Improving Abstractive Text Summarization with History Aggregation

Pengcheng Liao[1,2], Chuang Zhang[2], Xiaojun Chen[2], Xiaofei Zhou[2]
School of Cyber Security University of Chinese Academy of Sciences Beijing, China
Institute of Information Engineering Chinese Academy of Sciences Beijing, China
{liaopengcheng, zhangchuang, chenxiaojun, zhouxiaofei}@iie.ac.cn

***Abstract*—Recent neural sequence to sequence models have provided feasible solutions for abstractive summarization. However, such models are still hard to tackle long text dependency in the summarization task. A high-quality summarization system usually depends on strong encoder which can refine important information from long input texts so that the decoder can generate salient summaries from the encoder's memory. In this paper, we propose an aggregation mechanism based on the Transformer model to address the challenge of long text representation. Our model can review history information to make encoder hold more memory capacity. Empirically, we apply our aggregation mechanism to the Transformer model and experiment on CNN/DailyMail dataset to achieve higher quality summaries compared to several strong baseline models on the ROUGE metrics.**

I. INTRODUCTION

The task of text summarization is automatically compressing a long text to a shorter version while keeping the salient information. It can be divided into two approaches: extractive and abstractive. The extractive approach usually selects sentences or phrases from the source text directly. On the contrary, the abstractive approach first understands the semantic information of the source text and generates novel words not appeared in the source text. Extractive summarization is easier, but abstractive summarization is more like the way humans process text. This paper focuses on the abstractive approach. Unlike other sequence generation tasks in NLP(Natural Language Processing) such as NMT(Neural Machine Translation), in which the lengths of input and output text are close, the summarization task exists severe imbalance on the lengths. It means that the summarization task must model long-distance text dependencies.

As RNNs can tackle time sequence text, various sequence-to-sequence models [6] based on them have emerged on a large scale and these models can generate promising results. To solve the long-distance text dependencies, [7] first proposes the attention mechanism which allows each decoder step to refer to all encoder hidden states. [36] first incorporates attention mechanism to summarization task. There are also other attention-based models to ease the problem of long input texts for summarization task, like Bahdnau attention [2], hierarchical attention [4], graph-based attention [11] and simple attention [44]. [8] segments and encodes text independently then broadcasts their encoding to others. Though these systems are promising, they exhibit undesirable behaviors such as producing inaccurate factual details and repeating themselves as it is hard to decide where to attend and where to ignore for one-pass encoder.

Modeling an effective encoder for representing a long text is still a challenge in previous work, and we are committed to solving long text dependency problems by aggregation mechanism. The key idea of the aggregation mechanism is to collect history information to improve the expressiveness of the encoder by attention mechanism. It suggests that the encoder can read long input texts a few times to understand the text clearly. We build our model by reconstructing the Transformer model [1] by incorporating our novel aggregation mechanism. Empirically, we first analyze the features of summarization and translation dataset. Then we experiment with different encoder and decoder layers and the results reveal that the ability of the encoder layer is more important than the decoder layer, which implies that we should focus more on the encoder. Finally, we experiment on CNN/DailyMail dataset, and our model generates higher quality summaries compared to strong baselines on ROUGE metrics and human evaluations.

The main contributions of this paper are as follows:

- We put forward a novel aggregation mechanism to collect history information and apply it to the Transformer model.
- Our model outperforms about 1 ROUGE scores on CNN/DailyMail dataset and 5 ROUGE scores on our Chinese news dataset compared to the Transformer model.

II. RELATED WORK

In this section, we first introduce extractive summarization then introduce abstractive summarization.

A. *Extractive Summarization*

Extractive summarization aims to select salient sentences from source texts directly. This method is always modeled as a sentence ranking problem via selecting sentences with high scores [21], sequence labeling(binary label) problem [26] or integer linear programmers [22]. The models above mostly leverage manually engineered features, but they are now replaced by the neural network to extract features automatically. [9] gets sentence representation using

[1] Corresponding author: Chuang Zhang, zhangchuang@iie.ac.cn

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

CNN(convolutional neural network) and document representation using RNN(recurrent neural network) and then selects sentences/words using hierarchical extractor. [3] treats the summarization as a sequence labeling task. The model gets sentence and document representations using RNNs and after a classification layer, each sentence will get a label which indicates whether this sentence should be selected. [35] presents a model for extractive summarization by jointly learning score and selecting sentences. [41] puts forward a latent variable model to tackle the problem of sentence label bias.

B. Abstractive Summarization

Abstractive summarization aims to rewrite source texts with understanding semantic meaning. Most methods of this task are based on sequence to sequence models. [36] first incorporates the attention mechanism to abstractive summarization and achieves state of the art scores on DUC-2004 and Gigaword datasets. [10] improves the model performance via RNN decoder. [4] adopts a hierarchical network to process long source text with hierarchical structure. [16] is the first to show that a copy mechanism can take advantage of both extractive and abstractive summarization by copying words from the source text (extractive summarization) and generating original words (abstractive summarization). [2] incorporates copy and coverage mechanisms to avoid generating inaccurate and repeated words. [8] splits text to paragraph and applies encoder to each paragraph, then broadcasts paragraph encoding to others. Recently, [1] gives a new view of sequence to sequence model. It employs the self-attention to replace RNN in sequence to sequence model and uses multi-head attention to capture different semantic information.

Lately, more and more researchers focus on combine abstractive and extractive summarization. [18] builds a unified model by using inconsistency loss. [14] first trains content-selector to select and mask salient information then trains the abstractive model (Pointer Generator) to generate abstractive summarization.

III. MODEL

In this section, we first describe the attention mechanism and the Transformer baseline model, after that, we introduce the pointer and BPE mechanism. Our novel aggregation mechanism is described in the last part. The code for our model is available online.[1]

Notation We have pairs of texts $\{X, Y\}$, where $d \in X$ is a long text and $y \in Y$ is the summary of corresponding d. The lengths of d and y is ld and ly respectively. Each text d is composed by a sequence of words w, and we embed word w into vector e. So we represent document d with embedding vector $\{e^1, e^2, ..., e^{ld}\}$ and we can get representation of y the same as d.

