An Abstractive Summarizer Based on Improved Pointer-Generator Network

Wenbo Nie, Wei Zhang, Xinle Li and Yao Yu

May 30, 2019

An Abstractive Summarizer Based on Improved Pointer-Generator Network

1st Wenbo Nie
School of Automation and Electrical Engineering
University of Science and Technology Beijing
Beijing, China
969185900@qq.com

2nd Wei Zhang
School of Automation and Electrical Engineering
University of Science and Technology Beijing
Beijing, China
2033329616@qq.com

3rd Xinle Li
School of Automation and Electrical Engineering
University of Science and Technology Beijing
Beijing, China
xinle_li@yeah.net

4th Yao Yu*
School of Automation and Electrical Engineering
University of Science and Technology Beijing
Beijing, China
yuyao@ustb.edu.cn

***Abstract*—Aiming at the problems of insufficient semantic understanding, fluency and accuracy of abstracts in the field of neural abstractive summarization, an automatic text summarization model is proposed. First, we introduce the decoder attention mechanism in the reference network, which effectively improves the ability to understand words and generate vocabulary words. Second, the ability to extract words from the original text is improved by using the multi-hop attention mechanism, which improves the ability of the model to process out-of-vocabulary words. The experimental results on the CNN/Daily Mail dataset show that the model performs well on the standard evaluation system and improves the summary accuracy and sentence fluency.**

Keywords—natural language processing, abstractive summarization, recurrent neural network, attention mechanism

I. Introduction

Nowadays, with the rapid development of the Internet, the data is increasing at an exponential rate, which makes the problem of “information overload” appear. It is especially important for us to filter out useful information, and the effective way to solve this problem is to generate a text summary.

The method of generating the text summary mainly includes two types: extractive and abstractive. Extractive summarization mainly selects specific phrases, sentences and paragraphs from the original text into a summary; abstractive summarization reorganizes words to summarize the article according to the semantic information.

Extractive methods only need to find important sentences from the original text to compose the summary, but the consistency of the summary is difficult to guarantee. For example, if the pronoun is included in the sentence, it is hard to know what the pronoun refers to.

Abstractive methods are AI-based approaches that require the system to understand the meaning of the document and concisely summarize it in a readable human language. In recent years, with the development of neural networks and deep learning techniques, its advantages over traditional natural language processing methods are more obvious in terms of text representation, feature learning and text generation. The deep learning model has been widely applied and achieved amazing results in terms of natural language processing. With the deepening of deep learning techniques, especially the growing of the seq2seq and attention model, the abstractive summarization study has been improved a level, many neural abstractive summarization model has surpassed the best extractive summarization model on the DUC-2004 test dataset.

In order to improve the accuracy and fluency of the abstract, this paper introduces the decoder attention mechanism in the pointer-generator network. When generating summary word on the current timestep, the model can pay attention to the words generated at the previous moment. The multi-hop attention mechanism is introduced to improve the copy probability distribution. When copying words from the original text, the original text and the generated partial summary are considered. And it improves the ability to process the out-of-vocabulary (OOV) words. We introduce scheduled sampling proposed by Samy Bengio [1]. For the decoder stage in the sequence-to-sequence framework, Recurrent neural network will randomly use true label as the input on the next timestep, instead of using only the predicted output as before.

II. Related Work

Abstractive summarization based on neural network can reflect the semantic features of text well and has become a research hotspot. Traditional text summarization methods rely heavily on features. With the popularity of deep learning, especially the breakthrough of the seq2seq (sequence-to-sequence) and attention model in the field of machine translation, the text summarization task has also ushered in a new approach.

Abstractive text summarization mainly relies on the deep neural network architecture. The sequence-to-sequence model was proposed by Google [2] in 2014, which opened up the hot research of end-to-end network in the field of natural language processing. In the paper published in 2014 by Bahdanau et al. [3], the attention mechanism was first applied to natural language processing tasks. The attention mechanism is a resource allocation mechanism that always focuses on the content related to it, while other content is selectively ignored.

