electronics

MDPI

Article

A Hierarchical Representation Model Based on Longformer and Transformer for Extractive Summarization

Shihao Yang, Shaoru Zhang, Ming Fang, Fengqin Yang and Shuhua Liu *

School of Information Science and Technology, Northeast Normal University, Changchun 130117, China; yangsh861@nenu.edu.cn (S.Y.); zhangsr030@nenu.edu.cn (S.Z.); fangm000@nenu.edu.cn (M.F.); yangfq147@nenu.edu.cn (F.Y.)

* Correspondence: liush129@nenu.edu.cn

Abstract: Automatic text summarization is a method used to compress documents while preserving the main idea of the original text, including extractive summarization and abstractive summarization. Extractive text summarization extracts important sentences from the original document to serve as the summary. The document representation method is crucial for the quality of the generated summarization. To effectively represent the document, we propose a hierarchical document representation model Long-Trans-Extr for Extractive Summarization, which uses Longformer as the sentence encoder and Transformer as the document encoder. The advantage of Longformer as sentence encoder is that the model can input long document up to 4096 tokens with adding relative a little calculation. The proposed model Long-Trans-Extr is evaluated on three benchmark datasets: CNN (Cable News Network), DailyMail, and the combined CNN/DailyMail. It achieves 43.78 (Rouge-1) and 39.71 (Rouge-L) on CNN/DailyMail and 33.75 (Rouge-1), 13.11 (Rouge-2), and 30.44 (Rouge-L) on the CNN datasets. They are very competitive results, and furthermore, they show that our model has better performance on long documents, such as the CNN corpus.

Keywords: extractive summarization; transformer; longformer; deep learning

Citation: Yang, S.; Zhang, S.; Fang, M.; Yang, F.; Liu, S. A Hierarchical Representation Model Based on Longformer and Transformer for Extractive Summarization. *Electronics* **2022**, *11*, 1706. https://doi.org/10.3390/electronics11111706

Academic Editors: Phivos Mylonas, Katia Lida Kermanidis and Manolis Maragoudakis

Received: 30 April 2022
Accepted: 24 May 2022
Published: 27 May 2022

1. Introduction

Since Luhn [1] started automatic summarization research in 1958, great achievements have been made in this field. Text summarization can be divided into two categories: namely, abstractive and extractive summarization. Abstractive summarization [2] refines its ideas and concepts on the basis of understanding the semantic meaning of the original text to realize semantic reconstruction. Although more similar to the logic of human beings, abstractive summarization still faces a great challenge to produce a coherent, grammatical, and general summary of the original text, due to the limitations of natural language generation technology. The extractive summarization method extracts key sentences from a document to generate a summary. The input document is initially encoded, and then, the scores of sentences in the document are calculated. The sentences are sorted according to the scores, and those with high scores are selected to form a summary.

This study focuses on extractive summarization, since it not only generates semantically and grammatically correct sentences in news articles but also computes faster than abstractive summarization. At present, both generative and extractive summarization methods have some difficulties in processing long text, which is caused by the computational complexity of the encoder network. Recent studies have shown that Transformer [3] outperforms LSTM [4] in the area of natural language processing, both in terms of experimental results and computational complexity. However, even Transformer, which is capable of parallel computation, is unable to handle long text, resulting in the text summarization method being limited to short text. For a long text, there are usually two processing methods: (1) Discard the exceeding part directly. This method is simple to implement, but it has a great impact on the quality of the final summary. (2) Divide the long text into

several shorter text spans and process them one by one. As a result of this processing, different text spans cannot interact with each other, and therefore, a lot of information is inevitably lost. Of course, there are other mechanisms that can be added to enhance the interaction between text spans, but these new mechanisms are complex to implement, often task-specific, and not universal.

The main contributions of this paper are summarized as follows:

(1) This study proposes the hierarchical document representation method, which employs Longformer as the sentence encoder and Transformer as the document encoder to encode input text. Different from CNN (Convolutional Neural Network) or LSTM (Long and Short-Term Memory) as encoders [5–7], the model can deal with a long document, up to 4096 tokens, due to adopting Longformer as a sentence encoder, and makes it possible to directly encode long text.
(2) Both global attention and local attention [8] are adopted by encoders, which not only ensures that key tokens do not lose global information but also reduces computational complexity.
(3) The proposed hierarchical model achieves the best Rouge-1 and Rouge-L [9] on CNN/DailyMail datasets [10], and it achieves the state-of-the-art Rouge-1, Rouge-2, and Rouge-L on the long text dataset CNN. The best Rouge-1 and Rouge-L are achieved on the short text dataset DailyMail. Experimental results show that Longformer, as a sentence encoder, has good performance on long documents.