A. Attention Mechanism

The attention mechanism is widely used in text summarization models as it can produce word significance distribution in source text for disparate decode steps. [7] first proposes the attention mechanism where attention weight distribution can be calculated:

$$e_i^t = v^\top tanh(w_s s_t + w_h h_i + b_i^t) \tag{1}$$

$$Attention^t = softmax(e^t) \tag{2}$$

Where h_i is the encoder hidden states in ith word, s_t is decoder hidden states at time step t. v, w_s, w_h and b_i^t are learnable parameters. $Attention^t$ is probability distribution that represents the importance of different source words for decoder at time step t.

Transformer redefines attention mechanism more concisely. In practice, we compute the attention function on a set of queries simultaneously, packed together into a matrix Q. The keys and values are also packed together into matrices K and V.

$$Attention(Q, K, V) = softmax(\frac{QK^\top}{\sqrt{d_k}})V \tag{3}$$

where $\top$ is transpose function, $Q \in R^{n\times dk}, K \in R^{m\times dk}, V \in R^{m\times dv}$, R is the real field, n, m are the lengths of query and key/value sequences, dk, dv are the dimensions of key and value. For summarization model we assume $K = V$. Self-attention can be defined from basic attention with $Q = K = V$. And multi-head attention concatenates multiple basic attentions with different parameters. We formulate multi-head attention as:

$$MH(Q, K, V) = Concat(hd_1, hd_2 ..., hd_i)w_{mh} \tag{4}$$

where $hd_i = Attention(Qw_i^Q, Kw_i^K, Vw_i^V)$, and $w_i^Q, w_i^K, w_i^V, w_{mh}$ are learnable parameters.

B. Transformer Baseline Model

Our baseline model corresponds to the Transformer model in NMT tasks. The model is different from previous sequence-to-sequence models as it applies attention to replace RNN. The Transformer model can be divided into encoder and decoder, and we will discuss them respectively below.

Input The attention defined in the Transformer is the bag of words(BOW) model, so we have to add extra position information to the input. The position encodes with heuristic sine and cosine function:

$$PE_{(pos,2i)} = \sin(pos/10000^{2i/d_{model}}) \tag{5}$$

$$PE_{(pos,2i+1)} = \cos(pos/10000^{2i/d_{model}}) \tag{6}$$

where pos is the position of word in text, i is the dimension index of embedding, and the dimension of model is d_{model}. The input of network U is equal to source text word embeddings $E_w = \{e^1, e^2, ..., e^{ld}\}$ added position embeddings $E_p = \{p^1, p^2, ...p^{ld}\}$.

[1] https://github.com/Pc-liao/Transformer_agg

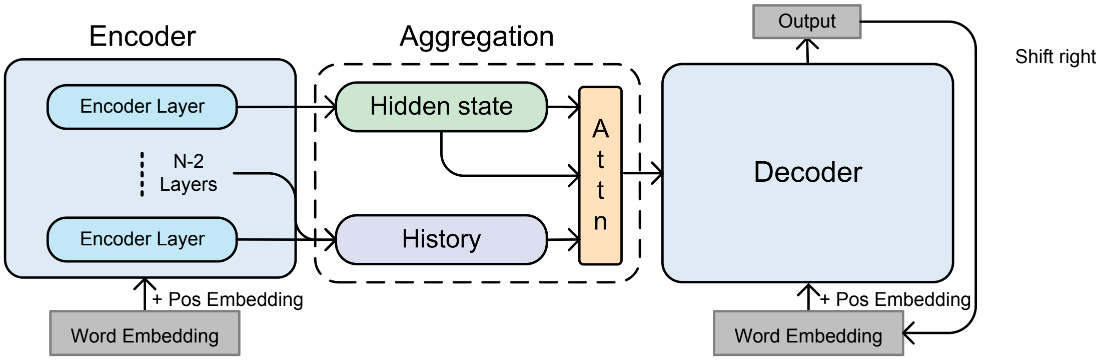

Fig. 1. Aggregation Transformer model overview. Compared with the Transformer baseline model, we apply the aggregation layer between encoder and decoder. The aggregation layer can collect history information to redistribute the encoder's final hidden states.

Encoder The goal of encoder is extracting the features of input text and map it to a vector representation. The encoder stacks with N encoder layers. Each layer consists of multi-head self-attention and position-wise feed-forward sublayers. We employ a residual connection around each of the two sublayers, followed by layer normalization. From the multi-head attention sublayer, we can extract different semantic information. Then we compute each encoder layer's final hidden states using position-wise feed-forward. The lth encoder layer is formulated as:

$$\begin{aligned} h_s^{(l)} &= Norm(MH(Q_s^{(l)}, K_s^{(l)}, V_s^{(l)}) + Q_s^{(l)}) \\ h_{el}^{(l)} &= Norm(PFF(h_s^{(l)}) + h_s^{(l)}) \\ &= Norm((relu(h_s^{(l)} w_s^{l1} + b_s^{l1}) w_s^{l2} + b_s^{l2}) + h_s^{(l)}) \end{aligned} \tag{7}$$

where $h_s^{(l)}$ is the multi-head self-attention output after residual connection and $Norm(.)$ is layer normalization function, $h_{el}^{(l)}$ means the output of encoder layer l. $Q_s^{(l)} = K_s^{(l)} = V_s^{(l)} = U$ if $l = 1$, or $Q_s^{(l)} = K_s^{(l)} = V_s^{(l)} = h_{el}^{(l-1)}$, w_s^{l1}, w_s^{l2} and b_s^{l1}, b_s^{l2} are learnable parameters, and $PFF(.)$ is the position-wise feed-forward sublayer. This sublayer also can be described as two convolution operations with kernel size 1.