In 2015, Rush et al. [4] proposed an abstractive summarization method based on neural network, and combined with the attention mechanism to construct a summary generation model. The experimental results on the DUC-2004 and Gigaword datasets show that the accuracy for a single word can reach 31%. In 2016, Facebook and Harvard NLP group Chopra et al. proposed the RAS model based on CNN-RNN seq2seq architecture [5] which used a convolutional attention mechanism. Nallapati et al. of IBM

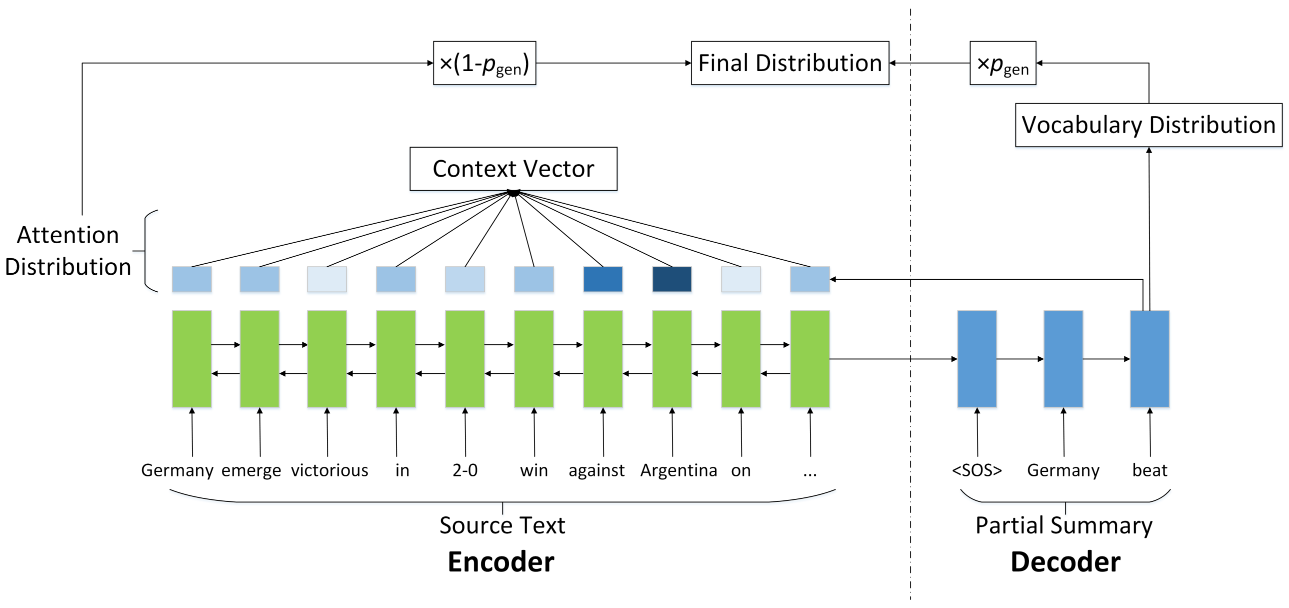

Fig. 1. Pointer-generator model

Watson Labs introduced techniques such as large vocabulary trick (LVT) to text summarization [6]. The author also proposed a new dataset CNN/Daily Mail, which is used to evaluate the task of multiple sentences and provides data protection for a large amount of related work in the future.

In 2016, Kikuchi et al. constructed four methods to control the length of the summary in the seq2seq architecture, and the experimental results did not have much loss. The innovation of the paper is to control the generated sentence length [7]. Zeng et al. proposed the read again encoder mechanism [8], which verified that their method was more effective than the technical level on the Gigaword and DUC datasets.

In the process of generating summary, OOV (out-of-vocabulary, such as a large number of names, place names, organization names, may not appear in the training corpus) problem will occur. Hong Kong University and Huawei Noah's Ark Lab proposed the COPYNET model in 2016 [9], which incorporated the copy mode into the seq2seq model, and mixed the traditional generation mode to build a new model. The model solved the OOV problem very well.

The main implementation method of the current abstractive text summarization is based on the seq2seq neural network model, but this method has problems such as incorrect information extraction and repetition of its own content. Stanford and Google Brain's paper [10] published in 2017 built a pointer-generator network that combined the seq2seq and attention model with the pointer network, and used the coverage mechanism to solve the problems of inaccurate information generated in the summary, the weak processing ability for vocabulary words and the high repetition rate.

