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
{liao pengcheng, 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 baselines 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 Bahdanau 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
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:

\[ e^t_i = v^T \tanh(w_s s_i + w_h h_i + b^t_i) \]  

\[ \text{Attention}^t = \text{softmax}(e^t) \]

Where \(h_i\) is the encoder hidden states in \(i\)th word, \(s_i\) is decoder hidden states at time step \(t\). \(v, w_s, w_h\) and \(b^t_i\) are learnable parameters. \(\text{Attention}^t\) 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 \(V\).

\[ \text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^\top}{\sqrt{d_k}})V \]

where \(\top\) is transpose function, \(Q \in \mathbb{R}^{n \times d_k}\), \(K \in \mathbb{R}^{m \times d_k}\), \(V \in \mathbb{R}^{m \times d_v}\), \(R\) is the real field, \(n, m\) are the lengths of query and key/value sequences, \(d_k, d_v\) 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:

\[ MH(Q, K, V) = \text{Concat}(hd_1, hd_2, ..., hd_l)w_{mh} \]

where \(hd_i = \text{Attention}(Qw_i^Q, Kw_i^K, Vw_i^V)\), 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:

\[ PE_{(pos, 2i)} = \sin(pos/10000^{2i/d_{model}}) \]

\[ PE_{(pos, 2i+1)} = \cos(pos/10000^{2i/d_{model}}) \]

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 \(E_p = \{p^1, p^2, ..., p^{ld}\}\).

1https://github.com/Pc-liao/Transformer_aggregation
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 semantic information. Then we compute each encoder layer’s multi-head attention sublayer, we can extract different sublayers. We employ a residual connection around each of the sublayers, followed by layer normalization. From the multi-head self-attention, multi-head attention, and feed-forward sublayers. Similar to the encoder, we employ residual connections and layer normalization sublayers to compute final states. The aggregation layer can collect history information to redistribute the encoder’s final hidden states. The aggregation layer can collect history information to redistribute the encoder’s final hidden states.

The aggregation layer can collect history information to redistribute the encoder’s final hidden states.

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:

$$h_{ms}^{(l)} = \text{Norm}(MH^*(Q_{ms}^{(l)}, K_{ms}^{(l)}, V_{ms}^{(l)}) + h_{dl}^{(l-1)})$$

where \(Q_{ms}^{(l)} = K_{ms}^{(l)} = V_{ms}^{(l)} = h_{dl}^{(l-1)}\), \(w_1^{(l)}, w_2^{(l)}\) are learnable parameters. and \(MH^*(\cdot)\) is the position-wise feed-forward sublayer. This sublayer also can be described as two convolution operations with kernel size 1.

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 semantic information. Then we compute each encoder layer’s multi-head attention sublayer, we can extract different sublayers. We employ a residual connection around each of the sublayers, followed by layer normalization. From the multi-head self-attention, multi-head attention, and feed-forward sublayers. Similar to the encoder, we employ residual connections and layer normalization sublayers to compute final states. The aggregation layer can collect history information to redistribute the encoder’s final hidden states.

The aggregation layer can collect history information to redistribute the encoder’s final hidden states.

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:

$$h_{ms}^{(l)} = \text{Norm}(MH^*(Q_{ms}^{(l)}, K_{ms}^{(l)}, V_{ms}^{(l)}) + h_{dl}^{(l-1)})$$

where \(Q_{ms}^{(l)} = K_{ms}^{(l)} = V_{ms}^{(l)} = h_{dl}^{(l-1)}\), \(w_1^{(l)}, w_2^{(l)}\) are learnable parameters. and \(MH^*(\cdot)\) is the position-wise feed-forward sublayer. This sublayer also can be described as two convolution operations with kernel size 1.

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 semantic information. Then we compute each encoder layer’s multi-head attention sublayer, we can extract different sublayers. We employ a residual connection around each of the sublayers, followed by layer normalization. From the multi-head self-attention, multi-head attention, and feed-forward sublayers. Similar to the encoder, we employ residual connections and layer normalization sublayers to compute final states. The aggregation layer can collect history information to redistribute the encoder’s final hidden states.

The aggregation layer can collect history information to redistribute the encoder’s final hidden states.