2. Related Work

Automatic text summarization includes abstractive and extractive summarization. In recent years, deep learning technology has provided a novel idea for research on summarization. Among the related literature, Cho et al. [11] and Sutskever et al. [12] proposed the widely studied sequence-to-sequence (seq2seq) model, which consists of an encoder and a decoder. Its basic idea is to use the global information of the input sequence to infer the corresponding output sequence. Rush et al. [13] first applied the above model to text summarization task.

In extractive summarization, an important issue is how to extract important sentences from the original document. Some studies are based on statistical methods [14,15]. With the success of deep neural networks in natural language processing, extractive summarization has achieved better results than traditional machine learning. The core of the extractive summarization model, based on a neural network, is the encoder-decoder structure. For the encoder, CNN, RNN (Recurrent Neural Network), and LSTM were adopted to capture the context information of the document [16–18]. However, with the above models, it is usually hard to capture long-distance dependency, especially in the case of a long document. With the success of BERT, the transformer is found to effectively capture sequence information of the input. Liu and Lapata [19] proposed a sentence-level encoder based on BERT, which is able to encode a document and obtain representations of its sentences. Then, they used Transformer to encode these sentence representations. Zhang et al. [20] proposed HIerachical BERT (HIBERT) for document encoding and pre-trained it with unlabeled data. First, they applied HIBERT, with unlabeled data, to the Sentence Prediction task and, then, to classify sentences. Wang et al. [21] presented HSG based on GNN (Graph Neural Network), adding fine-grained semantic nodes to assist in sentence extraction. For the decoder, Multilayer Perceptron (MLP) or LSTM are commonly used to output the score of sentences.

Due to the complexity of neural networks, the above methods have difficulty processing long documents. In order to reduce complexity, researchers proposed different methods: Wu and Hu [22] and Al-Sabahi et al. [16] limited the maximum sentence length and sentence number of documents; Zhong et al. [23] and Narayan et al. [17], respectively, intercepted the first 512 and 600 words of the document as input. Zhang et al. [20] limited the length of sentences and split long documents into short ones. The most direct and effective way to enable models to have longer input sequences is to reduce the complexity

of the network. Some studies have been performed by researchers [24,25]. Beltagy et al. [8] proposed the Longformer network. Longformer has three improved attention modes, from Transformer's attention mechanism, to reduce the complexity of the network: (1) sliding window attention; (2) dilated window attention; (3) sliding window attention + global attention. The author's experiments, on tasks such as question answering system and coreference analysis, show that the "local attention + global attention" model can achieve good performance under the premise of reducing computational complexity. Compared with Transformer, the computational complexity of Longformer is reduced from $O(n^2)$ to $O(n)$, where n is the length of the input sequence. Inspired by the above work, this paper adopts Longformer to encode text in an extractive summarization model to accept longer text input.

3. Proposed Model

In this section, summarization is modeled as a sequence labeling problem. Figure 1 shows the proposed extractive summarization model, Long-Trans-Extr, which is divided into three components: the sentence encoder, document encoder, and the classifier.