IBM's Nema et al. proposed a generative model for query-based summarization in 2017 [11]. Based on seq2seq model, the attention mechanism was used on the query to obtain the context vector associated with the query. Tan et al. [12] of Peking University introduced a graph-based attention mechanism in the traditional encoder-decoder model to improve the model's ability to adapt to sentences. Celikyilmaz et al. [13] of Stanford University divided the understanding of long text into the understanding of many short texts. The author proposed to use multiple encoders to encode the paragraphs in the document one-to-one, for better understanding of the original long text. Li et al. [14] of Chinese University of Hong Kong proposed a neural network framework based on actor-critic method of reinforcement learning in 2018. During the training process, the model could be informed of the quality of the generated summary. It could greatly alleviate the problem of outputting meaningless content. Cao et al. [15] proposed a dual attention mechanism seq2seq framework that generated summaries based on source text and factual descriptions.

III. Our Models

In this section, we first introduce the baseline pointer-generator network model [10], then introduce our decoder attention mechanism, finally introduce the improvement of copy probability by using our multi-hop attention mechanism.

A. Baseline model

The baseline model uses the pointer-generator network proposed by Abigail See [10], as shown in Fig. 1. The pointer-generator network can copy words from the original text through the pointer mechanism, or generate new words from the vocabulary. The pointer-generator network is an improvement of the sequence-to-sequence attention model which includes the encoder, decoder and attention mechanism. The encoder is responsible for encoding the original text into a feature vector, and the decoder is responsible for generating the summary from the feature vector. The original words are sent into the encoder one by one, producing a series of encoder hidden states h_i. On each timestep t, the decoder accepts the word from the previous moment as input and generates the words of the summary one by one through decoder hidden state s_t. When the decoder generates the next word, the attention mechanism indicates which words in the original text are focused on. Attention distribution a^t is calculated as:

$$e_i^t = v^T \tanh\left(W_h h_i + W_s s_t + b_{attn}\right) \quad (1)$$

$$a^t = \mathrm{softmax}\left(e^t\right) \quad (2)$$

where v, W_h, W_s and b_{attn} are learnable parameters. Next, the context vector h_t^* is generated by the weighted sum of encoder hidden states:

$$h_t^* = \sum\nolimits_i a_i^t h_i \tag{3}$$

The context vector is the feature representation of original text on the current timestep t. The context vector h_t^* and the decoder hidden state s_t are spliced, and then the linear layers is used to generate the vocabulary probability distribution P_{vocab}:

$$P_{\text{vocab}} = \text{softmax}\left(V'\left(V[s_t, h_t^*] + b\right) + b'\right) \tag{4}$$

where V, V', b and b' are learnable parameters, The prediction of the next word bases on the vocabulary probability distribution P_{vocab}.

Due to the limitation of the size of the vocabulary, OOV problems will inevitably occur (a large number of names, place names and organizational names may not appear in the training corpus). To solve this problem, it is expected that model can copy some important words and fragments directly from the original text while maintaining the abstract generation ability. On the timestep t, the generation probability p_{gen} is generated by the context vector h_t^*, the decoder hidden state s_t and the decoder input x_t:

$$p_{\text{gen}} = \sigma\left(w_{h^*}^T h_t^* + w_s^T s_t + w_x^T x_t + b_{ptr}\right) \tag{5}$$

where w_{h^*}, w_s, w_x and b_{ptr} are learnable parameters, σ is the sigmoid function. p_{gen} is a soft switch that controls whether words are taken from the vocabulary probability distribution P_{vocab} or from the source text according to the attention distribution a^t. Finally, the final probability distribution of the words to be generated in the summary is:

$$P(w) = p_{\text{gen}} P_{\text{vocab}}(w) + \left(1 - p_{\text{gen}}\right) \sum\nolimits_{i:w_i = w} a_i^t \tag{6}$$

$P(w)$ is the final probability distribution of words. According to $P(w)$, the word to be generated can be selected by sampling or greedy search. During the training process, the loss of the t-th timestep is the negative likelihood log of the target word on that timestep:

$$\text{loss}_t = -\log P\left(w_t^*\right) \tag{7}$$

For sequence-to-sequence models, generating duplicate fragments is a constant problem, especially when generating multiple sentences. The pointer-generator network applies the coverage mechanism to solve this problem. First get a coverage vector c^t which is the sum of the attention values of the previous steps:

$$c^t = \sum\nolimits_{t'=0}^{t-1} a^{t'} \tag{8}$$

The coverage vector is used as a new input to compute the attention mechanism, and the attention mechanism (1) is rewritten as:

