Memory-Enhanced Abstractive Summarization

Zepeng Hao1,*, Taihua Shao1, Shengwei Zhou1 and Honghui Chen1

1 Science and Technology on Information Systems Engineering Laboratory, National University of Defense Technology, Changsha, China, 410073.

*Email: hao3184045@gmail.com

Abstract. The RNN network has been widely investigated in automatic abstractive summarization and has achieved good results in previous studies. However, in the process of processing and storing information in the RNN structure, the problem of losing long-term information may occur, resulting in the inability to generate high-quality summaries containing comprehensive information about the corresponding documents. In this paper, in order to overcome this problem as well as enhance the global information, we propose a memory-enhanced abstractive summarization (MEAS) model consisting of a memory enhancement module and a Seq2Seq module. Our model is able to capture and store global information about the entire document, such as the relationship between sentences and sentences, resulting in a richer representation of information to the Seq2Seq module to generate higher quality summaries. Our experimental results indicate that on the CNN/DailyMail corpus, our MEAS model achieve improvements of up to 1.17, 0.27 and 0.85 on the R-1, R-2, and R-L score, respectively, when compared with the related state-of-the-art baseline.

1. Introduction

With the development of the network and the development of the times, more and more people use the neural network to collect information and obtain information, which makes the field of automatic summarization greatly developed. Automatic summarization allows people to get and understand the important content and deep meaning of the article quickly. As shown in Figure 1, the automatic summarization tasks generally has three types of classification methods.

![Figure 1. Classification of summary tasks.](image-url)
Abstractive summarization is a task of generating summaries based on the underlying meaning of documents. The benefit of the abstractive summarization over the extractive summarization is the ability to use a more accurate and readable paragraph and sentence to form summaries rather than statements that is confined to the original document. In the abstractive summarization task, most approaches are based on the sequence to sequence architecture. The most important module is the recurrent neural networks [1][2]. This model can capture the information in the document and generate more accurate and readable sentences as a summary. However, the model memory storage based on the RNN structure is too small to record the entire document accurately and comprehensively, especially for longer documents. This results in the lack of information, so that the generated summary cannot contain all the important meanings and potential meanings of the document. In order to overcome this problem, the model of memory network is proposed in the article of [3]. This model can better capture and store features in long documents by training how to update and store the memory [4].

In our paper, we propose a hybrid network architecture for abstractive summarization named memory-enhanced abstractive summarization (MEAS), which consists of the memory enhancement module and the Seq2Seq module, to improve performance in abstractive summarization. This model consists of two parts, the memory enhancement module and the Seq2Seq module. In the memory enhancement module, the MEAS mainly reads related documents, performs word embedding and sentence embedding, and processes the information to constitute the memory of this document. And then, in the Seq2Seq module, MEAS reads all the sentences embedding in this document and the memory of this document, and then generates sentences as a summary. This structure captures and store more potential information about the document, so it ensures that there is enough information to generate richer document representations and a richer and more accurate summary.

We test and evaluate the MEAS on the CNN/DailyMail datasets [5]. Our experimental results show that the performance of the MEAS is better than the baselines in the ROUGE evaluation score. Compared with the related state-of-the-art baseline, MEAS is 1.17, 0.27 and 0.85 higher on R-1, R-2 and R-L, respectively.

In this paper, our main contributions are as follows:

- We propose a hybrid network architecture for abstractive summarization called memory-enhanced abstractive summarization (MEAS);
- We apply the memory network to store the feature of a processed document, which can make the stored features more abundant, can better capture the relationship between the sentences in the document and the features of sentences, which is beneficial to generate more accurate and readable sentences to form a summary;
- We test the performance of the MEAS model with fixed summaries of different lengths and the results demonstrate that the MEAS have a better effect on generating longer summaries.

2. Approach
Firstly, we make a definition of the abstractive summarization task study. Given a document D, is a sequence of N word vectors, \( z_1, z_2, \ldots, z_n \) from a fixed vocabulary \( V \), which through a neural network structure to generate another sequence of length \( M < N \) word vectors also from a same vocabulary \( V \), but can roughly reflect the connotation of the document and have certain readability as a summary. Unlike some other models, we won’t assume \( M \) is fixed to allow for different length summarizations.

