lapata, Long short-term memory-networks for machine reading, preprint, arXiv:1601.06733. https://doi.org/10.48550/arXiv.1601.06733
10. R. Nallapati, F. Zhai, B. Zhou, Summarunner: A recurrent neural network based sequence model for extractive summarization of documents, in *Thirty-first AAAI Conference on Artificial Intelligence*, 2017.
11. A. Jadhav, V. Rajan, Extractive summarization with swap-net: Sentences and words from alternating pointer networks, in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, **1** (2018), 142–151, https://doi.org/10.18653/v1/P18-1014.
12. Q. Zhou, N. Yang, F. Wei, S. Huang, M. Zhou, T. Zhao, Neural document summarization by jointly learning to score and select sentences, preprint, arXiv:1807.02305. https://doi.org/10.48550/arXiv.1807.02305
13. D. Wang, P. Liu, Y. Zheng, X. Qiu, X. Huang, Heterogeneous graph neural networks for extractive document summarization, preprint, arXiv:2004.12393. https://doi.org/10.48550/arXiv.2004.12393
14. M. Zhong, P. Liu, Y. Chen, D. Wang, X. Qiu, X. Huang, Extractive summarization as text matching, preprint, arXiv:2004.08795. https://doi.org/10.48550/arXiv.2004.08795

15. Y. Dong, Z. Li, M. Rezagholizadeh, J. C. K. Cheung, EditNTS: An neural programmer-interpreter model for sentence simplification through explicit editing, preprint, arXiv:1906.08104. https://doi.org/10.48550/arXiv.1906.08104
16. M. A. Ranzato, S. Chopra, M. Auli, W. Zaremba, Sequence level training with recurrent neural networks, preprint, arXiv:1511.06732. https://doi.org/10.48550/arXiv.1511.06732
17. D. Bahdanau, P. Brakel, K. Xu, A. Goyal, R. Lowe, J. Pineau, et al., An actor-critic algorithm for sequence prediction, preprint, arXiv:1607.07086. https://doi.org/10.48550/arXiv.1607.07086
18. R. Paulus, C. Xiong, R. Socher, A deep reinforced model for abstractive summarization, preprint, arXiv:1705.04304. https://doi.org/10.48550/arXiv.1705.04304
19. S. Narayan, S. B. Cohen, M. Lapata, Ranking sentences for extractive summarization with reinforcement learning, preprint, arXiv:1802.08636. https://doi.org/10.48550/arXiv.1802.08636
20. Y. Mao, Y. Qu, Y. Xie, X. Ren, J. Han, Multi-document summarization with maximal marginal relevance-guided reinforcement learning, preprint, arXiv:2010.00117. https://doi.org/10.48550/arXiv.2010.00117
21. L. Page, S. Brin, R. Motwani, T. Winograd, *The Pagerank Citation Ranking: Bringing Order to the Web*, Technical Report, Stanford InfoLab, 1998.
22. P. Zhang, X. Huang, Y. Wang, C. Jiang, S. He, H. Wang, Semantic similarity computing model based on multi model fine-grained nonlinear fusion, *IEEE Access*, **9** (2021), 8433–8443. https://doi.org/10.1109/ACCESS.2021.3049378
23. G. Malik, M. Cevik, D. Parikh, A. Basar, Identifying the requirement conflicts in SRS documents using transformer-based sentence embeddings, preprint, arXiv:2206.13690. https://doi.org/10.48550/arXiv.2206.13690
24. Y. Kim, Convolutional neural networks for sentence classification, in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, (2014), 1746–1751. https://doi.org/10.3115/v1/D14-1181
25. Y. Zhang, B. Wallace, A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification, preprint, arXiv:1510.03820. https://doi.org/10.48550/arXiv.1510.03820
26. C. Y. Lin, F. Och, Looking for a few good metrics: ROUGE and its evaluation, in *Ntcir workshop*, 2004.
27. T. Ma, H. Wang, Y. Zhao, Y. Tian, N. Al-Nabhan, Topic-based automatic summarization algorithm for Chinese short text, *Math. Biosci. Eng.*, **17** (2020), 3582–3600. https://doi.org/10.3934/mbe.2020202
28. T. Zhang, I. C. Irsan, F. Thung, D. Han, D. Lo, L. Jiang, iTiger: An automatic issue title generation tool, preprint, arXiv:2206.10811. https://doi.org/10.48550/arXiv.2206.10811
29. A. Mullick, A. Nandy, M. N. Kapadnis, S. Patnaik, R. Raghav, R. Kar, An evaluation framework for legal document summarization, preprint, arXiv:2205.08478. https://doi.org/10.48550/arXiv.2205.08478
30. S. Li, Y. Yan, J. Ren, Y. Zhou, Y. Zhang, A sample-efficient actor-critic algorithm for recommendation diversification, *Chin. J. Electron.*, **29** (2020), 89–96. https://doi.org/10.1049/cje.2019.10.004
31. Project Webpage, Available from: https://github.com/vietnguyen91/Super-mario-bros-A3C-pytorch.

32. N. Xie, S. Li, H. Ren, Q. Zhai, Abstractive summarization improved by wordnet-based extractive sentences, in *CCF International Conference on Natural Language Processing and Chinese Computing*, Springer, Cham, (2018), 404–415. https://doi.org/10.1007/978-3-319-99495-6_34
33. K. Yao, L. Zhang, T. Luo, Y. Wu, Deep reinforcement learning for extractive document summarization, *Neurocomputing*, **284** (2018), 52–62. https://doi.org/10.1016/j.neucom.2018.01.020
34. J. Tong, Z. Wang, X. Rui, A multi-model-based deep learning framework for short text multiclass classification with the imbalanced and extremely small data set, *Comput. Math. Appl.*, **113** (2022), 34–44. https://doi.org/10.1016/j.camwa.2022.03.005
35. M. Liu, Z. Cai, J. Chen, Adaptive two-layer ReLU neural network: I. Best least-squares approximation, *Comput. Math. Appl.*, **113** (2021), 34–44. https://doi.org/10.1016/j.camwa.2022.03.005
36. A. Maniatopoulos, N. Mitianoudis, Learnable Leaky ReLU (LeLeLU): An alternative accuracy-optimized activation function, *Information*, **12** (2021), 513. https://doi.org/10.3390/info12120513
37. B. H. Nayef, S. N. H. S. Abdullah, R. Sulaiman, Z. A. A. Alyasseri, Applications, Optimized leaky ReLU for handwritten Arabic character recognition using convolution neural networks, *Multimedia Tools Appl.*, **81** (2022), 2065–2094. https://doi.org/10.1007/s11042-021-11593-6
38. S. K. Karn, N. Liu, H. Schuetze, O. J. Farri, Differentiable multi-agent actor-critic for multi-step radiology report summarization, preprint, arXiv:2203.08257. https://doi.org/10.48550/arXiv.2203.08257
39. Y. Guo, D. Z. Tang, W. Tang, S. Q. Yang, Q. C. Tang, Y. Feng, et al., Agricultural price prediction based on combined forecasting model under spatial-temporal influencing factors, *Sustainability*, **14** (2022). https://doi.org/10.3390/su141710483.