$$e_i^t = v^T \tanh\left(W_h h_i + W_s s_t + W_c c_i^t + b_{\text{attn}}\right) \tag{9}$$

where W_c and v are learnable parameters. The coverage vector c^t is the accumulated value of the attention distribution calculated on the past timesteps. It is equivalent to recording which parts of the original text the model has paid attention to on the past timesteps. When calculating attention distribution on the current step, the model does not know which part of the original text was previously focused, so it may pay attention to certain words repeatedly. Now when calculating the attention distribution, the coverage vector c^t tell the model which words it has focused on before, avoiding focusing on certain words. The model introduces coverage loss to penalize duplicate attention:

$$\text{covloss}_t = \sum\nolimits_i \min\left(a_i^t, c_i^t\right) \tag{10}$$

Coverage loss is bounded, $\text{covloss}_t \le \sum_i a_i^t = 1$. Finally, coverage loss pluses the original loss function (7) through a hyperparameter, generating a new loss function:

$$\text{loss}_t = -\log P\left(w_t^*\right) + \lambda \sum\nolimits_i \min\left(a_i^t, c_i^t\right) \tag{11}$$

The overall loss of the entire sequence is:

$$\text{loss} = \frac{1}{T} \sum\nolimits_{t=0}^{T} \text{loss}_t \tag{12}$$

B. *Decoder attention mechanism*

The attention mechanism of the pointer-generator network is for the encoder to calculate the attention weight by decoder hidden state s_t and a series of encoder hidden states h_i on the current timestep. The attention mechanism explains which words in the original text are focused on when the decoder generates the word of summary. We introduce the attention mechanism for the decoder. As shown in Fig. 2, when generating the summary word on the current timestep, the model should also pay attention to the previously generated summary, and calculate the weights of the words from the generated summary, which can make the summary more smooth and coherent. It also avoids generating duplicate content. For the current timestep t, the decoder hidden state is s_t, and the previous decoder hidden states are s_i, they are applied to compute the attention weight of the decoder. decoder attention weight r^t is calculated like this:

$$u_i^t = w^T \tanh\left(Q_i s_i + Q_s s_t + b_{dattn}\right) \tag{13}$$

$$r^t = \text{softmax}\left(u^t\right) \tag{14}$$

where w^T, Q_i, Q_s and b_{dattn} are learnable parameters, σ is the sigmoid function. The decoder summary vector s_t^* is then generated by the weighted sum of the decoder hidden states on previous timesteps:

$$s_t^* = \sum\nolimits_{i=1}^{t-1} r_i^t s_i \tag{15}$$

The summary vector s_t^* represents a feature representation of the summary that has been generated. Summary vector s_t^*