In the process of supervised training, we aim to optimize:

\[
\arg\max_{y \in V} s(z, y; \theta)
\]

where \( s \) is some scoring function and \( y \) is the total vocabulary and model parameters \( \theta \). This differs from extractive models [6] that optimize:

\[
\arg\max_{n \in 1, 2, \ldots, N^M} s(z, z_{[n_1, n_2, \ldots, n_M]}; \theta)
\]

2
which extracts words from the input document to use as the summary. Firstly, we apply the Word2vec approach to obtain the representation of words in the document \(D\). Then, in order to get a better sentence embedded representation, including the relationship between the words inside the sentence, we apply the bidirectional LSTM with max pooling network to obtain the sentence embedded representations \([7]\). Given a sequence of words of a sentence \((z_1, z_2, \ldots, z_R)\), a bidirectional LSTM computes a set of \(R\) vectors \((h_1, h_2, \ldots, h_R)\). For all of them \(h_r\) are composed of the forward LSTM as well as backward LSTM, which are obtained by reading sentences from two opposite directions:

\[
\overrightarrow{h_r} = \text{LSTM}_f(z_1, z_2, \ldots, z_R)
\]

\[
\overleftarrow{h_r} = \text{LSTM}_b(z_1, z_2, \ldots, z_R)
\]

\[
h_r = [\overrightarrow{h_r}, \overleftarrow{h_r}]
\]

Then we combine the different numbers of \(\{h_r\}_r\) by selecting the maximum value on each dimension of the hidden unit to form a fixed-size vector \(x_i\) as an embedded representation of a sentence. As shown in Figure 2, we will illustrate our MEAS model from two modules (In the figure is the memory enhancement module on the left side and the Seq2Seq module on the right side).

\[I_i = \tanh(W_i x_i + b_i)\]  

2.1. Memory Enhancement module
The memory enhancement module consists of following three parts: memory input, memory slots and memory generalization.

2.1.1. Memory Input. Memory Generalization is a sentence-level process, so the entire set of sentence embeddings \(x = (x_1, x_2, \ldots, x_n)\) are converted into a feature representation \(I = (I_1, I_2, \ldots, I_n)\) by feeding them to a Dense layer:
In the above formula, \( W_f \) and \( b_l \) are the embedding matrix and bias term, respectively. At the same time, the sentence embeddings of the document \( x = (x_1, x_2, \ldots, x_n) \) are used as input and through another Dense layer to obtain another feature representation \( T = (T_1, T_2, \ldots, T_n) \):

\[
T_i = \tanh(W_T x_i + b_t)
\]  

(7)

In the above formula, \( W_T \) and \( b_t \) is another embedding matrix and bias term, respectively.

2.1.2. **Memory Slots.** The memory slot \( m \) is a special vector, which is consist of a sequence of objects. It can be represented \( m = (m_1, m_2, \ldots, m_k) \).

2.1.3. **Memory Generalization.** This module is based on new input information, by interacting with the memory slot to determine how to add, update or delete memory. Firstly, we store the input sequence \( x = (x_1, x_2, \ldots, x_n) \) into the memory slot as the initial memory \( m = (m_1, m_2, \ldots, m_n) \) by using an embedding matrix \( A \) to embed each \( x_i \) to the contiguous space to convert the sentence representation to the initial memory representation \( m_i \). Next, we match the sentence feature representation \( T \) and the initial memory \( m \) by calculating the dot product to get \( P \):

\[
P = mT^T
\]  

(8)

Then the dot product \( P \) is fed to a one-layer of softmax to obtain a probability matrix \( p \) over the inputs:

\[
p = \text{softmax}(P)
\]  

(9)

Finally, we can obtain the memory representation vector \( o \) by generalization. The memory representation vector \( o \) is a product of the input feature representation \( I \) with the probability matrix \( p \) from the input:

\[
o = pl
\]  

(10)

2.2. **Seq2Seq module**

Through the first module, we obtain the memory representation vector that belongs to a document, i.e., \( o \). In order to overcome the problem of long-term loss of RNN in the Seq2Seq module, we use the memory of this document to enhance the memory to ensure a more accurate and comprehensive summary is generated. Next, we will introduce the seq2seq module combined with the document memory in two components, i.e., encoder and decoder.

2.2.1. **Encoder.** We decide to enrich the input of the encoder, especially for the richness of the global information to overcome the problem of long-term information loss in RNN. Thus, we add the global information \( o \) obtained by the memory network to the input of the encoder, so that the obtained output of the encoder is more reasonable, and a better summary is generated in the subsequent process. The mixed input of the encoder is calculated by the following formula:

\[
x_{mix} = (1 - \gamma)x + \gamma o
\]  