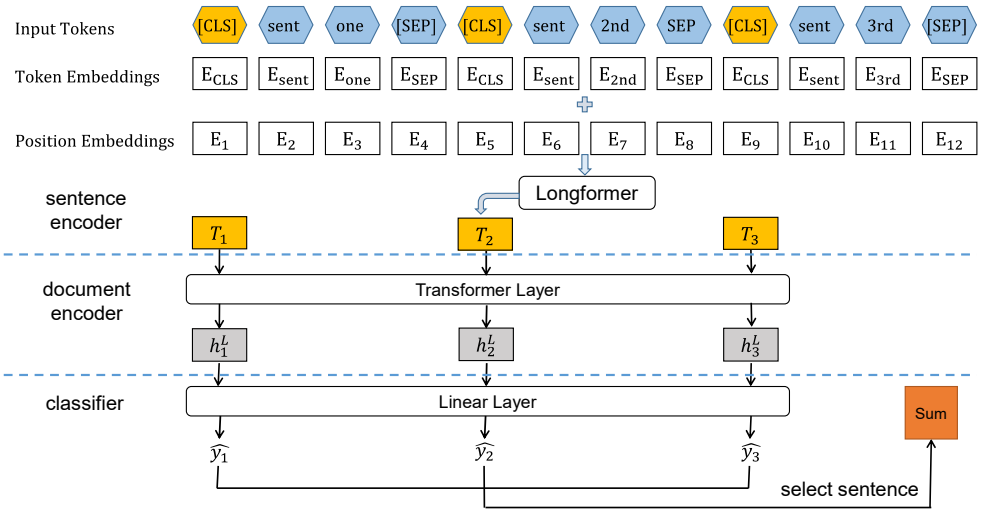

Figure 1. Long-Trans-Extr model structure.

3.1. Sentence Encoder

As described in Section 2, some previous models cannot directly input long documents. The original Transformer model has a global-attention mechanism with $O(n^2)$ time and memory complexity, where n is the input sequence length. With the increase in the input length, the computational complexity is unacceptable. Therefore, in order to encode long input text and, at the same time, reduce the computational complexity from $O(n^2)$ to $O(n)$, we introduce Longformer [8] as the sentence encoder. In addition, Longformer adopts the global attention + local attention mechanism. Local attention is a sliding window attention pattern. Each token only attends to its nearby w tokens. Attention calculation complexity has a linear relationship with the text sequence length n, and it is $O(w*n)$, where w is the size of the attention window. Global attention is a full length attention pattern. The token with global attention attends to all input tokens.

In this study, pre-trained language model Longformer is used as an encoder to output the representation vector of each sentence. We add [CLS] as the first token of each sentence to obtain its feature representation vector. It is worthwhile to note that [CLS] is a key token because its feature vector represents the feature vector of the current sentence. Therefore, we hope that [CLS] can capture more semantic information. In order to make Longformer serve the extractive summarization task, global attention is used for [CLS] tokens of the input sequence to capture more semantic information, and local attention is used for other tokens of the sentence to reduce computational complexity. The global attention + local

attention mechanism is shown in Figure 2. In this way, the model can capture long distance dependencies by adding a little computation.

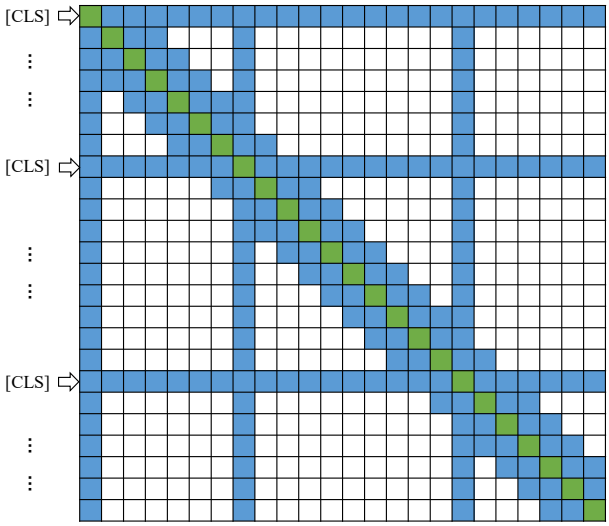

Figure 2. Attention mode of sentence encoder.

As shown in Figure 1, the vector for each word in the document is obtained as:

$$w_j^i = e_j^i + p_j \tag{1}$$

where w_j^i denotes the j-th word of the i-th sentence, e_j^i is the word embedding, p_j is position embedding [26], and $W = \left[w_1^1, w_2^1 \ldots, w_1^2, w_2^2 \ldots, \ldots, w_1^m, w_2^m, \ldots\right]$. Next, Longformer is used to encode all the words as:

$$T = Longformer(W) \tag{2}$$

where $T = [T_1, T_2, \ldots\ T_{\mathrm{m}}]$, m is the number of sentences in the document, T_i is output of the corresponding position of the ith [CLS], and it can be regarded as the representation vector of the ith sentence.