Decoder The decoder is used for generating salient and fluent text from the encoder hidden states. Decoder stacks with N decoder layers. Each layer consists of masked multi-head self-attention, multi-head attention, and feed-forward sublayers. Similar to the encoder, we employ residual connections around each of the sublayers, followed by layer normalization. And we take lth decoder layer as example. We use the masked multi-head attention to encode summary as vector $h_{ms}^{(l)}$:

$$h_{ms}^{(l)} = Norm(MH^*(Q_{ms}^{(l)}, K_{ms}^{(l)}, V_{ms}^{(l)}) + h_{ms}^{(l)}) \tag{8}$$

where $Q_{ms}^{(l)} = K_{ms}^{(l)} = V_{ms}^{(l)} = (E_{gw} + E_{gp})$ in the first layer and $Q_{ms}^{(l)} = K_{ms}^{(l)} = V_{ms}^{(l)} = h_{dl}^{(l-1)}$ in other layers. $h_{dl}^{(l-1)}$ is the output of the $(l-1)$th decoder layer, E_{gw}, E_{gp} is the word embeddings and position embeddings of generated words respectively. The $MH^*(.)$ is masked multi-head self-attention and the mask is similar with the Transformer decoder. Then we execute multi-head attention between encoder and decoder:

$$h_d^{(l)} = Norm(MH(Q_d^{(l)}, K_d, V_d) + Q_d^{(l)}) \tag{9}$$

where $Q_d^{(l)} = h_{ms}^{(l)}$ is hidden states of decoder masked multi-head attention and $K_d = V_d = h_{el}^N$ is the last encoder layer output states. Finally, we use position-wise feed-forward and layer normalization sublayers to compute final states $h_{dl}^{(l)}$:

$$\begin{aligned} h_{dl}^{(l)} &= Norm(PFF(h_d^{(l)}) + h_d^{(l)}) \\ &= Norm((relu(h_d^{(l)} w_d^{l1} + b_d^{l1}) w_d^{l2} + b_d^{l2}) + h_d^{(l)}) \end{aligned} \tag{10}$$

where w_d^{l1}, w_d^{l2} and b_d^{l1}, b_d^{l2} are learnable parameters. After projecting the decoder final hidden states to vocab size, we can get vocabulary probability distribution P_{vocab}.

C. Pointer and BPE Mechanism

In generation tasks, we should deal with the OOV(out of vocabulary) problem. If we do not tackle this problem, the generated text only contains a limited vocabulary words and replaces OOVs with $< unk >$. Things get worse in summarization task, the specific nouns(like name, place, etc.) with low frequency are the key information of summary, however, the vocabulary built with top k words with the most frequent occurrence while those specific nouns may not occur in vocabulary.

The pointer and BPE(byte pair encoder) mechanism are both used to tackle the OOV problem. The original BPE mechanism is a simple data compression technique that replaces the most frequent bytes pair with unused byte. [37] first uses this technique for word segmentation via merging characters instead of bytes. So the fixed vocabulary can load more subwords to alleviate the problem of OOV.

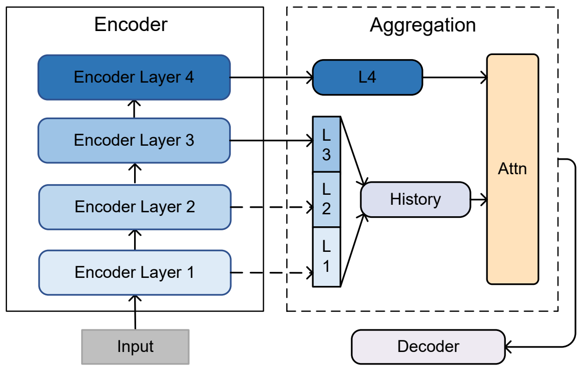

Fig. 2. The overview of projection aggregation mechanism with 4 encoder layers.

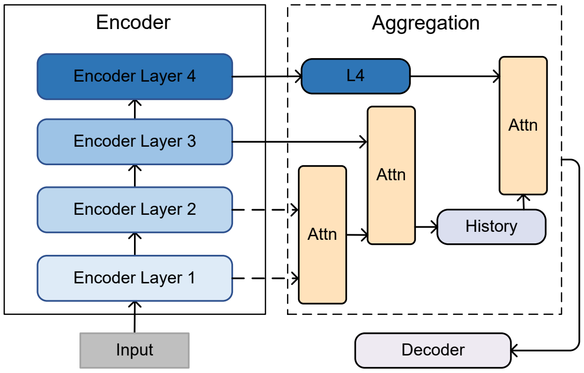

Fig. 3. The overview of attention aggregation mechanism with 4 encoder layers.

The pointer mechanism allows both copying words from the source text and generating words from a fixed vocabulary. For pointer mechanism at each decoder time step, the generation probability $P_{gen} \in [0, 1]$ can be calculated:

$$P_{gen} = \sigma(w_{dl} h_{dl}^N + b_{gen}) \tag{11}$$

where w_{dl} and b_{gen} are learnable parameter. h_{dl}^N is the last decoder output states. We compute the final word distribution via pointer network:

$$\alpha = softmax(h_{dl}^N u^\top + b_{copy}) \tag{12}$$

$$P_{copy} = \sum_{1}^{ld} \alpha z_i \tag{13}$$

$$P_{final} = P_{copy}(1 - P_{gen}) + P_{vocab} P_{gen} \tag{14}$$

where u is representation of input, z_i is one-hot indicator vector for w^i, P_{copy} is probability distribution of source words and P_{final} is final probability distribution.

D. Aggregation Mechanism

The overview of our model is in Fig. 1. To enhance memory ability, we add the aggregation mechanism between encoder and decoder for collecting history information. The aggregation mechanism reconstructs the encoder's final hidden states by reviewing history information. And we put forward two primitive aggregation approaches that can be proved effective in our task.

