Two-Stage Encoder for Pointer-Generator Network with Pretrained Embeddings

Zhou Lin[1,2,3], Qifeng Zhou✉[1,2,3], Langcai Cao[1,3]
[1]Automation Department, Xiamen University, Xiamen, China
[2]Shenzhen Research Institute of Xiamen University, Shenzhen, China
[3]Xiamen Key Laboratory of Big Data Intelligent Analysis and Decision, Xiamen University, Xiamen, China
zhouqf@xmu.edu.cn

2021 16th International Conference on Computer Science & Education (ICCSE) | 978-1-6654-1468-5/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICCSE51940.2021.9569678

***Abstract*—Abstractive summarization has become the mainstream of automatic text summarization because of its unique flexibility. Pointer-Generator Network(PGN) has already been the de facto standard for the text summarization model. However, this model still has two main shortcomings: first, since the word embeddings obtained from the word embedding matrix lack of contextual semantic information, the model tends to ignore the global information of the input text when generating the summary. Second, the embedding matrix is only learned on a single corpus from scratch, which decreases the convergence speed of the PGN model and affects the generalization ability of the word embeddings. In this paper, we propose a new text summarization framework, Two-Stage Encoder for pointer-generator network with Pretrained Embeddings, which aims to solve the problems mentioned above. We introduce an encoder based on the multi-head self-attention for the first-stage encoding. This encoder module can improve the word representation by solving the polysemy problem in the PGN model and alleviate the problem of ignoring global information. Besides, we apply the pretrained word embeddings GloVe to initialize the word embedding matrix and then fine-tune it. A range of experiments are conducted on the CNN/Daily Mail dataset. For the CNN/Daily mail dataset, comparing with the baseline model, the proposed model enhances the ROUGE metrics by 0.75, 0.74 and 0.77 respectively.**

***Index Terms*—Abstractive summarization, Pointer-generator network, Pretrained word embeddings, Multi-head self-attention**

I. Introduction

Text summarization aims at extracting the core content from a long text. A concise abstract may save the reader's time and help them withdraw the necessary information efficiently. There are two mainstream methods for obtaining text summaries: abstractive methods and extractive methods. The most common extractive methods take sentences as the smallest unit. This method can be regarded as a problem of binary classification for sentences. Since the main theme of texts is distributed in a few sentences, the summary obtained by the extraction method has inherent advantages and obtain a higher ROUGE metrics [1]. However, if the text is long and there are no exact subject sentences, a more complex system needs to be constructed at this time. This system can simulate the way that humans generate summaries: reading the article through and paraphrasing it. This is what we call abstractive methods.

The abstractive method is more difficult, therefore the extractive method is mainly used in the early stage of text summarization. After the seq2seq [2] model being proposed, abstractive methods begin to be widely used. Some early work [3] [4] [5] employ different feature extractors to build the seq2seq model based on encoder-decoder framework [6]. Nevertheless, the above methods generate too many unknown words. Besides, there exists the repetition problem at the word and sentence level in the summary. See et al. [7] introduce a hybrid PGN model, which reduces the appearance of unknown words. In addition, this paper also draws on the coverage mechanism in machine translation to alleviate the repetitive problem. As Word2vec [8], GloVe [9], FastText [10], ELMo [11], BERT [12] and other pretrained models have been proposed, Anh et al. [13] adopt pretrained word embeddings FastText to directly query the vocabulary in FastText to get word embeddings, alleviating the problem of unknown words and low-frequency words. However, FastText can not get the word embedding based on the context. In other words, the same word corresponds to the same meaning in different contexts. This is what we called the polysemy problem. Mastronardo et al. [14] employ a feature-based pretrained model ELMo to solve this problem. The ELMo model stack bidirectional LSTM to get deeper word embeddings with contextual semantic information. However, when it is used in a specific downstream task, there are few parameters that can be fine-tuned. Therefore it can not well represent words in a specific task.