(11)

where \( \gamma \) is a learnable parameter. Then in the encoder, the embeddings representation of tokens in the mixed input \( z_i \) are fed into a bi-directional LSTM, which concatenates the hidden states at each time step from both the previous and next word, capturing semantics from both sides of each word, producing a sequence of hidden states \( h_i \). The number of neural units in each encoder LSTM module was half of the number of neural units in the output vector, so that in the same number of units the forward and backward state can be concatenated.
2.2.2. Decoder. In the decoder, we feed the embedded representation of the previous word to a LSTM and get the decoder hidden state $s_t$. Then we can obtain the attention distribution $a_t^t$ by the following formula:

$$e_t^t = v^T \tanh(W_h h_i + W_s s_t + b_{attn})$$

$$a_t^t = \text{softmax}(e_t^t)$$

where $v$, $W_h$, $W_s$ and $b_{attn}$ are learnable parameters. The attention distribution is determined by the relationship between the decoder hidden state $s_t$ and the encoder hidden state $h_t$. It can be regarded as a probability distribution of words in the document and guide the range of the source document that the decoder focuses on when generating the next word. Next, we obtain the content vector $c_t$ by use the attention distribution to calculate a weighted sum of the encoder hidden state $h_i$:

$$c_t = \sum_i a_t^i h_i$$

The content vector can be thought of as a representation of the short-term information from the corresponding document before each step of generating the word. We combine the content vector $c_t$ and the decoder hidden state $s_t$ as input and feed the combined vector through two linear layers with the softmax layer to generate the vocabulary probability distribution $P_v$:

$$P_v = \text{softmax}(W'(W[s_t, c_t] + b) + b')$$

where $W$, $W'$, $b$ and $b'$ are learnable parameters. In addition, in order to solve the out-of-vocabulary (OOV) problem, we added the pointer-generator mechanism [10] in the decoder, through the content vector $c_t$ and the decoder hidden state $s_t$ for each time step to get the generation probability $p_g \in [0,1]$:

$$p_g = \sigma(w_c^T c_t + w_s^T s_t + b_p)$$

In the above formula, $w_c$, $w_s$ and $b_p$ are learnable parameters and $\sigma$ is the sigmoid function. The main effect of $p_g$ is to choose between generating a word by sampling from $P_v$, or copying a word from the input sequence by sampling from the attention distribution $a_t^t$. Through $P_v$, which is a probability distribution over the entire vocabulary, and $p_g$ we can get the final probability distribution of the predictive word $t$:

$$P(z) = p_g P_v(z) + (1 - p_g) \sum_{i : z_i = z} a_t^i$$

During training, we use the sum of the negative log likelihood of the target word $z_i^*$ in each time step $t$ in the whole sequence as the total loss:

$$\text{TotalLoss} = \sum_{t=0}^T -\log P(z_i^*)$$

We can update the parameters $\theta$ of the two modules in our model by minimizing the TotalLoss:

$$\min_{\theta} \text{TotalLoss}(\theta)$$

In our model, we use the beam-search, which is used commonly in Seq2Seq models, to overcome the optimization problem in model testing and to improve the accuracy of output. In beam-search, we improve our search for a local optimum by computing the best $K$ potential hypothesis at each time step of the summary. Next each of those hypotheses is used as input for the next time step. Although we can get $K^2$ potential hypothesis, only the top $K$ hypotheses are kept. After the above process, we can get the sequence of generated words as the summary.
3. Experiments

3.1. Model Summary
Our MEAS model is a hybrid network architecture for abstractive summarization that combines the memory enhancement module and the Seq2Seq module. We compare MEAS model with two baselines of abstractive summarization:

- ABS: a standard abstractive summarization model using attentional sequence to sequence network;
- PGC: the state-of-the-art abstractive text summarization model based on the pointer-generator coverage networks.

3.2. Research questions
RQ1: How does our MEAS model perform compared to the baselines?
RQ2: What is the quality of the generated summaries when models generate summaries of different lengths, i.e., 75 bytes vs. 175 bytes vs. full length?

3.3. Evaluation Metric
We evaluate the performance of these models on the ROUGE metrics, where $\text{ROUGE} – 1$ (R-1), $\text{ROUGE} – 2$ (R-2) represent the informativeness of a summary, and $\text{ROUGE} – L$ (R-L) evaluates the readability and fluency of a summary.

3.4. Datasets and Parameters
We train and evaluate abstractive summarization models on the CNN/Daily Mail datasets. This dataset is widely used in abstractive summarization domain. It includes news stories in CNN and Daily Mail websites. Meanwhile, a summary of multiple sentences generated by humans corresponding to each document. This corpus has 287,226 training pairs, 13,368 validation pairs and 11,490 test pairs. For the experiment, we use 128-dimensional word2vec embeddings trained on this dataset as input and 256-dimensional hidden states for the model. For the pointer-generator mechanism, we use a vocabulary of 50k words.