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:

$$h_{ms}^{(l)} = \text{Norm}(MH^*(Q_{ms}^{(l)}, K_{ms}^{(l)}, V_{ms}^{(l)}) + h_{dl}^{(l-1)})$$

where \(Q_{ms}^{(l)} = K_{ms}^{(l)} = V_{ms}^{(l)} = h_{dl}^{(l-1)}\), \(w_1^{(l)}, w_2^{(l)}\) are learnable parameters. and \(MH^*(\cdot)\) is the position-wise feed-forward sublayer. This sublayer also can be described as two convolution operations with kernel size 1.

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 semantic information. Then we compute each encoder layer’s multi-head attention sublayer, we can extract different sublayers. We employ a residual connection around each of the sublayers, followed by layer normalization. From the multi-head self-attention, multi-head attention, and feed-forward sublayers. Similar to the encoder, we employ residual connections and layer normalization sublayers to compute final states. The aggregation layer can collect history information to redistribute the encoder’s final hidden states.

The aggregation layer can collect history information to redistribute the encoder’s final hidden states.

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:

$$h_{ms}^{(l)} = \text{Norm}(MH^*(Q_{ms}^{(l)}, K_{ms}^{(l)}, V_{ms}^{(l)}) + h_{dl}^{(l-1)})$$

where \(Q_{ms}^{(l)} = K_{ms}^{(l)} = V_{ms}^{(l)} = h_{dl}^{(l-1)}\), \(w_1^{(l)}, w_2^{(l)}\) are learnable parameters. and \(MH^*(\cdot)\) is the position-wise feed-forward sublayer. This sublayer also can be described as two convolution operations with kernel size 1.

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 semantic information. Then we compute each encoder layer’s multi-head attention sublayer, we can extract different sublayers. We employ a residual connection around each of the sublayers, followed by layer normalization. From the multi-head self-attention, multi-head attention, and feed-forward sublayers. Similar to the encoder, we employ residual connections and layer normalization sublayers to compute final states. The aggregation layer can collect history information to redistribute the encoder’s final hidden states.

The aggregation layer can collect history information to redistribute the encoder’s final hidden states.

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:

$$h_{ms}^{(l)} = \text{Norm}(MH^*(Q_{ms}^{(l)}, K_{ms}^{(l)}, V_{ms}^{(l)}) + h_{dl}^{(l-1)})$$

where \(Q_{ms}^{(l)} = K_{ms}^{(l)} = V_{ms}^{(l)} = h_{dl}^{(l-1)}\), \(w_1^{(l)}, w_2^{(l)}\) are learnable parameters. and \(MH^*(\cdot)\) is the position-wise feed-forward sublayer. This sublayer also can be described as two convolution operations with kernel size 1.

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 semantic information. Then we compute each encoder layer’s multi-head attention sublayer, we can extract different sublayers. We employ a residual connection around each of the sublayers, followed by layer normalization. From the multi-head self-attention, multi-head attention, and feed-forward sublayers. Similar to the encoder, we employ residual connections and layer normalization sublayers to compute final states. The aggregation layer can collect history information to redistribute the encoder’s final hidden states.

The aggregation layer can collect history information to redistribute the encoder’s final hidden states.

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:

$$h_{ms}^{(l)} = \text{Norm}(MH^*(Q_{ms}^{(l)}, K_{ms}^{(l)}, V_{ms}^{(l)}) + h_{dl}^{(l-1)})$$

where \(Q_{ms}^{(l)} = K_{ms}^{(l)} = V_{ms}^{(l)} = h_{dl}^{(l-1)}\), \(w_1^{(l)}, w_2^{(l)}\) are learnable parameters. and \(MH^*(\cdot)\) is the position-wise feed-forward sublayer. This sublayer also can be described as two convolution operations with kernel size 1.
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:

$$P_{gen} = \sigma(w_{dl}h_{el}^N + b_{gen})$$  \hspace{1cm} (11)

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

$$\alpha = softmax(h_{el}^N u^T + b_{copy})$$  \hspace{1cm} (12)

$$P_{copy} = \sum_{l=1}^{ld} \alpha z_i$$  \hspace{1cm} (13)