Although these works are very promising, there are still some disadvantages in obtaining the context-based word representation in the PGN model so that the generated summary will lose some global information of the original text. In addition, due to the training of the embedding matrix carried out on a single corpus from scratch, which decreases the convergence speed of the PGN model. In this paper, we propose a new text summarization model named TSEPE to solve the above problems. There are two main contributions:

a. To solve the problem that the global information in the text is easy to be ignored in the PGN model, we propose an encoder module based on multi-head self-attention to obtain the contextualized word embeddings.

b. To speed up the convergence of the PGN model and

improve its' generalization performance, in this paper, we use the pretrained word embeddings GloVe to initialize the word embedding matrix. Then we can obtain the initial representation of words and fine-tune it as the training process of the text summarization task.

The remainder of this paper is organized as follows. In section II, we introduce the related work of our research. In section III we present the proposed TSEPE model. Section IV is the experimental setups and the experimental result analysis. And section V shows the conclusion and future work.

II. Related works

Filtering redundant information from a long text has become a hot research issue in the field of NLP. In recent years, with the development of the seq2seq model and pretrained technology, there are more and more models in abstractive summarization task combining the seq2seq model with the pretrained technology. Because the pre-training and fine-tuning method can better learn word representation in the text summarization task. In this section we mainly introduce the current abstractive method of summarization based on the PGN model and present the development of pretrained word embeddings in recent years.

A. Text summarization

Text summarization methods include extractive methods and abstractive methods. The extractive methods can be regarded as the classification of the original text at the sentence or word level. The extractive method has a high ROUGE metric due to its natural characteristics, but its readability is poor. Therefore, after the seq2seq model is proposed, abstractive methods become the mainstream. Machine translation and abstractive summarization tasks have great similarities. Many researchers are inspired by machine translation tasks. Rush et al. [3] first adopt the encoder-decoder architecture for text summarization tasks. They utilize a local attention based module as encoder and employ a neural network language model with beam search algorithm as decoder. Chopra et al. [4] propose a model which can be regarded as an extension of Rush et al. [3]. They use a convolution module to get the contextual representation of words. Besides The RNNs with attention mechanism are used as decoder. Nallapati et al. [5] replace all the feature extractors of the encoder and decoder with RNNs which can better handle sequence problems. What's more, they utilize switching decoder/pointer architecture to handle the out-of-vocabulary. See et al. [7] apply pointer-generator network and coverage mechanism to the seq2seq model based on LSTM. In this work, the pointer-generator network is different from that of Nallapati et al. [5] The pointer-generator network mix the probability from the copy and vocabulary distribution to get the final distribution. It can reduce unknown words in the output text and extract proper nouns in the source article. The coverage mechanism alleviates the occurrence of repeated words and sentences by adding penalty to the loss function. However, on account of vocabulary size, in the previous work, the word does not appear in the vocabulary is represented by the embedding of 'UNK' token. This makes an inaccurate embedding representation hence loss information for this word. To handle this problem, Anh et al. [13] utilize a pretrained model FastText with a 3-million-word vocabulary to get the word embeddings. There still exists polysemy problem(the same word has different meanings in different context) in the PGN model. To solve this problem, Mastronardo et al. [14] employ ELMo to obtain word embeddings with contextual semantic information. Li et al. [15] consider the previously predicted words when generating new words to avoid generating a summary of redundant repetition words.

B. Pretrained word embeddings

Pretrained Word Embeddings are the embeddings learned in one task which are used for solving another similar task. These embeddings are trained on large datasets, saved, and then used for solving other tasks. Qiu et al. [16] provide a comprehensive review of the pretrained word embedding technology. With word2vec [8], GloVe [9], FastText [10] and other pretrained models proposed, how to get the better word representation becomes a hot issue. The above methods lack of contextualized information. Peters et al. [11] propose the ELMo model, stacking bidirectional LSTM to get deeper word embeddings with contextual semantic information. The above methods aims to enhance on the semantic and the syntactic aspects of a text, remaining oblivious to sentimental information. Vaswani et al. [17]propose a new seq2seq model based on multi-head self-attention. The multi-head self-attention module can well get the word representation in the context better than LSTMs. But learning word embeddings from scratch is a challenging problem due to the sparsity of training data and large number of trainable parameters. Devlin et al. [12] draw on the encoder module in Transformer(Vaswani et al. [17]). They propose a pretrained model BERT which learn parameters from a large corpus and fine-tune in a specific task.