The first approach is using full-connected networks to collect historical information(see Fig. 2). This approach first goes through normal encoder layers to get the outputs of each layer, and we select middle L layers' outputs then concatenate them as input of full connected networks to obtain history information $H = h^h$. Finally, we compute multi-head attention between history state H and the output of the last encoder layer. This process can be formulated as:

$$h^h = w^h(Concat(h_{el}^{(N-L)}, ..., h_{el}^{(N-1)})) + b^h \tag{15}$$

where w^h and b^h are learnable parameters, L is hyper-parameter to be explored. Then we add the multi-head attention layer between the last encoder layer output h_{el}^N and history information h^h. The output of attention is the final states of encoder:

$$h^a = MH(Q^p, K^p, V^p) \tag{16}$$

where Q^p is history information h^p and $K^p = V^p = h_{el}^N$.

The second approach is using attention mechanism to collect history information(see Fig. 3). We select middle L encoder layers' outputs to iteratively compute multi-head attention between current encoder layer output and previous history information. And the lth history information $h^{h(l)}$ can be calculated as follows:

$$h^{h(l)} = MH(Q^{p(l)}, K^{p(l)}, V^{p(l)}) \tag{17}$$

where $l \in [N - L, N)$ is index of selected encoder layers, $Q^{p(l)}$ is previous history state $h^{h(l-1)}$ and $K^{p(l)} = V^{p(l)}$ is encoder output h_{el}^l . Iteratively calculating history information until the last selected encoder layer, we can get final history hidden states h^a and make the states as the final states of the encoder.

Finally, we define the objective function. Given the golden summary Y and input text X, we minimize the negative log-likelihood of the target word sequence. The training objective function can be described:

$$J(\theta) = \sum^{N} - \log p(Y|X; \theta) \tag{18}$$

where θ is model parameter and N is the number of source-summary text pairs in training set. The loss for one sample can be added by the loss of generated word y_t in each time step t:

$$\log(y|d; \theta) = \sum_{t=1}^{T} \log p(y_t|y_1, y_2, ...y_{(t-1)}, d; \theta) \tag{19}$$

where $p(y_t|y_1, y_2, ...y_{(t-1)}, X; \theta)$ can be calculated in decoder t time step, T is total decoding steps.

IV. EXPERIMENTS

In this section, we first define the setup of our experiment and then analyze the results of our experiments.

A. Experimental Setup

Dataset We conduct our experiments on CNN/DailyMail dataset[43], [4], which has been widely used for long document summarization tasks. The corpus is constructed by collecting online news articles and human-generated summaries on CNN/Daily Mail website. We choose the non-anonymized version[1][2], which is not replacing named entity with a unique identifier. The dataset contains pairs of articles and summaries. The details of this dataset are in section IV-B.

Training Details We conduct our experiments with 1 NVIDIA Tesla V100. During training and testing time we truncate the source text to 500 words and we build a shared vocabulary for encoder and decoder with small vocabulary size 50k, due to the using of the pointer or BPE mechanism. Word embeddings are learned during training time. We use Adam optimizer with initial learning rate 10^{-4} and parameter $\beta_1 = 0.9, \beta_2 = 0.999$ in training phase. We adapt the learning rate according to the loss on the validation set (half learning rate if validation set loss is not going down in every two epochs). And we use regulation with all $dropout = 0.1$. The training process converges about 200,000 steps for each model.

In the generation phase, we use the beam search algorithm to produce multiple summary candidates in parallel to get better summaries and add repeated words to blacklist in the processing of search to avoid duplication. For fear of favoring shorter generated summaries, we utilize the length penalty. In detail, we set beam size 10, no-repeated n-gram size 3 and length penalty parameter 2.0. We also constrain the maximum and minimum length of the generated summary to 120 and 50 respectively.

We evaluate our system using F-measures of ROUGE-1, ROUGE-2, ROUGE-L metrics which respectively represent the overlap of N-gram and the longest common sequence between the golden summary and the system summary. The scores are computed by python pyrouge[2] package.

Experiment explorations We explore the influence of different experiment hyper-parameters setup for the model's performance, which includes 11 different experiment settings.

Firstly, we explore the number of Transformer encoder/decoder layers (see Table III).

Secondly, we dig out the different aggregation methods with 1 aggregation layer (see Table IV). The exploration includes our baseline model(**m1**) and Transformer model with add function(**m2**), projection aggregation method(**m4**) and attention aggregation method(**m6**).

Thirdly, we also explore the different performance of different number of aggregation layers (see Table IV). There are 3 groups of experiments with different number of aggregation layers: Transformer adding last 2 layers(**m2**) and last 3 layers(**m3**), Transformer with projection aggregation method using 1 layer(**m4**) and 2 layers(**m5**) and Transformer with attention aggregation method using 1 layer(**m6**) and 2 layers(**m7**). For all models except the exploration of encoder/decoder layers, we use 4 encoder and 4 decoder layers.

Human Evaluation The ROUGE scores are widely used in the automatic evaluation of summarization, but it has great limitations in semantic and syntax information. In this case, we use manual evaluation to ensure the performance of our models. We perform a small scale human evaluations where we randomly select about 100 generated summaries from each of the 3 models(Pointer Generator, Transformer, and aggregation Transformer) and randomly shuffle the order of 3 summaries to anonymize model identities, then let 20 anonymous volunteers with excellent English literacy skills score random 10 summaries for each 3 models range from 1 to 5(high score means high-quality summary). then we using the average score of each summary as their final score. the evaluation criteria are as follows: (1) salient: summaries have the important point of the source text, (2) fluency: summaries are consistent with human reading habits and have few grammatical errors, (3) non-repeated: summaries do not contain too much redundancy word.