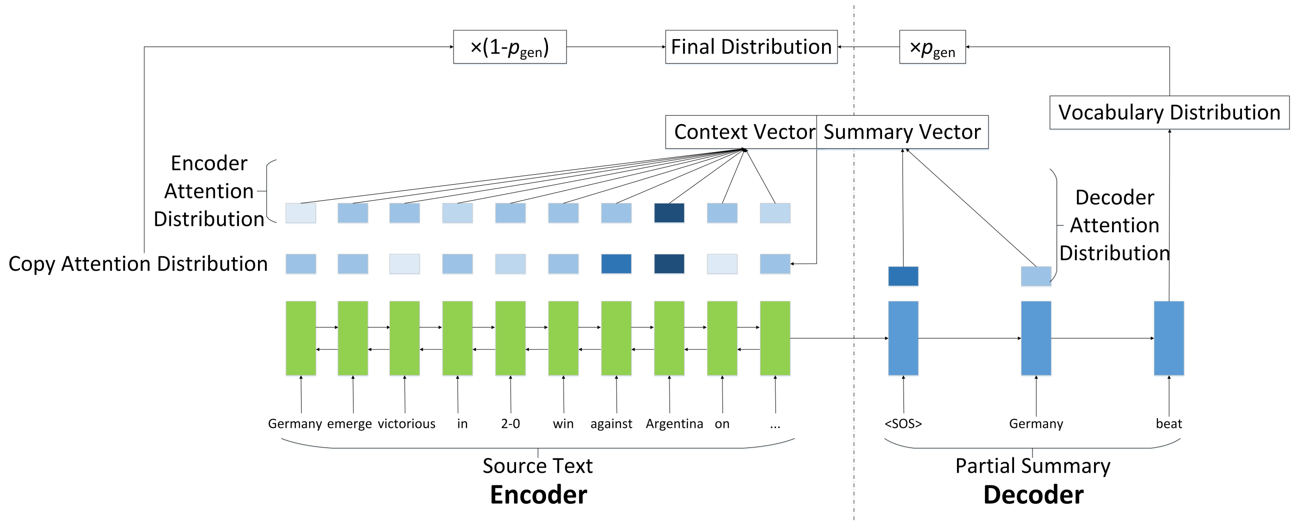

Fig. 2. Our model

is used for the decoder to generate the next word. Vocabulary probability distribution P_{vocab} is generated by linear layer after context vector h_t^*, decoder hidden state s_t and summary vector s_t^* splicing, changing (4) to:

$$P_{\text{vocab}} = \text{softmax}\left(V'\left(V\left[s_t, h_t^*, s_t^*\right]+b\right)+b'\right) \qquad (16)$$

where V, V', b and b' are learnable parameters, The probability distribution P_{vocab} represents the probability distribution of the words in the vocabulary. In this way, when the word of the summary is generating on the current timestep, not only the original text but also the generated partial summary are considered, and the generation ability of the model is improved. On the timestep t, the model produces the generation probability p_{gen} through the context vector h_t^*, the summary vector s_t^*, the decoder hidden state s_t and the decoder input x_t, changing (5) to:

$$p_{\text{gen}} = \sigma\left(w_{h^*}^T h_t^* + w_{s^*}^T s_t^* + w_s^T s_t + w_x^T x_t + b_{ptr}\right) \qquad (17)$$

where w_{h^*}, w_{s^*}, w_s and b_{ptr} are learnable parameters, σ is the sigmoid function. p_{gen} can control whether a word is taken from the vocabulary probability distribution P_{vocab} or copied from the input sequence.

C. *multi-hop attention improving copy probability*

Pointer-generator network combines the probability of generating words with the probability of copying words by the generation probability p_{gen}. The pointer-generator network directly uses the calculated attention distribution as the probability distribution of extracting words from the original text. Although this method reduces the model parameters, the global information of the original text and the generated summary are not used when extracting words from the original text. So the copy probability is not accurate enough.

We follow the process of constantly observing the original text and the generated summary when people do information extraction, and improve the copy probability based on the baseline pointer-generator network model, as shown in Fig. 2. First, the context vector h_t^* and the summary vector s_t^* are generated separately using the encoder attention and the decoder attention mechanism, which are the feature representations of the original text and the generated summary. Then apply the context vector h_t^* and the summary vector s_t^* to compute the attention distribution z^t with the original text:

$$d_i^t = v_m^T \tanh\left(V_i h_i + V_{h^*} h_t^* + V_{s^*} s_t^* + V_s s_t + V_c c_i^t + b_{mattn}\right) \quad (18)$$

$$z^t = \text{softmax}\left(d^t\right) \qquad (19)$$

where v_m, V_i, V_{h^*}, V_{s^*}, V_s, V_c and b_{mattn} are learnable parameters, s_t is the decoder hidden state , c^t is the coverage vector. The attention distribution z^t is used as copy probability distribution for extracting words from the original text.

Then, using the generation probability p_{gen} to combine the probability of generating the word with the probability of extracting the word from the original text, and the new probability distribution (including the OOV word) can be rewritten by (6):

$$P(w) = p_{\text{gen}} P_{\text{vocab}}(w) + \left(1 - p_{\text{gen}}\right)\sum_{i:w_i=w} z_i^t \qquad (20)$$

The next word can be generated using the extended vocabulary probability distribution $P(w)$, $P_{\text{vocab}}(w)$ represents the vocabulary probability distribution generated by the decoder, and $\sum_{i:w_i=w} z_i^t$ represents the copy probability distribution of extracting words from the original text.

IV. Experiment

A. *Experimental Setup*

We use the CNN/Daily Mail dataset in the experiment. Hermann et al. [16] proposed CNN/Daily Mail dataset to be used for reading comprehension task, and Nallapati et al. [6] specifically processed the dataset for summarization task. Unlike the previous Gigaword dataset [4], articles and summaries of the CNN/Daily Mail dataset are longer and the summaries are multi-sentence. The standard CNN/Daily Mail dataset includes 287,227 training examples, 13,368 validation examples, and 11,490 testing examples.