III. Method

The proposed model framework TSEPE is shown in Fig. 1. In this section, we first introduce how to initialize the word embedding matrix in our model. Next, we describe the first-stage encoder for extracting the context semantic information of words. And finally, we present the PGN model.

A. Word embedding matrix initialization

The initialization of parameters is very important in deep learning. In learning procession, a good initialization for parameters can accelerate convergence and alleviate the problems of gradient disappearance and explosion. In the PGN model, the parameters of the word embedding matrix are randomly initialized. They work well in some time, but most of the time they perform poorly. In the TSEPE model, in order to speed up the convergence of the PGN model, the parameters of the word embedding matrix are initialized with the pretrained word embeddings GloVe. Then we fine-tune the parameters of the network and make the model converge to a better solution.

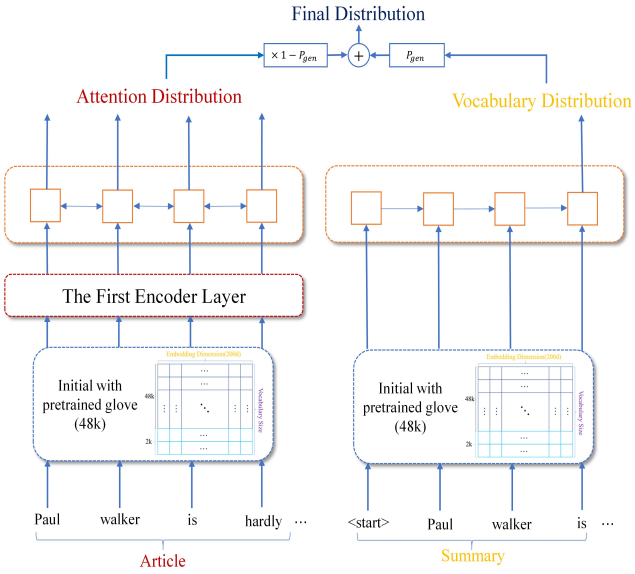

Fig. 1. The architecture of the TSEPE model

The same as the PGN model, we also define the word embedding matrix E to map the words in the dictionary into word embeddings. In the PGN model, See et al. do not pretrain the word embeddings matrix, it is learned from scratch during training. Unlike them, we do not train the matrix from scratch. 95% of words can be found in the model Glove pretrained on Wikipedia 2014 and Gigaword 5, so we define this part of E as E_1. After initialization, E_1 is changed into E_{GloVe}. The other 5% of E responding to the word embeddings that can not found from the pretrain model is defined as E_2. The results show that the uniform distribution is the best initialization option. The two word-embedding matrices are concatenated to obtain the final word embedding matrix E and then the parameters are fine-turned during training.

$$E_{GloVe}, E_{XU} = GloVe_{init}\left(E_1\right), XU_{init}\left(E_2\right) \tag{1}$$

$$E = Concat\left(E_{GloVe}, E_{XU}\right) \tag{2}$$

where $E_{GloVe} \in R^{v_1 \times d_{emb}}$, $E_{XU} \in R^{v_2 \times d_{emb}}$, $E \in R^{v \times d_{emb}}$. v_1 represents the vocabulary size of words which can be found in the pretrained word embeddings. v_2 represents the vocabulary size of words that randomly initialized, and v is the vocabulary size of our model.

$$x_{emb} = E(w) \tag{3}$$

where $E(.)$ represents a function that mapping tokens into the embedding through the word embedding matrix. Then the rough word representation can be obtained.