B. Results

Dataset Analysis To demonstrate the difference between summarization and translation tasks, we compare the dataset for two tasks (see Table II). The summarization dataset CNN/DailyMail contains 287226 training pairs, 13368 validation pairs, and 11490 test pairs. The translation dataset iwslt14 and wmt17 have 160239/3961179 training pairs, 7283/40058 validation pairs, and 6750/3003 test pairs respectively. Then we find the characteristics of those two different tasks after comparison. The summarization source text can include more than 2000 words and the average length of the source texts is 10 times longer than the target texts, while the translation task contains at most 250 words and the average length of the source texts is about the same as the target texts. Because of that, we need a strong encoder with memory ability to decide where to attend and where to ignore.

Quantitative Analysis The experimental results are given in Table I. Overall, our model improves all other baselines(reported in their articles) for ROUGE-1, 2 F1 scores, while our model gets a lower ROUGE-L F1 score than the RL (Reinforcement Learning) model [45]. From [8], the ROUGE-L F1 score is not correlated with summary quality, and our model generates the most novel words compared with other baselines in novelty experiment Fig. 5. The novel words are harmful to ROUGE-2, L F1 scores. This result also account for our models being more abstractive.

Fig. 4 shows the ground truth summary, the generated summaries from the Transformer baseline model and our aggregation Transformer using the attention aggregation method. The source text is the main fragment of the truncated text. Compared with the aggregation Transformer, the summary generated by the Transformer baseline model have two prob-

[1] https://github.com/abisee/cnn-dailymail
[2] https://pypi.org/project/pyrouge/

TABLE I

COMPARISON OF DIFFERENT MODEL RESULTS ON CNN/DALIYMAIL TEST DATASET USING F1 SCORES OF ROUGE-1, ROUGE-2, ROUGE-L WITH 95% CONFIDENCE INTERVAL. THE FIRST PART IS PREVIOUS ABSTRACTIVE BASELINE MODELS, THE SECOND PART IS THE TRANSFORMER BASELINE MODEL AND OUR TRANSFORMER MODEL WITH AGGREGATION MECHANISM. THE BEST SCORES ARE BOLDED.

Model	ROUGE-1	ROUGE-2	ROUGE-L
lead-3	40.24	17.52	36.34
words-1vt2k-temp-att [4]	36.64	15.66	33.42
ConvS2S [13]	39.75	17.29	36.54
Pointer Generator + Coverage [2]	39.53	17.28	36.38
Pointer Generator + Coverage + cbdec + RL[5]	40.66	17.87	37.06
Inconsistency Loss [18]	40.68	17.97	37.13
rnn-ext + abs + RL + rerank [45]	40.88	17.80	**38.54**
Transformer	40.05	17.72	36.77
Aggregation Transformer(attention)	**41.06**	**18.02**	**38.04**

TABLE II

THE COMPARISON OF TRANSLATION AND SUMMARIZATION DATASETS. WE REMOVE SENTENCE TAGS IN THE SOURCE TEXT AND SPLIT SENTENCES WITH BLANK, THEN COUNT MAXIMAL AND AVERAGE LENGTH TOKEN IN EACH DATASET.

Dataset	Train	Valid	Test
CNN/DailyMail(summarization)	287226	13368	11490
max-token-len(art/abs)	2882 / 2096	2134 / 1684	2377 / 678
avg-token-len(art/abs)	790 / 55	768 / 61	777 / 58
Our Dataset(summarization)	48600	4800	6600
max-token-len(art/abs)	1914 / 80	1687 / 80	1670 / 80
avg-token-len(art/abs)	768 / 65	763 / 65	769 / 65
iwslt14-de-en(translation)	160239	7283	6750
max-token-len(de/en)	244 / 228	169 / 154	245 / 217
avg-token-len(de/en)	24 /24	24 / 24	23 / 22
wmt17-en-de(translation)	3961179	40058	3003
max-token-len(en/de)	250 /250	224 / 233	101 / 93
avg-token-len(en/de)	28 /29	28 / 29	26 / 27

TABLE III

WE COMPARE DIFFERENT LAYERS OF ENCODER(E) AND DECODER(D) AND REPORT RESULTS ON CNN/DAILYMAIL TEST DATASET USING PRECISION/RECALL/F1 SCORES OF ROUGE.

E/D	ROUGE-1(P/R/F1)			ROUGE-2(P/R/F1)			ROUGE-l(P/R/F1)		
4/4	40.46	41.53	40.05	18.11	18.42	17.72	36.42	37.15	36.77
4/3	40.88	40.47	39.75	18.40	17.93	17.63	37.07	36.70	36.50
4/2	41.70	39.23	39.54	18.78	17.26	17.47	37.96	35.87	36.51
2/4	39.88	41.26	39.57	17.67	18.07	17.30	35.97	37.00	35.98
3/4	40.46	40.01	39.80	18.05	18.10	17.54	36.63	37.07	36.43

TABLE IV

THE AGGREGATION MECHANISM EXPERIMENTS. OUR EXPERIMENTS USE 3 AGGREGATION METHODS WITH 2 DIFFERENT AGGREGATION LAYERS.

Model	ROUGE-1	ROUGE-2	ROUGE-L
(m1)Transformer	40.05	17.72	36.77
(m2)Transformer(add 1 layer)	39.79	17.52	36.32
(m3)Transformer(add 2 layers)	39.69	17.34	36.15
(m4)Agg-Transformer(proj 1 layer)	40.58	17.77	36.60
(m5)Agg-Transformer(proj 2 layers)	40.67	17.84	36.70
(m6)Agg-Transformer(attn 1 layer)	**41.06**	**18.02**	**38.04**
(m7)Agg-Transformer(attn 2 layers)	40.03	17.59	36.60

lems. Firstly, the summary of the baseline model is lack of salient information marked with red in the source text. Secondly, it contains unnecessary information marked with blue in the source text.

we hold the opinion that the Transformer baseline model has weak memory ability compared to our model. Therefore, it can not remind the information far from its current states which will lead to missing some salient information and it may remember irrelevant information which will lead to unnecessary words generated in summaries. Our model uses the aggregation mechanism that can review the primitive information to enhance the model memory capacity. Therefore, the aggregation mechanism makes our model generate salient and non-repetitive words in summaries.