The encoder and decoder of our model use a single-layer GRU network. Our model has a 128-dimensional hidden states and 200-dimensional word embeddings. The article and summary vocabulary size is 50k, we use the Adagrad algorithm, the learning rate is 0.1, the initial adagrad accumulator is 0.1. The maximum gradient clipping used is 1 and the hyperparameter λ used is 1.

During training, the length of the original article was truncated, the length of the original article was limited to 400, and the length of the summary was 100. During testing, the length of the article was truncated to 400 and the length of the summary was truncated to 120.

In the pointer-generator network, while training, the decoder input on the current timestep is the word embedding of the word in the reference summary. At test time the input is the word embedding of the output word of the decoder on the previous timestep. The difference leads to a problem: when a wrong choice is made at a certain step at test time, a cumulative error may occur later.

To address this problem, we introduce the technique, scheduled sampling [1]. During training, the network will select the word in the reference summary (true sequence tag) with a probability ϵ_i, and select the output of the model itself by 1-ϵ_i. The value of ϵ_i is decreasing during the training process. At first, the network training is not enough. Then ϵ_i should be chosen a large value, that is, the model try to use the real mark. And the training process becomes more and more sufficient, ϵ_i should also decrease as the number of training iterations increases.

For the scheduled sampling, we use linear decay with an offset and slope of 1 and 0.00002. We select the sampling method to generate the next word from the extended vocabulary probability distribution.

We train the model on the NVIDIA GeForce RTX 2080Ti with batch size of 16. The beam search with a beam size of 4 is used for the test.

B. *Results and Analysis*

We use the standard ROUGE metric [17] to evaluate model, using the F1 values of ROUGE-1, ROUGE-2 and ROUGE-L. They respectively measure the word-overlap, bigram-overlap, and longest common sequence between the reference summary and the summary to be evaluated.

The ROUGE-1, ROUGE-2 and ROUGE-L scores of the our model are shown in TABLE I. It can be seen that the indicators have improved by comparing to the pointer-generator network. TABLE II is an example showing the output of the model. This example is taken from the testing set. It can be seen that our summary is readable and contains the main information of the article.

TABLE I. RESULTS FOR VARIOUS MODELS

Model	ROUGE-1	ROUGE-2	ROUGE-3
seq2seq +attention model	30.49	11.17	28.08
abstractive model [6]	35.46	13.30	32.65
pointer-generator [10]	39.53	17.28	36.38
our model	**39.62**	**17.68**	**36.79**

TABLE II. AN EXAMPLE

Article:
A stunning Italian political candidate who has posted dozens of pictures of herself in skimpy bikinis has denied using her looks to get votes. The relatively unknown Stefania La Greca was suddenly catapulted into the limelight after the sexy shots of the Lega Sud Ausoni party candidate went viral. But the 36-year-old, who is hoping to represent Caldoro in Campania, southern Italy after May's regional elections, defended her photographs. She told Italian TV show The Morning: 'I have not posted intimate photographs. The bikini? And what woman does not wear it?' 'We must go beyond appearance. The truth is that in this country women are still judged and only as a sexual object'. On her Facebook page, where she refers to her self as 'Divine', she has posted numerous pouting selfies and pictures of her in swimwear. Even Ms La Greca's election poster features a picture of her wearing a skimpy black dress and pulling a sultry pose. On her social media page she said the snaps were not an attempt to boost her election campaign - she had simply been born beautiful. Ms La Greca added that she was 'free to express herself' but complained about Italian attitudes towards women. She said: 'Mother Nature gave me the good fortune of being beautiful. 'But unfortunately, I am born in a context where woman are perceived as easy. 'I want to be different. I want to help my region. I am and want to be free to express myself.' She also defended the name 'Divine' saying it had been a nickname since she was a young girl and was now 'part of her.' Ms La Greca's sudden rise from obscurity to demand independence for her region echos the career of British politican Nicola Sturgeon. The Scottish National Party leader, recently shot up the popularity charts to be referred to as 'the only party leader with positive approval ratings' after a series of election debates. She is also known for her style transformation and her cheeky side, after it was recently revealed she has an oil painting of 'Naughty Nicola' hanging in her home. The photographs have split the opinions of voters with some criticizing the aspiring politician, while many more praised the sexy candidate. Writing on her Facebook page, local Domenico Vastarelli, said: 'You have all the skills to be a councillor.' And Davide Fabbri wrote: 'Magnificent!! You look good in a mini... Sexy legs. Perfect... Many congratulations!!' Ms La Greca, from the region of Campania in southern Italy, is standing for Lega Sud Ausonia - a small independent party that wants to see the region of Ausonia become independent. It is led by Gianfranco Vestuto who said the infamous bikini shot of the candidate was a 'few years old.' So far it has no representation in the Italian parliament, the European parliament or any regional or provincial assemblies but Ms La Greca is hoping to secure the first seat hen voters go the polls in the regional elections at the end of May.
Reference Summary: stefania la greca is standing for election for the lega sud ausonia party. she has posted dozens of selfies and pictures of herself in skimpy bikinis. but the stunning 36 - year - old denied she was using her looks to get votes.
Our Summary: stefania la greca , 36 , is standing for local elections for the lega sud ausonia in southern italy. ms la greca 's election poster features a picture of her wearing a skimpy black dress and pulling a sultry pose to the camera. the 36 - year - old has defended her pictures and denied she was using her looks to get votes.

