A Combined Model for Extractive and Abstractive Summarization Based on Transformer Model

Xin Liu, Liutong Xu
The Academy of Computer Science and Technology, Beijing University of Post and Telecommunication
Beijing, China
liuxiaoxin@bupt.edu.cn, xliutong@bupt.edu.cn

Abstract—Summary generates by summarizing automatically main information from the critical sentences of the article. The traditional method of generating text summarization uses extractive or abstractive algorithm model built based on neural attention sequence to sequence framework. This kind of model has performance bug and weak parallel computing capability when getting summary, which causes the summary doesn’t fit the meaning of the original and has no smooth sentences. Therefore, we put up with a joint summary generation model based on improving transformer. This model can put attention on sentence and provide sequence information for periodical transformer model by recurrent neural network. On the other hand, in the generation stage, Transformer model is used to learn the long distance dependence between words, and the summary statement is more consistent with the original meaning by adding pointer mechanism and consistency loss function. Experiments were carried out on three datasets and a manual evaluation was added to verify that the model has good summary significance.

Index Terms—Text Summarization , Transformer, Sequence Information, Joint Model

I. INTRODUCTION

Summary generation aims to get a simplified input text representation to capture the core meaning of the original content. There are two types of methods: extractive and abstractive. Extractive methods usually selects the original sentence or word [1], such as Lead3, Summarunner [2], SwapNet [3] model. The summary obtained by these models are not smooth due to the lack of connectives. Abstractive methods can generate new words and phrases that are not included in the source text. But the summary has incorrect fact details and duplicate information, and words that are out of vocabulary. In recent years, the pointer generator model proposed by see et al [4] which has the ability to extract words from the original text and reduce the repetition rate. Hsu et al. [5] proposed the inconsistent loss function which combines extractive methods and abstractive methods.

The transformer model proposed by Ashish et al. [9] is effective for capturing the global context semantic relationship and parallel computing. In this paper, we proposed TP-EABS (Transformer added Pointer and combine the Extractive and Abstractive methods) model. It adopted the advantages of two types of algorithms and transformer model. The model uses the hidden layer information of GRU to supplement the sequence information of transformer position, and dynamically adjusts the attention of words in the second phase through sentence level attention, so as to reduce the probability of words in sentences with lower weight appearing in the abstract. And we add a pointer mechanism to the transformer model, which enables the transformer to copy words from the original text.

In summary, our contributions are as follows:
• We propose a joint model based on improved Transformer.
• We improve the Transformer architecture by adding position information and pointer mechanisms.
• We have made comparative experiments on CNN/Daily, Papers and DUC-2004 datasets, and the results have been improved.

II. RELATED WORK

In recent years, summary generation has been widely studied. Generally, in the extraction method, key sentences or words in the original text are extracted and presented as abstracts. [3] and [4], [6] used recurrent neural network to code the text, and then mark whether the sentence or word belongs to the summary statement. Although some extraction methods [10] can get high Rouge scores, their readability is very low.

Abstractive methods are mostly based on the sequence to sequence framework based on neural attention [11], [14]. [12] proposed a new model, which first selects key sentences, then rewrites them with abstract algorithm to generate a brief summary of the text. [11] proposed a new model, which can not only retain the ability to generate new words, but also copy words from the original text accurately to reproduce information, and reduce the repetition rate of the generated words in the summary by updating the attention weight. [9] proposed Transformer, which is completely based on attention mechanism and eliminates recursion and convolution. It can solve the problem of long-distance dependence and realize parallel computing. It can obtain the text semantic information and structural information better.

III. OUR MODEL

This chapter introduces three aspects: sentence extraction model, generating model, dynamic word-level mechanism. Fig. 1 gives the overview of TP-EABS model.
A. Sentence Extraction Model

The model input is a series of sentences $S = [s_1, s_2, ..., s_m]$, where the representative $m$ is the $m$th sentence and $s_i$ is the $i$th sentence, expressed as $s_i = [w_1, w_2, ..., w_n]$, the input sentence maps each word to a vector through the language training model, and the $i$th sentence is expressed as $s_i = [w_1, w_2, ..., w_n]$, where $n$ represents the nth word embedding vector. We use BiGRU (Bi-directional Gated Recurrent Unit) to process the input word sequence. After reading the words of the sentence, we update its word representation $x_j^{i} = [\overrightarrow{h}_j^i; \overleftarrow{h}_j^i]$. We use matrix $X$ to represent the input vector. Get the sentence vector by summing the word vectors in the sentence. Then the sentence vector is input to the second layer of BiGRU, and then calculated by the sigmoid function to obtain sentence-level weight $\beta_n$. The extractor loss is calculated using the following cross-entropy loss.