B. The first-stage encoder

Since the word representation in the PGN model lacks contextual semantic information, we propose a module based on self-attention [17] to deal with it. The architecture of this module is shown in Fig. 2. In this paper, we call it the first-stage encoder. The first-stage encoder consists of five parts, positional encoding, multi-head self-attention, residual connection, layer normalization, and feed forward neural network.

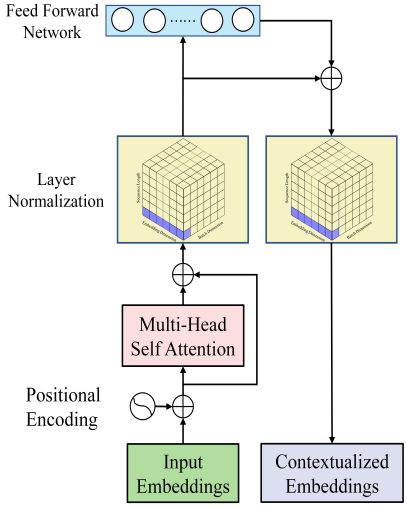

Fig. 2. The architecture of the first stage encoder

The word embeddings obtain by the word embedding matrix contains the semantic information of words, but the order of the word is not considered. In the TSEPE model, we use the fixed the position encoding with sine and cosine function first proposed by [17].

Since the multi-head self-attention is an extension of scaled dot-product attention, we introduce the scaled dot-product attention at first. The input consists three vectors of queries, keys, and values with the dimension of d_q, d_k, d_v. They are obtained by mapping the word embedding of the token through W^Q,W^K,W^V matrices respectively. We do the dot product of each query with all keys, divide the result by the scaling

factor $\sqrt{d_k}$, and then utilize the Softmax function to obtain the weights on each value. In the actual implementation, this calculation is done in matrix form for fast processing. We pack all the embeddings of tokens into matrix X. Then we get Q,K,V through W^Q,W^K,W^V respectively.

$$Q, K, V = XW^Q, XW^K, XW^V \tag{4}$$

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{5}$$

Multi-head self-attention extends the model's ability to focus on different positions and gives the attention layer multiple representation subspaces. In the multi-head self-attention, we have not only one, but multiple sets of W_i^Q,W_i^K,W_i^V weight matrices. Then after training, each set is used to project the input embeddings into different representation subspaces.

For each head, we do the same operation as scaled dot-product attention. Then we connect all the output heads Z_i and project them into Z through the weight matrix of W^o. It is worth noting that, we keep the input X and output Z in the same dimension.

$$Z = Concat\left(Z_1, \ldots, Z_h\right)W^O \tag{6}$$

$$Z_i = Attention\left(XW_i^Q, XW_i^K, XW_i^V\right) \tag{7}$$

where $W_i^Q \in R^{d_{model}\times d_k}$,$W_i^K \in R^{d_{model}\times d_k}$,$W_i^V \in R^{d_{model}\times d_v}$,$W_i^O \in R^{d_{model}\times hd_v}$.

C. Pointer-generator network with coverage mechanism

In this part, we introduce two core modules of the TSEPE model. One is the pointer-generator network, the other is the coverage mechanism.

a) Pointer-generator network: Pointer network [18] uses attention as a pointer to select a member of the input sequence as the output. See et al. [7] propose a hybrid network named pointer-generator network. It can not only copy words from the original article but also generate words from a fixed vocabulary. We define the generating probability as $p_{gen} \in [0, 1]$. When decoding at timestep t, the calculation formula of generating probability is

$$p_{gen} = \sigma\left(w_c^T c_t + w_s^T s_t + w_y^T y_t + b_{ptr}\right) \tag{8}$$

y_t is decoder input of timestep t and p_{gen} is a soft switch. It presents the probability to generate words. When decoding at timestep t, the probability distribution of the next word is the comprehensive probability distribution. It combines the generating probability of vocabulary words with the attention probability of the original article.