Encoder/Decoder Layers Analysis The first exploration experiment consists of Transformer models using different encoder and decoder layers. And we only experiment if the number of encoder/decoder layers is no more than 4. We also tried 6 encoder and decoder layers, however, there is no notable difference with 4 encoder and decoder layers but increasing a lot of parameters and taking more time to converge. Therefore we make the Transformer baseline model have 4 encoder and decoder layers.

We decrease the layers of encoder or decoder respectively, and the results are shown in Table III. It can be concluded from the comparison of each model results that we can get lower precision but higher recall score when the encoder layers are decreasing and we have opposite results on the decoder layers decreasing experiments. Meanwhile, we can get a higher ROUGE-1 F1 score and lower ROUGE-2, L F1 scores in the model decreasing each 1 decoder layer compared to that decreasing each 1 encoder layer. Therefore, we can conclude that the encoder captures the features of the source text while the decoder makes summaries consistently.

Aggregation mechanism Analysis The second exploration experiment consists of our baseline model(**m1, m2**) and aggregation Transformer model using different aggregation mechanism(**m4, m6**) in Table IV. If we use baseline model adding the last L layer(s) simply(**m2**), the result scores will decrease beyond our expectation. However, simply adding the last L layer(s) can re-distribute the encoder final states with history states, it will average the importance weights of those layers and that maybe get things worse. Compared with the baseline model, the result scores of our aggregation models(**m4, m6**) are boosting. We compute attention between history(query) and encoder final states(key/value) to re-distribute the final

Source Text(truncated 500): (......) national grid has revealed the uk 's first new pylon for nearly 90 years . called the t-pylon -lrb- artist 's illustration shown -rrb- it is a third shorter than the old lattice pylons . but it is able to carry just as much power - 400,000 volts . it is designed to be less obtrusive and will be used for clean energy purposes . national grid is building a training line of the less obtrusive t-pylons at their eakring training academy in nottinghamshire . britain 's first pylon , erected in july 1928 near edinburgh , was designed by architectural luminary sir reginald blomfield , inspired by the greek root of the word ' pylon ' -lrb- meaning gateway of an egyptian temple -rrb- . the campaign against them - they were unloved even then - was run by rudyard kipling , john maynard keynes and hilaire belloc . five years later , the biggest peacetime construction project seen in britain , the connection of 122 power stations by 4,000 miles of cable , was completed . it marked the birth of the national grid and was a major stoking of the nation 's industrial engine and a vital asset during the second world war (......)

Ground Truth: national grid has revealed the uk 's first new pylon for nearly 90 years . called the t-pylon it is a third shorter than the old lattice pylons . but it is able to carry just as much power - 400,000 volts . it is designed to be less obtrusive and will be used for clean energy .

Transformer Baseline: the t-pylon -lrb- artist 's shown -rrb- it is a third shorter than the old lattice pylons . but it is able to carry just as much power - 400,000 volts . it is designed to be less obtrusive and will be used for clean energy purposes .

Our model: national grid has revealed the uk 's first new pylon for nearly 90 years . called the t-pylon it is a third shorter than the old lattice pylons . but it is able to carry just as much power - 400,000 volts . it is designed to be less obtrusive and will be used for clean energy purposes .

Fig. 4. The comparison of ground truth summary and generated summaries of 2 abstractive summarization models on CNN/DailyMail dataset. The red represents missed information, the blue means unnecessary information and the green signify appropriate information.

states so that the encoder obtains the ability to fusing history information with different importance.

The third exploration contains 3 groups experiments: add group(**m2, m3**), projection group(**m4, m5**) and attention group(**m6, m7**). The aggregation Transformer models here use different aggregation layers. We also experiment with the model in the above 3 groups with 3 aggregation layers, but they all get extraordinary low ROUGE scores (all 3 models have ROUGE-1 39.3, ROUGE-2 14.5, ROUGE-L 34.3 roughly). They all incorporate the output of the first encoder layer which may not have semantic information which may be harmful to the re-distributing of the encoder final states. So we do not compare with those models explicitly.

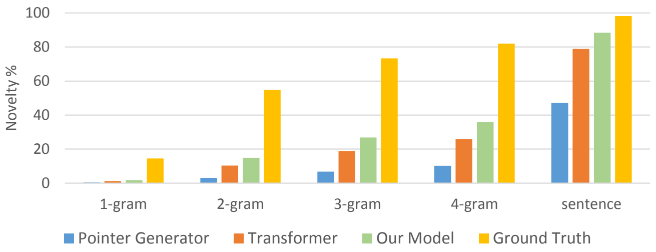

Fig. 5. The statistics of novel n-grams and sentences. Our model can generate far more novel n-grams and sentences than Pointer Generator and Transformer baseline.

For add aggregation group, we increase the added layers while the ROUGE scores will get down. If we add more layers, the final state distributions will tend to be the uniform distribution which makes decoder confused about the key ideas of source text. For that reason, we may get worse scores when we add more layers.

For the projection aggregation group, we increase the aggregation layers and the ROUGE scores will rise. If we aggregate more layers, the history states will contain more information which will lead to performance improvement. However, we will lose a lot of information when the aggregation layers increasing. And we achieve the best result with 2 aggregation layers.

For the attention aggregation group, we get the best score with 1 aggregation layer but the ROUGE scores will decline if we increase the aggregation layers. We just need one layer attention to focus on history states, because too much attention layers may have an excessive dependency on history states. If the encoder final distribution focus more on shallow layers which introduced a lot of useless information, it is harmful to the encoder to capture salient features.

Abstractive analysis Fig. 5 shows that our model copy 10% whole sentences from source texts, and the copy rate is almost close to reference summaries. However, there is still a huge gap in n-grams generation, and this is the main area for improvement.