3.2. Document Encoder

To make sentence representation vectors interact in higher dimensions, we design a document encoder based on Transformer. The document encoder is shown in Figure 3. First, the initial input of the document encoder is computed by adding the position embedding p_i to each sentence vector T_i.

$$h_i^0 = T_i + p_i \tag{3}$$

where $h^0 = \left[h_1^0, h_2^0, \ldots, h_m^0\right]$ is the initial input of the document encoder and is input into the document encoder composed of the L-layer transformer.

$$\widetilde{h}^l = LN\left(h^{l-1} + MHAtt\left(h^{l-1}\right)\right) \tag{4}$$

$$h^l = LN\left(\widetilde{h}^l + FFN\left(\widetilde{h}^l\right)\right) \tag{5}$$

where $LN()$ is Layer Normalization [27], $MHAtt()$ is Multi Head Attention, and FFN is FeedForward Network. Equations (4) and (5) are the internal Transformer calculations in one layer of the document encoder, and they are executed L times, in turn, to obtain the sentence vectors using the L-layer transformer document encoder. The output of the last layer in the Transformer is $h^L = \left[h_1^L, h_2^L, \ldots, h_m^L\right]$, which will be input to the decoder.

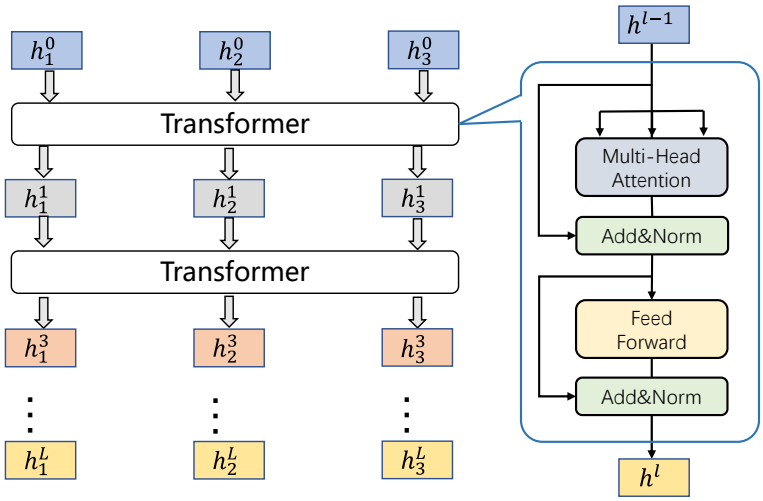

Figure 3. Document encoder structure.

3.3. Decoder

In extractive summarization, the decoder is, commonly, a binary classifier that sequentially predicts whether each sentence in the document should be extracted. Specifically, the model predicts a '0' or '1' label for each sentence. If the label is '1', the sentence is considered important and should be extracted to form a summary. In this study, we classify the final output from the document encoder by a binary classifier:

$$\hat{y}_i = sigmoid\left(h_i^L W_o + b_o\right) \tag{6}$$

where h_i^L is the sentence representation vector of $sent_i$, W_o is trainable weights, and b_o is a bias. The loss function is calculated as follows:

$$loss = BCE(\hat{y}_i, y_i) \tag{7}$$

Here, we use the *BCE* (Binary Cross Entropy) loss function, where $\hat{y}_i$ is the prediction label, and y_i is the ground truth.

4. Experiments

4.1. Datasets

Experiments are carried out on the CNN, DailyMail, and CNN/DailyMail benchmark datasets [10]. The "story highlights" in every document are assumed as standard summaries [27,28]. These standard summaries are used as the ground truth. The standard split of Hermann et al. [10] is adopted. Table 1 shows the details of the datasets. Each document contains 28 sentences and approximately 751 words in CNN datasets and 653 words in DailyMail datasets. Each gold summary contains three or four sentences. CoreNLP is used to split the sentence and preprocess datasets, following [28]. The documents and summaries are tokenized using the RoBERTa subwords tokenizer.

Table 1. Benchmark datasets (CNN, DailyMail, and CNN/DailyMail).