$$L_{ext} = -\frac{1}{n} \sum_{n=1}^{N} (g_n \log \beta_n + (1 - g_n) \log (1 - \beta_n))$$ (1)

In the above formula, $g_n$ is the n-th sentence a summary, the value belongs to 1 and the value does not belong to 0. To get ground truth labels $g = \{g_n\}_n$. We use the unsupervised method proposed by Nillan et al. [5] to get extracted labels.

B. Generating Model

The benchmark model in this paper uses Transformer. We will extract the vector $Z = [z_1, z_2, ..., z_m, ...]$ of the first layer GRU encoding output in the model as the input of the Transformer layer, we think $z_i$ contains word information and location information.

$$[Q; K; V] = W_Z \cdot Z + B_Z$$ (2)

$$Attention (Q, K, V) = softmax \left( \frac{QK^T}{\sqrt{d_k}} \right) V$$ (3)

$$MultiHead (Q, K, V) = Concat (head_1, ..., head_h) W_O$$ (4)

$$Attention (QW_Q^i, KW_K^i, VW_V^i)$$ (5)

We represent the output of the Encoder layer as $E$ and denote the output of the Decoder layer as $D$. We calculate the attention weight between the encoder and decoder and take it out to calculate the probability $p_{gen}$, which determines whether the word is copied from the original text or generated in the dictionary. Enter the last layer of the decoder into softmax to get the probability of getting words from the vocabulary $p_{vocab}$. In the end, we use Beam-search with the Beam-size set to 3.

$$P_{vocab} = softmax(V' (V \cdot D_4 + b) + b')$$ (6)

$$[K_E; V_E] = W_E \cdot E + B_E$$ (7)

$$Q_d = W_d \cdot D_1 + B_d$$ (8)

$$C_t = MulAttn (Q_d, K_E, V_E)$$ (9)

We use the following formula to calculate $p_{gen}$, where $X$ is the input vector and $C$ is the context text. The pointer
generator network is a hybrid between our baseline and the pointer network, because it can both copy words by pointing and generate words from a fixed vocabulary.

\[
p_{gen} = \sigma(W_d^T E + W_{d}^T D + W_{C}^T C_t + b_{par})
\]  

(10)

Next, we calculate the probability of the final word through the probability \( p_{vocab} \) and \( p_{gen} \), where \( C_t \) is the output of the encoder and decoder mutual attention matrix at time \( t \).

\[
P(w) = p_{gen}P_{vocab} + (1 - p_{gen})C_t
\]  

(11)

\( loss_t \) represents the loss function at time \( t \), and \( L_{abs} \) represents the loss function of the generated model part.

\[
loss_t = -\log P(W_t^*)
\]  

(12)

\[
L_{abs} = \frac{1}{T} \sum_{t=0}^{T} loss_t
\]  

(13)

C. Dynamic word-level mechanism

The dynamic word-level mechanism is to reduce the word-level attention through the sentence’s attention weight, so that the generative summary can pay more attention to a certain sentence to generate a word, which also makes the sentence weights of the same information different, which reduces the repeatability of the generated words to a certain probability.

This article uses a BiGRU to obtain sentence-level weights in the sentence extraction model. It needs to be added to the word-level weights. The obtained sentence-level weights are the matrix \( A_{sent} \) formed by \( \beta \), and a sentence vector \( E_4^t \) obtained by multiplying the word-level attention matrix and value according to the multi-head attention mechanism, and then obtain the updated encoding matrix using the following formula.

\[
E_4^t = (W_{sent} A_{sent} + B_{sent})E_4^t + B_E
\]  

(14)

In order to ensure that the two levels of attention can be kept consistent during the training process, a unified loss function is added here. We use the \( L_{sw} \) loss function to represent the error function calculated between sentences and words. Where \( m(n) \) is a mapping relationship between words and sentences.

\[
L_{sw} = -\frac{1}{T} \sum_{t=1}^{T} \log(\frac{1}{K}) \sum_{n \in K} E_n^t \times A_{m(n)}
\]  

(15)