V. CONCLUSION

In this work, we introduce the decoder attention mechanism and the multi-hop attention mechanism in the reference network, which effectively improves the understanding of the words and the generation ability of the model. While training, the introduction of scheduled sampling improves the model training quality. The experimental results on the CNN/Daily Mail dataset show that the model performs well in the standard evaluation system, improving the summary accuracy and sentence fluency.

ACKNOWLEDGMENT

This work was supported by National Natural Science Foundation of China under grants 61520106009, 61533008 and 61703037.

REFERENCES

[1] S. Bengio, O. Vinyals, N. Jaitly. Scheduled sampling for sequence prediction with recurrent neural networks[C]//Advances in Neural Information Processing Systems. 2015: 1171-1179.

[2] I. Sutskever, O. Vinyals, Q. V. Le. Sequence to sequence learning with neural networks[C]//Advances in neural information processing systems. 2014: 3104-3112.

[3] D. Bahdanau, K. H. Cho, Y. Bengio. Neural Machine Translation by Jointly Learning to Align and Translate[J]. 2017.

[4] A. M. Rush, S. Chopra, J. Weston. A Neural Attention Model for Abstractive Sentence Summarization[C]//Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. 2015: 379-389.

[5] S. Chopra, M. Auli, A. M. Rush. Abstractive sentence summarization with attentive recurrent neural networks[C]//Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. 2016: 93-98.

[6] R. Nallapati, B. Zhou, C. dos Santos. Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond[C]//Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning. 2016: 280-290.

[7] Y. Kikuchi, G. Neubig, R. Sasano. Controlling Output Length in Neural Encoder-Decoders[C]//Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing. 2016: 1328-1338.

[8] W. Zeng, W. Luo, S. Fidler. Efficient Summarization with Read-Again and Copy Mechanism[J]. 2016.

[9] J. Gu, Z. Lu, H. Li. Incorporating Copying Mechanism in Sequence-to-Sequence Learning[C]//Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2016, 1: 1631-1640.

[10] A. See, P. J. Liu, C. D. Manning. Get To The Point: Summarization with Pointer-Generator Networks[C]//Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2017: 1073-1083.

[11] P. Nema, M. M. Khapra, A. Laha. Diversity driven attention model for query-based abstractive summarization[C]//Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2017: 1063-1072.

[12] J. Tan, X. Wan, J. Xiao. Abstractive document summarization with a graph-based attentional neural model[C]//Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2017, 1: 1171-1181.

[13] A. Celikyilmaz, A. Bosselut, X. He. Deep Communicating Agents for Abstractive Summarization[C]//Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers). 2018: 1662-1675.

[14] P. Li, L. Bing, W. Lam. Actor-Critic based Training Framework for Abstractive Summarization[J]. 2018.

[15] Z. Cao, F. Wei, W. Li. Faithful to the Original: Fact Aware Neural Abstractive Summarization[J]. 2018.

[16] K. M. Hermann, T. Kočiský, E. Grefenstette. Teaching machines to read and comprehend[J]. Advances in Neural Information Processing Systems, 2015, 28.

[17] C. Y. Lin. Looking for a few good metrics: Automatic summarization evaluation-how many samples are enough?[C]//NTCIR. 2004.