Dataset	Training	Validation	Testing
CNN	90,266	1220	1093
DailyMail	196,961	12,148	10,397
CNN/DailyMail	287,277	13,368	11,490

4.2. Evaluation Criteria

Recall-Oriented Understudy for Gisting Evaluation (Rouge) [9] is used to evaluate the proposed model. Rouge is a method of evaluating text similarity, which is commonly used in machine translation and automatic summarization. Rouge-N and Rouge-L are commonly used in automatic summarization tasks and denote the overlap rate of N-gram, as well as the longest common substring between the extracted summary and the gold summary, respectively. Rouge-N is computed as follows:

$$\text{Rouge} - \text{N} = \frac{\sum_{S \in \{ReferenceSumm\}} \sum_{gram_N \in S} Count_{match}(gram_N)}{\sum_{S \in \{ReferenceSumm\}} \sum_{gram_N \in S} Count(gram_N)} \tag{8}$$

where N stands for the length of the N-gram (N = 1, 2, ...). Rouge-N calculates the proportion of n-grams co-occurring in the extracted summary and the gold summary to all N-grams in the gold summary. It is clear that Rouge-N is a recall-related measure. The more sentences we extract, the higher the Rouge-N score will be, so the Rouge-N F1 score is commonly used in automatic summarization task.

Rouge-L is computed as follows:

$$R_L = \frac{LCS(X,Y)}{M} \tag{9}$$

$$P_L = \frac{LCS(X,Y)}{N} \tag{10}$$

$$F_L = \frac{(1+\beta^2) R_L P_L}{R_L + \beta^2 P_L} \tag{11}$$

where $LCS(X,Y)$ is the length of a longest common subsequence of X and Y, X is the extracted summary with M words, and Y is a gold summary with N words. In our experiments, F1 scores are computed for Rouge-N and Rouge-L. We use a Python-based calculation tool to evaluate the proposed model.

4.3. Experimental Settings

PyTorch is used to implement the extractive summarization model. The hardware platform is Intel i9-10900, the memory is 64 GB, and the GPU is RTX 3090. Sentence encoder Longformer is initialized with pre-trained weights of 'longformer-base'. Document encoder Transformer and the classifier are initialized randomly. The vector dimensions of words and sentences are 768, and the batch size is set to 64. The layer number of the document encoder Transformer is set to 2. In the first 1000 steps of training, only the weights of document encoder and classifier are adjusted, while the weights of sentence encoder remain unchanged. After 1000 steps, adjust the parameters of the model as a whole. The model is calculated on the validation set every 500 steps. Then, the best model parameters are saved, and its performance on the test set is considered the final result. The learning rate is set to 0.003 in the first 1000 steps, while after 1000 steps, it is set to 0.00003.

4.4. Experimental Results and Analysis

Firstly, we conducted experiments on the CNN/DailyMail datasets. In Table 2, KL-Summ [29], SumBasic [30], and LexRank [15] do not adopt deep learning methods. These methods show a large performance gap with those that use deep learning. Lead-3 is a commonly used, and effective, baseline that extracts the first three sentences in the document. Compared with the baseline model Lead-3, the proposed method has better performance. DQN [18], BANDITSUM [7], HSASRL [31], and Refresh [17] use reinforcement learning method to train their models. BERT-Extr [32], HIBERT [20], and BERTSUMEXT + TRIBLK [19] are based on the Transformer structured encoder. BERT-Extr extracts sentence singletons or sentence pairs. HIBERT trained the encoder on unlabeled data by hiding several sentences in the document encoding stage. BERTSUMEXT + TRIBLK has a similar

structure to our model, Long-Trans-Extr, but its sentence encoder is BERT. Therefore, its maximum input sequence length is limited to 512. HSG + Tri-Blocking [21] is based on GNN (Graph Neural Network). Compared to recent models, our model, Long-Trans-Extr, performs best on the Rouge-1 and Rouge-L on the CNN/DailyMail dataset, and the experimental results increased by 0.53 and 0.08, respectively.

Table 2. The experimental results of Long-Trans-Extr and other methods on the CNN/DailyMail datasets.