$$P(w) = p_{gen}P_v(w) + (1 - p_{gen})\sum_{i:w_i=w} a_i^t \tag{9}$$

b) Coverage mechanism: Word repetition is a common problem in text summarization tasks, especially when generating multi-sentence summaries. We adopt the coverage mechanism proposed by Tu et al. [19] to solve the repetition problem in the seq2seq model. The coverage vector is defined as the sum of attention distributions over all previous timesteps.

$$cr^t = \sum_{t'=0}^{t-1} a^{t'} \tag{10}$$

In the TSEPE model, the coverage vector is also used as the input to calculate the attention distribution, which can be seen as prior information of the current decision. Therefore we can effectively prevent from paying attention to the same position repeatedly which can avoid generating duplicate text.

$$e_i^t = v^T \tanh\left(W_h h_i + W_s s_t + w_c cr_i^t + b_{attn}\right) \tag{11}$$

In addition, we also introduce coverage loss to our model. It can punish the model for repeating attention to reduplicative positions:

$$covloss_t = \sum_i \min\left(a_i^t, cr_i^t\right) \tag{12}$$

The loss function consists of two parts, the main cross entropy loss and the coverage loss. The weight of the coverage loss is determined by μ:

$$loss_t = -\log P\left(w_t^*\right) + \mu \times covloss_t \tag{13}$$

IV. Experiments

In this section, we introduce the datasets used in the following experiments and then give a description of the implementation details.

A. Datasets

To verify the effectiveness of our model, we evaluate it on the CNN/Daily Mail dataset.

To ensure consistency with the baseline model, we perform the data preprocessing like See et al. [7]. The non-anonymized version obtains 281,173 training instances, 13368 validation instances, and 11490 test instances.

B. Evaluation Metrics

We take the common ROUGE metric to evaluate the quality of the models. ROUGE stands for Recall-Oriented Understudy for Gisting Evaluation and is commonly used metric for text summarization. ROUGE-N reflects the informativeness contained in the generated summary and ROUGE-L assess the fluency of the generated summary. When calculating the ROUGE metric, the pyrouge package is used.

C. Implementation Details

We utilize $Pytorch$ to build the experiment platform. Our model can hand unknown words, therefore we set a smaller size of the source vocabulary and target vocabulary as 50k(words that appear most frequently in the corpus). Our model has 256-dimensional hidden states and 200-dimensional word embeddings. For the first-stage encoder, the number of

TABLE I
THE INFLUENCE OF DIFFERENT MODULES ON MODEL ACCURACY

Method	ROUGE-1	ROUGE-2	ROUGE-L
PGN+cov(See et al.,2017)	39.53	17.28	36.38
PGN+cov(ours)	39.54	17.36	36.40
w/ pretrained embeddings	39.73	17.61	36.73
w/ two-stage encoder	40.10	17.87	36.99
w/both	**40.28**	**18.02**	**37.15**

the first-encoder layer and heads are 2 and 8. We set d_{model}, d_{ff}, d_q, d_k, d_v to 200, 1024, 200, 200, 200 respectively. Unlike See et al. [7], when training the word embedding matrix, we don't start training from scratch. We initialize the word embedding matrix by loading the pretrained word embeddings GloVe (about 95% parameters of the embedding matrix), the remaining 5% of the parameters are initialized randomly. And then we fine-turn them with our model.

We use Adagrad to optimize the parameters with the learning rate 0.15 and an initial accumulator value 0.1. The gradient clipping threshold is set to 2. In the training stage, we use the teacher forcing strategy to prevent the model from deviating. In the testing stage, when generating summaries, we use the beam search algorithm and set the beam size to 4. Moreover, due to the different length of the original article, we truncate it to 400 words, the maximum output length is set to 100, and the minimum output length is set to 35. We use GTX 1080Ti for training and set the batch size to 32. We performed 230,000 iterations for our model, it costs about 2 days. And then we added the coverage mechanism to our model and train 4000 iterations. We also find that the model works best when the weight of coverage loss is set to 0.85. We finally find that the coverage loss converged to 0.18 and the overall loss decrease to about 2.7. To find more optimal model, we save our model every 50 steps and calculate the loss on the validation set for these 80 models. We select the top 10 models to calculate the ROUGE metrics on the test set to find the best model.