Where \( K \) is the set of the first \( K \) participating words and \( t \) is the number of words in the abstract. The loss of inconsistency helps our unified model for end-to-end training benefit both the extractor and the abstractor, and also helps to generate longer digest lengths. Through sentence-level extraction, an improved Transformer generation layer, and a swap mechanism, we finally generate a training loss function \( L_{sum} \)

\[
L_{sum} = \varepsilon_1 L_{ext} + \varepsilon_2 L_{abs} + \varepsilon_3 L_{sw}
\]  

(16)

where \( \varepsilon_1, \varepsilon_2, \varepsilon_3 \) are hyper-parameters. In our experiment, we give \( L_{ext} \) a bigger weight (e.g., \( \varepsilon_1 = 5 \)) when end-to-end training with \( L_{sw} \), since we found that \( L_{sw} \) is relatively large such that the extractor tends to ignore \( L_{ext} \).
TABLE I: ROUGE F1 results for various models and ablations on the CNN/Daily Mail test set.

<table>
<thead>
<tr>
<th>Model</th>
<th>ROUGE-1</th>
<th>ROUGE-2</th>
<th>ROUGE-L</th>
</tr>
</thead>
<tbody>
<tr>
<td>AttS2S [6]</td>
<td>32.75</td>
<td>12.21</td>
<td>29.01</td>
</tr>
<tr>
<td>PGen [4]</td>
<td>36.44</td>
<td>15.66</td>
<td>33.42</td>
</tr>
<tr>
<td>Key information [10]</td>
<td>38.95</td>
<td>17.12</td>
<td>35.68</td>
</tr>
<tr>
<td>Transformer</td>
<td>35.26</td>
<td>14.12</td>
<td>31.08</td>
</tr>
<tr>
<td>TPABS</td>
<td>36.75</td>
<td>15.89</td>
<td>33.15</td>
</tr>
<tr>
<td>TP-EABS(MLP)</td>
<td>38.12</td>
<td>17.36</td>
<td>36.45</td>
</tr>
<tr>
<td>TP-EABS(ADD)</td>
<td>38.14</td>
<td>17.29</td>
<td>36.52</td>
</tr>
</tbody>
</table>

TABLE II: Head-to-head comparison between test set outputs of PGen and TP-EABS. Analyses done on summaries for Papers.

<table>
<thead>
<tr>
<th>Model</th>
<th>PGen</th>
<th>same</th>
<th>TP-EABS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-redundancy</td>
<td>65</td>
<td>62</td>
<td>182</td>
</tr>
<tr>
<td>Coherence</td>
<td>180</td>
<td>42</td>
<td>145</td>
</tr>
<tr>
<td>Focus</td>
<td>140</td>
<td>36</td>
<td>176</td>
</tr>
<tr>
<td>Overall</td>
<td>160</td>
<td>41</td>
<td>170</td>
</tr>
</tbody>
</table>

C. Papers and DUC-2004

In Papers data set, the length of the article is long, and its summary is mostly not in the original text. We mainly test our improved Transformer model, as shown in Table III. After adding the pointer mechanism, the model obtains The result is 2 percentage points higher than the Pgen model, which proves that the Transformer model is richer in obtaining context information than the recurrent neural network in the language model. On this dataset, we set the parameters of Pgen from See et al[4]. We did ablation experiments to evaluate the contributions of different mechanisms Transformer, TEABS (Transformer on the extraction model), TPABS (Transformer with a pointer mechanism added), and TP-EABS. The experimental results of the four models are shown in the Table III.

As shown in Table III, our method has made some progress on the current benchmark on DUC-2004 dataset, and ROUGE-1 and ROUGE-L scores have improved the RAS-LSTM model by absolute 0.3 and 1.5 percentage points, respectively. We also compare the model with Feats. With Feats, we can see that our model still performs better without introducing external information and reinforcement learning. TP-EABS improves the data set by 0.13 percentage points and 0.4 percentage points over TPABS. Considering the sequence context information, our model can capture important information and generate high-quality abstracts.

V. Conclusion

We propose a joint abstract generation model based on improved Transformer. Most importantly, we improved the Transformer model so that it has the ability to copy words from the original text. After adding sequence information and extraction stages, the model in this paper can obtain more complete summary information in the uniformly distributed original text, and it will not ignore its importance because the key information is located later. Through end-to-end training of our model, we conducted experiments on three datasets and conducted reliable human evaluation on private datasets, proving that the model has good summary information significance.

REFERENCES