Model	Rouge-1	Rouge-2	Rouge-L
SumBasic [30]	34.11	11.13	31.14
LexRank [15]	35.34	13.31	31.93
KLSumm [29]	29.92	10.50	27.37
Lead-3	40.0	17.5	36.2
DQN [18]	39.4	16.1	35.6
BANDITSUM [7]	41.5	18.7	37.6
HSASRL [31]	41.5	19.5	37.9
HSSAS [16]	42.3	17.8	37.6
Refresh [17]	40.0	18.2	36.6
BERT-Extr [32]	41.13	18.68	37.75
HIBERT [20]	42.37	19.95	38.83
HSG + Tri-Blocking [21]	42.95	19.76	39.23
BERTSUMEXT + TRIBLK [19]	43.25	**20.24**	39.63
Long-Trans-Extr (ours)	**43.78**	19.83	**39.71**

In order to prove the proposed Longformer + Transformer has better effectiveness, we compare this model with other extractive models, whose decoders are the same as ours, but the encoders are different, on CNN/DailyMail datasets. The results are shown in Table 3. NN-SE [27] uses CNN and LSTM as the sentence encoder and document encoder, respectively, while BERT-Extr [32] adopts BERT as encoder. Our model is better than theirs. HSSAS [16] adopts Bi-LSTM + attention as hierarchical encoders. Our Longformer + Transformer encoder is better than the above models. A competitive model is BERTSUMEXT + TRIBLK [19], which achieves the best Rouge-2, while Long-Trans-Extr achieves the best Rouge-1 and Rouge-L. The experimental results show that the proposed Longformer + Transformer encoder can capture the contextual semantic information of the document more accurately.

Table 3. The comparison between our Long-Trans-Extr model and other extractive models, with the same decoder, on CNN/DailyMail datasets.

Model	Sentence-Encoder	Document-Encoder	Rouge-1	Rouge-2	Rouge-L
NN-SE [27]	CNN	LSTM	35.5	14.7	32.2
HSSAS [16]	Bi-LSTM + Attention	Bi-LSTM + Attention	42.3	17.8	37.6
BERT-Extr [32]	BERT	—	41.13	18.68	37.5
BERTSUMEXT + TRIBLK [19]	BERT	Transformer	43.25	**20.24**	39.63
Long-Trans-Extr (ours)	Longformer	Transformer	**43.78**	19.83	**39.71**

Table 4 shows the results of Long-Trans-Extr and other extraction models on CNN and DailyMail datasets separately. Compared to the previous best HSASRL model, Long-Trans-Extr improves by 2.83 (Rouge-1), 0.91 (Rouge-2), 3.04 (Rouge-L), as well as 2.01 (Rouge-1) and 1.11 (Rouge-L) on CNN and DailyMail datasets, respectively. The above results show that our model performs better on dataset CNN. In view of this result, we further analyze the datasets of CNN and DailyMail, as shown in Table 5. It can be seen that CNN has a longer average article length than DailyMail. We believe that the previous models only retained a limited number of tokens inputs, so CNN would be discarded more. However,

Longformer can allow up to 4096 token inputs, and almost no content would be discarded. This also shows that Long-Trans-Extr can effectively solve the long sequence input problem of extractive text summarization.

Table 4. The experimental results of Long-Trans-Extr and other methods on the CNN and DailyMail datasets, respectively.

Model	CNN			DailyMail		
	Rouge-1	**Rouge-2**	**Rouge-L**	**Rouge-1**	**Rouge-2**	**Rouge-L**
NN-SE [27]	28.4	10.0	25.0	36.2	15.2	32.9
Refresh [17]	30.4	11.7	26.9	41.0	18.8	37.7
BANDITSUM [7]	30.7	11.6	27.4	42.1	18.9	38.3
DQN [18]	—	—	—	41.9	16.5	33.8
HSASRL [31]	30.92	12.2	27.4	42.88	**20.48**	39.71
Long-Trans-Extr (ours)	**33.75**	**13.11**	**30.44**	**44.89**	20.02	**40.82**

Table 5. Comparison of document length between CNN and DailyMail datasets, where "Word_num" and "Sent_num" denote the average number of words and sentences in the document, respectively.

Dataset	Word_Num	Sent_Num
CNN	760.50	33.98
DailyMail	653.33	29.33

Table 6 shows the GPU memory usage and training time of global attention and Global + Local attention. As we can see, the local attention mechanism consumes less GPU memory and training time. When global attention is used for all tokens, the memory and training time increased by 43.7% and 45.6%, respectively.

Table 6. GPU memory usage and training time of global attention and Global + Local attention.