In particular, the Pointer Generator model tends to examples with few novel words in summaries because of its lower rate of novel words generation. The Transformer baseline model can generate novel summaries and our model get great improvement (with 0.5, 4.6, 7.8, 10.1% novelty improvement for n-gram($n \in \{1, 2, 3, 4\}$)) compared to the Transformer baseline model. Because our model reviews history states and re-distribute encoder final states, we get more accurate semantic representation. It also proves that our aggregation mechanism can improve the memory capability of encoder.

Human Evaluation We conduct our human evaluation with setup in section IV-A, and the results show in Table V. We only compared three models on salient, fluency and non-repeated criteria, and our model gets the highest score in all criteria. But in fluency criterion, none of the models scores well, which means it is hard to understand semantic information for all models now. The Pointer Generator is our baseline

TABLE V
HUMAN EVALUATION OF THREE MODELS. WE COMPARE THE AVERAGE SCORE OF SALIENT, FLUENCY AND NON-REPEATED. THE BEST SCORES ARE BOLDED.

Model	Salient	Fluency	Non-Repeated
Pointer Generator	3.37	3.12	3.17
Pointer Generator + Coverage	3.42	3.23	3.61
Transformer	3.56	3.30	3.67
Transformer + Aggregation	**3.87**	**3.37**	**3.78**

TABLE VI
EXPERIMENTS ON OUR CHINESE DATASET. WE ONLY EXPERIMENT ON THREE BASELINE MODELS AND EVALUATE RESULTS WITH ROUGE F METRICS. THE BEST SCORES ARE BOLDED.

Model	ROUGE-1	ROUGE-2	ROUGE-L
Lead-3	54.09	42.46	34.56
Pointer Generator	55.49	43.59	48.03
Pointer Generator + Coverage	55.64	43.80	48.08
Transformer	52.69	39.86	43.66
Transformer + Aggregation	**58.00**	**44.42**	**48.85**

abstractive summarization approach and has the lowest scores. The Pointer Generator uses the coverage mechanism to avoid generating overlap words, which can make summaries more fluent and less repetitive. The Transformer is a new abstractive summarization based on attention mechanism, and it can get better performance than the Pointer Generator model. We equip the Transformer model with the aggregation mechanism, and it can get great improvement on all 3 criteria.

C. Our Chinese Experiments

We build our Chinese summarization dataset via crawling news website[1] and process the raw web page contents to character-based texts. The details of our dataset show in Table II where our dataset has a similar average length of source texts and summaries compared CNN/DM dataset. It is a temporary dataset, which only contains 60,000 pairs of text totally for now, and we are still adding data to our dataset.

We also experiment on our Chinese dataset and evaluate the result with ROUGE metrics. Our model gets the highest score, while the Pointer Generator model gets rather high ROUGE scores (see Table VI). Because the dataset does not contain many novel words where it is suitable for the Pointer Generator model. Our dataset contains (6.17, 14.51, 17.99, 20.10)% novel (1,2,3,4)-gram and 59.90% novel sentences; by comparison, the novel n-gram and sentences frequency of CNN/DM in Fig. 5 is (14.47, 54.75, 73.32, 82, 98.16)% respectively. And the Pointer Generator model generates summaries containing less novel words and sentences, which leads to high scores in our Chinese dataset. Finally, we compare our model with the Transformer baseline model, and our results improve 5.31 in ROUGE-1, 4.56 in ROUGE-2 and 5.19 in ROUGE-L scores.

V. CONCLUSIONS

In this paper, we propose a new aggregation mechanism for the Transformer model, which can enhance encoder memory ability. The addition of the aggregation mechanism obtains the best performance compared to the Transformer baseline model and Pointer Generator on CNN/DailyMail dataset in terms of ROUGE scores. We explore different aggregation methods: add, projection and attention methods, in which attention method performs best. We also explore the performance of different aggregation layers to improve the best score. We build a Chinese dataset for the summarization task and give the statistics of it in Table II. our proposed method also achieves the best performance on our Chinese dataset.

In the future, we will explore memory network to collect history information and try to directly send history information to the decoding processing to improve the performance in the summarization task. And the aggregation mechanism can be transferred to other generation tasks as well.

ACKNOWLEDGMENT

This work was supported by the National Natural Science Foundation of China (Grant No.61602474).

REFERENCES

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” *arXiv: Computation and Language*, 2017.

[2] A. See, P. J. Liu, and C. D. Manning, “Get to the point: Summarization with pointer-generator networks,” vol. 1, pp. 1073–1083, 2017.

[3] R. Nallapati, F. Zhai, and B. Zhou, “Summarunner: A recurrent neural network based sequence model for extractive summarization of documents,” *arXiv: Computation and Language*, 2016.

[4] R. Nallapati, B. Zhou, C. N. D. Santos, C. Gulcehre, and B. Xiang, “Abstractive text summarization using sequence-to-sequence rnns and beyond,” pp. 280–290, 2016.

[5] Y. Jiang and M. Bansal, “Closed-book training to improve summarization encoder memory,” pp. 4067–4077, 2018.

[6] 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.

[7] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” *arXiv preprint arXiv:1409.0473*, 2014.

[8] A. Celikyilmaz, A. Bosselut, X. He, and Y. Choi, “Deep communicating agents for abstractive summarization,” *arXiv preprint arXiv:1803.10357*, 2018.

[9] J. Cheng and M. Lapata, “Neural summarization by extracting sentences and words,” *arXiv preprint arXiv:1603.07252*, 2016.

[10] 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.

[11] J. Tan, X. Wan, and J. Xiao, “Abstractive document summarization with a graph-based attentional neural model,” vol. 1, pp. 1171–1181, 2017.

[12] X. Duan, M. Yin, M. Zhang, B. Chen, and W. Luo, “Zero-shot cross-lingual abstractive sentence summarization through teaching generation and attention,” in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 3162–3172.

[13] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, “Convolutional sequence to sequence learning,” in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 1243–1252.