D. Ablation Study

We conduct ablation experiments on the CNN/Daily Mail dataset to explore the effects of pretrained word embeddings and the first-encoder layer, in the ablation experiment, we only add each component to the baseline model, the experimental results are shown in Table I. It can be seen that using the pretrained word embeddings to initialize the word embedding matrix can not only speed up the convergence speed of the model but also improve the performance of model in a small degree. The ROUGE-1, ROUGE-2 and ROUGE-L scores increase 0.2, 0.33, and 0.35 respectively. The utilization of the two-stage encoder greatly improves the performance of our model. The ROUGE-1, ROUGE-2 and ROUGE-L index increase 0.67, 0.59, and 0.61 respectively. When we join both of two components, the ROUGE scores are enhanced largely. The ROUGE-1, ROUGE-2 ROUGE-L increase 0.75, 0.74, 0.77 respectively. The experiment results demonstrate the effectiveness of our proposed model.

TABLE II
ROUGE F1 SCORES ON THE TEST SET OF CNN/DAILY MAIL(WITH COVERAGE MECHANISM)

	ROUGE-1	ROUGE-2	ROUGE-L
PGN+cov(See et al., 2017)	39.53	17.28	36.38
PGN+cov(ours)	39.54	17.36	36.40
PGN+cov+Fasttext (DANG et al., 2019)	39.06	17.05	35.85
PGN+cov+Elmo (Claudio et al., 2019)	38.96	16.25	34.32
PGN+cov+Elmo (Ours)	39.15	17.36	36.09
TSEPE(ours)	**40.28**	**18.02**	**37.15**

TABLE III
ROUGE F1 SCORES ON THE TEST SET OF CNN/DAILY MAIL WITHOUT COVERAGE MECHANISM

	ROUGE-1	ROUGE-2	ROUGE-L
PGN(See et al., 2017)	36.95	16.17	33.81
PGN+Fasttext(DANG et al.,2019)	36.28	15.61	33.10
MHAS(Li et al.,2019)	36.82	15.86	33.58
TSEPE(ours)	**37.85**	**16.78**	**34.68**

E. Compared with Mainstream Abstractive Text Summarization Methods

For CNN/Daily Mail datasets, We further illustrate the effectiveness of the TSEPE model by comparing our model with the baseline model and other models designed to enhance the word representation for the PGN model. We divide the comparison results into two parts according to the models whether join the coverage mechanism:

Table II shows the comparison results of our model with some other models with coverage mechanism.

Table III shows the comparison results of our model with some other models without coverage mechanism.

It can be seen from Table II that although the PGN model with Fasttext can reduce the appearance of unknown words, the accuracy of this model is reduced because the word embeddings can not be fine-tuned for downstream tasks and cannot solve the polysemy problem. The method of replacing Fasttext with Elmo can get the word embeddings with contextual information, so it is slightly improved compared to Fasttext. However, there are few parameters that can be fine-tuned in downstream tasks for obtaining word embeddings, so they have poor performance in ROUGE metric. Compared with the baseline model, our TSEPE model improves the performance metrics by 0.75, 0.74 and 0.77 respectively.

In the MHAS model, during the decoding process, the information at both the encoder and decoder sides is combined. As shown in Table4, it can effectively prevent information loss. Our TSEPE model improves the ROUGE Metric by 0.9, 0.61, 0.87 respectively compared with the PGN model without coverage mechanism.