Attetion Mode	GPU Memory	Training Time/Epoch
Global	7014 MB	80.8 h
Global + Local	4881 MB	55.48 h

Figure 4 shows two extractive summarization examples, and each includes a generated summary and a gold summary. We mark different contents with different colors, and the same color represents the same content.

Generated summary 1 :the 25-year-old ghana international is out of contract in the summer and free to talk to foreign clubs .andre ayew is out of contract at marseille this summer , and swansea have joined the chase for his signatureswansea , meanwhile , have expressed an interest in schalke full-back christian fuchs .

Golden summary 1 :andre ayew is free to talk to foreign clubs with his marseille contract expiring at the end of the season . ghana international has received offers from swansea , newcastle and everton while wolfsburg and borussia dortmund are interested . the swans are also chasing schalke defender christian fuchs .

Generated summary 2 : harry kane and diego costa are locked on 19 goals in the race to finish as this season 's top scorer but rank fourth and fifth respectively , behind papiss cisse and olivier giroud .glenn murray has scored a goal on average every 91 minutes this season - giving him a better minutes-per-goal ratio than anyone else in the top flight .manchester city are coming up against statistically the deadliest striker in the premier league when they face crystal palace on easter monday .

Golden summary 2 : glenn murray has scored four goals in 364 minutes this season . crystal palace striker has best minutes-per-goal ratio in premier league . olivier giroud third on the list , harry kane fourth , diego costa fifth .

Figure 4. Examples of a generated summary and a golden summary.

5. Conclusions

In this study, we propose a Long-Trans-Extr extractive summarization model, which uses Longformer as a sentence encoder, Transformer as a document encoder, and finally, an MLP classifier is used to decide whether a sentence in a document should be extracted or not. This model solves the problem that it is difficult for previous models to deal with long documents. It enables sentence representation and document representation to notice longer text information without increasing too much computation and memory. Experimental results show that, under the same decoder condition, our model is superior to other models on the CNN/DailyMail dataset, and it achieves the best results on a long CNN dataset.

Author Contributions: This study was completed by the co-authors. S.L. led the research and wrote the draft. The model design is performed by S.Y. Major. Experiments and analyses were undertaken by S.Y. and S.Z. M.F. was responsible for data processing and drawing figures. F.Y. edited and reviewed the paper. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the project of Changchun Bureau of Science and Technology [grant number 21ZY31] and the project of Jilin Province Development and Reform Commission [the Grant 2022C047-5].

Data Availability Statement: All data generated or analysed during this study are included in this published article "Hermann Karl Moritz, Kocisky Tomas, Grefenstette Edward. Teaching machines to read and comprehend. Advances in neural information processing systems. 2015; 28 (1693–1701)".

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

References

1. Peter, L.H. The automatic creation of literature abstracts. *IBM J. Res. Dev.* **1958**, *2*, 159–165.
2. Chopra, S.; Auli, M.; Rush, A.M. 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, San Diego, CA, USA, 13–15 June 2016; pp. 93–98.
3. Vaswani, A.; Shazeer, N.; Parmar, N. Attention is all you need. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 5998–6008.
4. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
5. Bahdanau, D.; Brakel, P.; Xu, K. An actor-critic algorithm for sequence prediction. *arXiv* **2016**, arXiv:1607.07086.
6. Nallapati, R.; Zhai, F.; Zhou, B. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In Proceedings of the Thirty-first AAAI conference on artificial intelligence, San Diego, CA, USA, 4–9 February 2017; pp. 3075–3081.
7. Dong, Y.; Shen, Y.; Crawford, E. Banditsum:Extractive summarization as a contextual bandit. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, 2–4 November 2018; pp. 3739–3748.
8. Beltagy, I.; Peters, M.E.; Cohan, A. Longformer: The long-document transformer. *arXiv* **2020**, arXiv:2004.05150.
9. Lin, C.-Y. Rouge: A package for automatic evaluation of summaries. In Proceedings of the Text Summarization Branches Out, Barcelona, Spain, 25–26 July 2004; pp. 74–81.
10. Hermann, K.M.; Kocisky, T.; Grefenstette, E. Teaching machines to read and comprehend. *Adv. Neural Inf. Process. Syst.* **2015**, *28*, 1693–1701.
11. Cho, K.; van Merriënboer, B.; Gulcehre, C. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 25–29 October 2014.
12. Sutskever, I.; Vinyals, O.; Le Quoc, V. Sequence to sequence learning with neural networks. *Adv. Neural Inf. Process. Syst.* **2014**, *27*, 3104–3112.
13. Rush, A.M.; Chopra, S.; Weston, J. A neural attention model for abstractive sentence summarization. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP), Lisbon, Portugal, 17–21 September 2015; pp. 379–389.
14. Conroy, J.M.; O'leary, D.P. Text summarization via hidden markov models. In Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, New Orleans, LA, USA, 9–13 September 2001; pp. 406–407.
15. Mihalcea, R. Graph-based ranking algorithms for sentence extraction, applied to text summarization. In Proceedings of the ACL Interactive Poster and Demonstration Sessions, Barcelona, Spain, 21–26 July 2004; pp. 170–173.
16. Al-Sabahi, K.; Zuping, Z.; Nadher, M. A hierarchical structured self-attentive model for extractive document summarization (HSSAS). *IEEE Access* **2018**, *6*, 24205–24212. [CrossRef]