[14] S. Gehrmann, Y. Deng, and A. M. Rush, “Bottom-up abstractive summarization,” *arXiv preprint arXiv:1808.10792*, 2018.

[15] W. Zeng, W. Luo, S. Fidler, and R. Urtasun, “Efficient summarization with read-again and copy mechanism,” *arXiv: Computation and Language*, 2017.

[16] J. Gu, Z. Lu, H. Li, and V. O. K. Li, “Incorporating copying mechanism in sequence-to-sequence learning,” vol. 1, pp. 1631–1640, 2016.

[1] https://www.thepaper.cn/

[17] H. Guo, R. Pasunuru, and M. Bansal, "Soft layer-specific multi-task summarization with entailment and question generation," *arXiv: Computation and Language*, 2018.

[18] W. Hsu, C. Lin, M. Lee, K. Min, J. Tang, and M. Sun, "A unified model for extractive and abstractive summarization using inconsistency loss," vol. 1, pp. 132–141, 2018.

[19] P. Kouris, G. Alexandridis, and A. Stafylopatis, "Abstractive text summarization based on deep learning and semantic content generalization," pp. 5082–5092, 2019.

[20] L. Lebanoff, K. Song, F. Dernoncourt, D. S. Kim, S. Kim, W. Chang, and F. Liu, "Scoring sentence singletons and pairs for abstractive summarization," *arXiv: Computation and Language*, 2019.

[21] J. M. Kupiec, J. O. Pedersen, and F. R. Chen, "A trainable document summarizer," pp. 68–73, 1995.

[22] K. Woodsend and M. Lapata, "Automatic generation of story highlights," pp. 565–574, 2010.

[23] H. Lin and V. Ng, "Abstractive summarization: A survey of the state of the art," vol. 33, pp. 9815–9822, 2019.

[24] J. Lin, X. Sun, S. Ma, and Q. Su, "Global encoding for abstractive summarization," *arXiv: Computation and Language*, 2018.

[25] L. Liu, Y. Lu, M. Yang, Q. Qu, J. Zhu, and H. Li, "Generative adversarial network for abstractive text summarization," *arXiv: Computation and Language*, 2017.

[26] J. M. Conroy and D. P. Oleary, "Text summarization via hidden markov models," pp. 406–407, 2001.

[27] Y. Liu and M. Lapata, "Hierarchical transformers for multi-document summarization," pp. 5070–5081, 2019.

[28] ——, "Text summarization with pretrained encoders," *arXiv: Computation and Language*, 2019.

[29] T. Makino, T. Iwakura, H. Takamura, and M. Okumura, "Global optimization under length constraint for neural text summarization," pp. 1039–1048, 2019.

[30] E. Moroshko, G. Feigenblat, H. Roitman, and D. Konopnicki, "An editorial network for enhanced document summarization." *arXiv: Computation and Language*, 2019.

[31] R. Nallapati, B. Zhou, and M. Ma, "Classify or select: Neural architectures for extractive document summarization," *arXiv: Computation and Language*, 2017.

[32] S. Narayan, S. B. Cohen, and M. Lapata, "Ranking sentences for extractive summarization with reinforcement learning," vol. 1, pp. 1747–1759, 2018.

[33] M. Ott, S. Edunov, A. Baevski, A. Fan, S. Gross, N. Ng, D. Grangier, and M. Auli, "fairseq: A fast, extensible toolkit for sequence modeling," pp. 48–53, 2019.

[34] T. Oya, Y. Mehdad, G. Carenini, and R. T. Ng, "A template-based abstractive meeting summarization: Leveraging summary and source text relationships," pp. 45–53, 2014.

[35] Q. Zhou, N. Yang, F. Wei, S. Huang, M. Zhou, and T. Zhao, "Neural document summarization by jointly learning to score and select sentences," vol. 1, pp. 654–663, 2018.

[36] A. M. Rush, S. Chopra, and J. Weston, "A neural attention model for abstractive sentence summarization," *arXiv: Computation and Language*, 2015.

[37] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," *arXiv: Computation and Language*, 2015.

[38] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer networks," pp. 2692–2700, 2015.

[39] K. Yao, L. Zhang, T. Luo, and Y. Wu, "Deep reinforcement learning for extractive document summarization," *Neurocomputing*, vol. 284, pp. 52–62, 2018.

[40] H. Zhang, Y. Gong, Y. Yan, N. Duan, J. Xu, J. Wang, M. Gong, and M. Zhou, "Pretraining-based natural language generation for text summarization." *arXiv: Computation and Language*, 2019.

[41] X. Zhang, M. Lapata, F. Wei, and M. Zhou, "Neural latent extractive document summarization," pp. 779–784, 2018.

[42] X. Zhang, F. Wei, and M. Zhou, "Hibert: Document level pre-training of hierarchical bidirectional transformers for document summarization," pp. 5059–5069, 2019.

[43] K. M. Hermann, T. Kociský, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom, "Teaching machines to read and comprehend," *arXiv: Computation and Language*, 2015.

[44] K. Lopyrev, "Generating news headlines with recurrent neural networks," *arXiv: Computation and Language*, 2015.

[45] Y. Chen and M. Bansal, "Fast abstractive summarization with reinforce-selected sentence rewriting," *arXiv: Computation and Language*, 2018.

[46] Y. You, W. Jia, T. Liu, and W. Yang, "Improving abstractive document summarization with salient information modeling," pp. 2132–2141, 2019.

[47] S. Takase, J. Suzuki, N. Okazaki, T. Hirao, and M. Nagata, "Neural headline generation on abstract meaning representation." pp. 1054–1059, 2016.

[48] C.-Y. Lin, "Rouge: A package for automatic evaluation of summaries," 2004.

[49] C. Gulcehre, S. Ahn, R. Nallapati, B. Zhou, and Y. Bengio, "Pointing the unknown words," *arXiv: Computation and Language*, 2016.