4. Results and Discussion

4.1. Experimental performance of the three models
To answer RQ1, we evaluate the ROUGE scores for all entire summaries generated by our MEAS model and the other two models. The results of the metric scores are shown in Table 1. By observing the comparison between baselines, we can find that PGC has not only a higher R-1 score and R-2 score but also a higher R-L score than ABS. This means that the PGC model can produce more informative, more fluent and more readable summaries than the ABS model. This may be because the pointer-generator mechanism and coverage mechanism in the PGC model overcomes the OOV problem to generate a better summary.

Compared to our MEAS model and baselines, there is a significant improvement in the overall ROUGE scores. Compared with the PGC model R-1, R-2 and R-L are increased by 1.17, 0.27 and 0.85, respectively. The experimental results show that our MEAS model can generate more fluent and more informative summary. It also shows that our MEAS model can capture and store more document information.

| Models | R-1 | R-2 | R-L |
|--------|-----|-----|-----|
| ABS    | 35.46 | 13.30 | 32.65 |
| PGC    | 39.53 | 17.28 | 36.38 |
| MEAS   | 40.77 | 17.55 | 37.23 |
4.2. Effect of the length of summaries on different models

For RQ2, we compare experimental results under conditions that generate summaries of different lengths: 75 bytes vs. 175 bytes vs. full length. The ROUGE scores for the full-length summaries and 75-byte, 175-byte summaries generated by different models are shown in Tables 1, 2 and 3, respectively. As can be seen from the table, our MEAS model is significantly higher in ROUGE score than the other two models for generating full length summaries. For generating 75-byte summaries, our model performs slightly better than the other two baselines, except on the R-2 scores. For generating 175-byte summaries, our model score is higher than the other two models, especially on the R-1 as well as R-L score. As shown in Figure 3, for the MEAS model itself, our model is more suitable for generating long summaries than short summaries. It also proves that our model can better capture and store the features of the document.

Table 2. ROUGE score on CNN dataset when generating summary is 75 bytes. (%)

| Models | R-1    | R-2    | R-L    |
|--------|--------|--------|--------|
| ABS    | 14.85  | 6.77   | 13.96  |
| PGC    | 21.56  | 9.82   | 20.26  |
| MEAS   | 21.74  | 9.82   | 20.38  |

Table 3. ROUGE score on CNN dataset when generating summary is 175 bytes. (%)

| Models | R-1    | R-2    | R-L    |
|--------|--------|--------|--------|
| ABS    | 27.58  | 11.71  | 25.68  |
| PGC    | 32.38  | 13.97  | 29.77  |
| MEAS   | 33.01  | 14.18  | 30.21  |

Figure 3. The performance of the MEAS model in generating different length summaries.

5. Conclusions and Future Work

In this paper, we propose a hybrid network architecture for abstractive summarization called memory-enhanced abstractive summarization (MEAS) that uses a memory enhancement module to store more information and capture the intrinsic connections in the document, resulting in a richer document representation that enables the Seq2Seq module to generate higher quality summary. At the same time, the experimental results demonstrate that the MEAS model is higher than the baselines on the ROUGE scores, and is more suitable for generating long summaries.
For future work, we will combine more information to enrich the representation of documents, such as topics and title, to generate higher quality summaries. At the same time, we will experiment our model on other different datasets and evaluate the generated summaries.

References

[1] Nallapati R, Zhou B, Gulcehre C, et al. Abstractive text summarization using sequence-to-sequence rnns and beyond[J]. arXiv preprint arXiv:1602.06023, 2016.

[2] Kim S, Kang S G. TL; DR: Improving Abstractive Summarization Using LSTMs[J].

[3] J. Weston, S. Chopra, and A. Bordes. Memory networks. In International Conference on Learning Representations (ICLR), 2015.

[4] Sukhbaatar S, Weston J, Fergus R. End-to-end memory networks[C]//Advances in neural information processing systems. 2015: 2440-2448.

[5] Hermann K M, Kocisky T, Grefenstette E, et al. Teaching machines to read and comprehend[C]//Advances in Neural Information Processing Systems. 2015: 1693-1701.

[6] Nallapati R, Zhai F, Zhou B. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents[C]//Thirty-First AAAI Conference on Artificial Intelligence. 2017.

[7] Conneau A, Kiela D, Schwenk H, et al. Supervised learning of universal sentence representations from natural language inference data[J]. arXiv preprint arXiv:1705.02364, 2017.

[8] Rush, A.M., Chopra, S., and Weston, J. A neural attention model for abstractive sentence summarization. In EMNLP, 2015.

[9] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need[C]//Advances in Neural Information Processing Systems. 2017: 5998-6008.

[10] See A, Liu P J, Manning C D. Get to the point: Summarization with pointer-generator networks[J]. arXiv preprint arXiv:1704.04368, 2017.