F. Qualitative Analysis

We analyze the quality of our model through the convergence speed and the real output on the CNN/Daily Mail dataset .

a) Speed of convergence: The curve drawn by a solid line in Fig. 3 shows the loss of the first 50,000 steps in the baseline

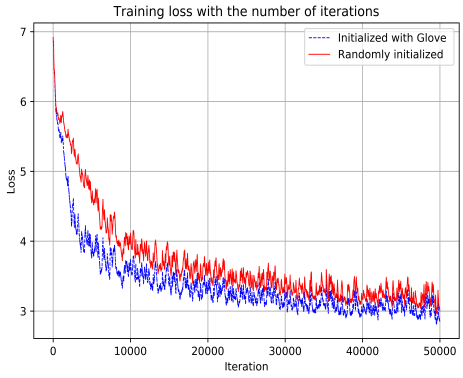

Fig. 3. Loss reduction curve during model training

model(the word embedding matrix is randomly initialized). The curve depicted by the dotted line shows the loss of the first 50,000 steps after using the pretrained word embeddings GloVe.

V. Conlusion

In this paper, we propose a text summarization model TSEPE. This model aims to solve the problem of the existing PGN model tending to ignore the global information and speed up the convergence of the PGN model. We first adopt a two-stage encoder method, which can well capture contextual semantic information of words and prevent the loss of global information. And then we initialize the embedding matrix by loading the pretrained embeddings GloVe which can be regarded as adding prior knowledge to TSEPE. This initialization strategy speeds up the convergence of the training process and gets a better solution. In future work, we plan to investigate how to integrate the semantic information of the decoded words into the decoding process. This is worthy of research because of the inconsistency between the training and testing phases during the decoding progress.

Acknowledgment

This work is supported by Shenzhen Science and Technology Planning Program under Grant No.JCYJ20170307141019252.

References

[1] C.-Y. Lin, “Rouge: A package for automatic evaluation of summaries,” in *Text summarization branches out*, 2004, pp. 74–81.

[2] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[3] A. M. Rush, S. Chopra, and J. Weston, “A neural attention model for abstractive sentence summarization,” in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015, pp. 379–389.

[4] S. Chopra, M. Auli, and A. M. Rush, “Abstractive sentence summarization with attentive recurrent neural networks,” in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2016, pp. 93–98.

[5] R. Nallapati, B. Zhou, C. dos Santos, G. Caglar, and B. Xiang, “Abstractive text summarization using sequence-to-sequence rnns and beyond,” in *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, 2016, pp. 280–290.

[6] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using rnn encoder–decoder for statistical machine translation,” in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1724–1734.

[7] A. See, P. J. Liu, and C. D. Manning, “Get to the point: Summarization with pointer-generator networks,” in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2017, pp. 1073–1083.

[8] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[9] J. Pennington, R. Socher, and C. D. Manning, “Glove: Global vectors for word representation,” in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[10] A. Joulin, E. Grave, and P. B. T. Mikolov, “Bag of tricks for efficient text classification,” *EACL 2017*, p. 427, 2017.

[11] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, “Deep contextualized word representations,” in *Proceedings of NAACL-HLT*, 2018, pp. 2227–2237.

[12] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019, pp. 4171–4186.

[13] D. T. Anh and N. T. T. Trang, “Abstractive text summarization using pointer-generator networks with pre-trained word embedding,” in *Proceedings of the Tenth International Symposium on Information and Communication Technology*, 2019, pp. 473–478.

[14] C. Mastronardo and F. Tamburini, “Enhancing a text summarization system with elmo,” in *CLiC-it*, 2019.

[15] J. Li, C. Zhang, X. Chen, Y. Cao, P. Liao, and P. Zhang, “Abstractive text summarization with multi-head attention,” in *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019, pp. 1–8.

[16] X. Qiu, T. Sun, Y. Xu, Y. Shao, N. Dai, and X. Huang, “Pre-trained models for natural language processing: A survey,” *Science China Technological Sciences*, pp. 1–26, 2020.

[17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[18] O. Vinyals, M. Fortunato, and N. Jaitly, “Pointer networks,” in *Advances in neural information processing systems*, 2015, pp. 2692–2700.

[19] Z. Tu, Z. Lu, Y. Liu, X. Liu, and H. Li, “Modeling coverage for neural machine translation,” in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2016, pp. 76–85.