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

where $u$ is representation of input, $z_i$ is one-hot indicator vector for $u^i$, $P_{copy}$ 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:

$$h^h = w^h(ConvCat(h_{el}^{(N-L)}, ..., h_{el}^{(N-1)})) + b^h$$  \hspace{1cm} (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:

$$h^a = MH(Q^p, K^p, V^p)$$  \hspace{1cm} (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 $l$th history information $h^{(l)}$ can be calculated as follows:

$$h^{(l)} = MH(Q^{p(l)}, K^{p(l)}, V^{p(l)})$$  \hspace{1cm} (17)

where $l \in [N - L, N]$ is index of selected encoder layers, $Q^{p(l)}$ is previous history state $h^{(l-1)}$ 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:

$$J(\theta) = \sum_{i=1}^{N} - \log p(Y|X; \theta)$$  \hspace{1cm} (18)

where $\theta$ 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 $t$:

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

where $p(y_t|y_1, y_2, ... y_{t-1}, X; \theta)$ can be calculated in decoder 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/DailyMail website. We choose the non-anonymized version[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 pyrouge² 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(m3) 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 baseline 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-

1https://github.com/abisee/cnn-dailymail
2https://pypi.org/project/pyrouge/
and non-repetitive words in summaries. The aggregation mechanism makes our model generate salient information to enhance the model memory capacity. Therefore, the aggregation mechanism that can review the primitive unnecessary words generated in summaries. Our model uses which will lead to missing some salient information and it can not remind the information far from its current states of salient information marked with red in the source text.

The first exploration Encoder/Decoder Layers Analysis

Our Dataset(summarization)

<table>
<thead>
<tr>
<th>max-token-len(en/de)</th>
<th>1602/239</th>
</tr>
</thead>
<tbody>
<tr>
<td>avg-token-len(en/de)</td>
<td>728/61</td>
</tr>
</tbody>
</table>

The second exploration Aggregation mechanism Analysis The second exploration experiment consists of our baseline model and our Transformer model with aggregation mechanism. The best scores are bolded.

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 layers decreasing experiments. Meanwhile, we can get a lower precision but higher recall score when the encoder layers are decreasing and we have opposite results on the decoder layers decreasing experiments. 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): The national grid has revealed the UK’s first new pylon for nearly 90 years. The t-pylon, designed by Sir Reginald Blomfield, was erected in July 1928 near Edinburgh. It is a third shorter than the old lattice pylons but 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.

Ground Truth: national grid has revealed the UK’s first new pylon for nearly 90 years. The t-pylon it is a third shorter than the old lattice pylons. 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.

Transformer Baseline: the t-pylon -lrb- artist’s shown -rrb- it is a third shorter than the old lattice pylons. 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. The t-pylon it is a third shorter than the old lattice pylons. 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.

Ground Truth: national grid has revealed the UK’s first new pylon for nearly 90 years. The t-pylon it is a third shorter than the old lattice pylons. 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.

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.

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

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

<table>
<thead>
<tr>
<th>Model</th>
<th>Salient</th>
<th>Fluency</th>
<th>Non-Repeated</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pointer Generator</td>
<td>3.37</td>
<td>3.12</td>
<td>3.17</td>
</tr>
<tr>
<td>Pointer Generator + Coverage</td>
<td>3.42</td>
<td>3.23</td>
<td>3.61</td>
</tr>
<tr>
<td>Transformer</td>
<td>3.56</td>
<td>3.30</td>
<td>3.67</td>
</tr>
<tr>
<td>Transformer + Aggregation</td>
<td>3.87</td>
<td>3.37</td>
<td>3.78</td>
</tr>
</tbody>
</table>

Table VI

<table>
<thead>
<tr>
<th>Model</th>
<th>ROUGE-1</th>
<th>ROUGE-2</th>
<th>ROUGE-L</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lead-3</td>
<td>54.09</td>
<td>42.46</td>
<td>34.56</td>
</tr>
<tr>
<td>Pointer Generator</td>
<td>55.49</td>
<td>43.59</td>
<td>48.03</td>
</tr>
<tr>
<td>Pointer Generator + Coverage</td>
<td>55.64</td>
<td>43.80</td>
<td>48.08</td>
</tr>
<tr>
<td>Transformer</td>
<td>52.69</td>
<td>39.86</td>
<td>43.66</td>
</tr>
<tr>
<td>Transformer + Aggregation</td>
<td>58.00</td>
<td>44.42</td>
<td>48.85</td>
</tr>
</tbody>
</table>

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 a higher score compared to the Transformer baseline model. 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.

C. Our Chinese Experiments

We build our Chinese summarization dataset via crawling news website 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 a higher score compared to the Transformer baseline model. 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


1https://www.thepaper.cn/