17. Narayan, S.; Cohen, S.B.; Lapata, M. Ranking sentences for extractive summarization with reinforcement learning. In Proceedings of the NAACL-HLT, New Orleans, LA, USA, 1–6 June 2018; pp. 1747–1759.
18. Yao, K.; Zhang, L.; Luo, T. Deep reinforcement learning for extractive document summarization. *Neurocomputing* **2018**, *284*, 52–62. [CrossRef]
19. Liu, Y.; Lapata, M. Text summarization with pretrained encoders. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing, Hong Kong, China, 3–7 November 2019; pp. 3730–3740.
20. Zhang, X.; Wei, F.; Zhou, M. HIBERT: Document level pre-training of hierarchical bidirectional transformers for document summarization. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL), Florence, Italy, 28 July–2 August 2019; pp. 5059–5069.
21. Wang, D.; Liu, P.; Zheng, Y. Heterogeneous graph neural networks for extractive document summarization. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL), Online, 6–8 July 2020.
22. Wu, Y.; Hu, B. Learning to extract coherent summary via deep reinforc-ement learning. In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018; pp. 5602–5609.
23. Zhong, M.; Liu, P.; Wang, D. Searching for effective neural extractive summarization: What works and what's next. In Proceedings of the 57th Conference of the Association for Computational Linguistics, Florence, Italy, 28 July–2 August 2019; pp. 1049–1058.
24. Kitaev, N.; Kaiser, Ł.; Levskaya, A. Reformer: The efficient transformer. In Proceedings of the ICLR, Addis Ababa, Ethiopia, 26–30 April 2020.
25. Ye, Z.; Guo, Q.; Gan, Q. Bp-transformer: Modelling long-range context via binary partitioning. *arXiv* **2019**, arXiv:1911.04070,.
26. Ba, J.L.; Kiros, J.R.; Hinton, G.E. Layer Normalization. *arXiv* **2016**, arXiv:1607.06450.
27. Cheng, J.; Lapata, M. Neural summarization by extracting sentences and words. In Proceedings of the Meeting of the Association for Computational Linguistics, Berlin, Germany, 7–12 August 2016.
28. See, A.; Liu, P.; Manning, C.D. Get to the point: Summarization with pointer-generator networks. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL), Vancouver, BC, Canada, 30 July–4 August 2017; pp. 1073–1083.
29. Haghighi, A.; Vanderwende, L. Exploring content models for multi-document summarization. In Proceedings of the Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Boulder, CO, USA, 31 May–5 June 2009; pp. 362–370.
30. Vanderwende, L.; Suzuki, H.; Brockett, C. Beyond SumBasic: Task-focused summarization with sentence simplification and lexical expansion. *Inf. Process.* **2007**, *43*, 1606–1618. [CrossRef] [PubMed]
31. Mohsen, F.; Wang, J.; Al-Sabahi, K. A hierarchical self-attentive neural extractive summarizer via reinforcement learning (HSASRL). *Appl. Intell.* **2020**, *50*, 2633–2646. [CrossRef]
32. Lebanoff, L.; Song, K.; Dernoncourt, F. Scoring sentence singletons and pairs for abstractive summarization. In Proceedings of the ACL, Florence, Italy, 28 July–2 August 2019.