Iterative Document Representation Learning Towards Summarization with Polishing

Xiuying Chen[1], **Shen Gao**[1], **Chongyang Tao**[1], **Yan Song**[2], **Dongyan Zhao**[1] **and Rui Yan**[1*]
[1]Institute of Computer Science and Technology, Peking University, Beijing, China
[2]Tencent AI Lab
{xy-chen,shengao,chongyangtao,zhaody,ruiyan}@pku.edu.cn
clksong@tencent.com

Abstract

In this paper, we introduce Iterative Text Summarization (ITS), an iteration-based model for supervised extractive text summarization, inspired by the observation that it is often necessary for a human to read an article multiple times in order to fully understand and summarize its contents. Current summarization approaches read through a document only once to generate a document representation, resulting in a sub-optimal representation. To address this issue we introduce a model which iteratively polishes the document representation on many passes through the document. As part of our model, we also introduce a selective reading mechanism that decides more accurately the extent to which each sentence in the model should be updated. Experimental results on the CNN/DailyMail and DUC2002 datasets demonstrate that our model significantly outperforms state-of-the-art extractive systems when evaluated by machines and by humans.

1 Introduction

A summary is a shortened version of a text document which maintains the most important ideas from the original article. Automatic text summarization is a process by which a machine gleans the most important concepts from an article, removing secondary or redundant concepts. Nowadays as there is a growing need for storing and digesting large amounts of textual data, automatic summarization systems have significant usage potential in society.

Extractive summarization is a technique for generating summaries by directly choosing a subset of salient sentences from the original document to constitute the summary. Most efforts made towards extractive summarization either rely on human-engineered features such as sentence length, word position, and frequency (Cohen, 2002; Radev et al., 2004; Woodsend and Lapata, 2010; Yan et al., 2011a,b, 2012) or use neural networks to automatically learn features for sentence selection (Cheng and Lapata, 2016; Nallapati et al., 2016a).

Although existing extractive summarization methods have achieved great success, one limitation they share is that they generate the summary after only one pass through the document. However, in real-world human cognitive processes, people read a document multiple times in order to capture the main ideas. Browsing through the document only once often means the model cannot fully get at the document's main ideas, leading to a subpar summarization. We share two examples of this. (1) Consider the situation where we almost finish reading a long article and forget some main points in the beginning. We are likely to go back and review the part that we forget. (2) To write a good summary, we usually first browse through the document to obtain a general understanding of the article, then perform a more intensive reading to select salient points to include in the summary. In terms of model design, we believe that letting a model read through a document multiple times, polishing and updating its internal representation of the document can lead to better understanding and better summarization.

To achieve this, we design a model that we call Iterative Text Summarization (ITS) consisting of a novel "iteration mechanism" and "selective reading module". ITS is an iterative process, reading through the document many times. There is one encoder, one decoder, and one iterative unit in each iteration. They work together to polish document representation. The final labeling part uses outputs from all iterations to generate summaries. The selective reading module we design is a modi-

*Corresponding author: Rui Yan (ruiyan@pku.edu.cn)

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4088–4097
Brussels, Belgium, October 31 - November 4, 2018.

fied version of a Gated Recurrent Unit (GRU) network, which can decide how much of the hidden state of each sentence should be retained or updated based on its relationship with the document.

Overall, our contribution includes:

1. We propose Iterative Text Summarization (ITS), an iteration based summary generator which uses a sequence classifier to extract salient sentences from documents.

2. We introduce a novel iterative neural network model which repeatedly polishes the distributed representation of document instead of generating that once for all. Besides, we propose a selective reading mechanism, which decides how much information should be updated of each sentence based on its relationship with the polished document representation. Our entire architecture can be trained in an end-to-end fashion.

3. We evaluate our summarization model on representative CNN/DailyMail corpora and benchmark DUC2002 dataset. Experimental results demonstrate that our model outperforms state-of-the-art extractive systems when evaluated automatically and by human.

2 Related Work

Our research builds on previous works in two fields: summarization and iterative modeling.

Text summarization can be classified into extractive summarization and abstractive summarization. Extractive summarization aims to generate a summary by integrating the most salient sentences in the document. Abstractive summarization aims to generate new content that concisely paraphrases the document from scratch.

With the emergence of powerful neural network models for text processing, a vast majority of the literature on document summarization is dedicated to abstractive summarization. These models typically take the form of convolutional neural networks (CNN) or recurrent neural networks (RNN). For example, Rush et al. (2015) propose an encoder-decoder model which uses a local attention mechanism to generate summaries. Nallapati et al. (2016b) further develop this work by addressing problems that had not been adequately solved by the basic architecture, such as keyword modeling and capturing the hierarchy of sentence-to-word structures. In a follow-up work, Nallapati et al. (2017) propose a new summarization model which generates summaries by sampling a topic one sentence at a time, then producing words using an RNN decoder conditioned on the sentence topic. Another related work is by See et al. (2017), where the authors use "pointing" and "coverage" techniques to generate more accurate summaries.

Despite the focus on abstractive summarization, extractive summarization remains an attractive method as it is capable of generating more grammatically and semantically correct summaries. This is the method we follow in this work. In extractive summarization, Cheng and Lapata (2016) propose a general framework for single-document text summarization using a hierarchical article encoder composed with an attention-based extractor. Following this, Nallapati et al. (2016a) propose a simple RNN-based sequence classifier which outperforms or matches the state-of-art models at the time. In another approach, Narayan et al. (2018) use a reinforcement learning method to optimize the Rouge evaluation metric for text summarization. The most recent work on this topic is (Wu and Hu, 2018), where the authors train a reinforced neural extractive summarization model called RNES that captures cross-sentence coherence patterns. Due to the fact that they use a different dataset and have not released their code, we are unable to compare our models with theirs.

The idea of iteration has not been well explored for summarization. One related study is Xiong et al. (2016)'s work on dynamic memory networks, which designs neural networks with memory and attention mechanisms that exhibit certain reasoning capabilities required for question answering. Another related work is (Yan, 2016), where they generate poetry with iterative polishing sn chema. Similiar method can also be applied on couplet generation as in (Yan et al., 2016). We take some inspiration from their work but focus on document summarization. Another related work is (Singh et al., 2017), where the authors present a deep network called Hybrid MemNet for the single document summarization task, using a memory network as the document encoder. Compared to them, we do not borrow the memory network structure but propose a new iterative architecture.

3 Methodology

3.1 Problem Formulation

In this work, we propose Iterative Text Summarization (ITS), an iteration-based supervised model for extractive text summarization. We treat the extractive summarization task as a sequence labeling problem, in which each sentence is visited sequentially and a binary label that determines whether or not it will be included in the final summary is generated.

ITS takes as input a list of sentences $s = \{s_1, \ldots, s_{n_s}\}$, where n_s is the number of sentences in the document. Each sentence s_i is a list of words: $s_i = \{w_1^i, \ldots, w_{n_w}^i\}$, where n_w is the word length of the sentence. The goal of ITS is to generate a score vector $\boldsymbol{y} = \{y_1, \ldots, y_{n_s}\}$ for each sentence, where each score $y_i \in [0, 1]$ denotes the sentence's *extracting probability*, that is, the probability that the corresponding sentence s_i will be extracted to be included in the summary. We train our model in a supervised manner, using a corresponding gold summary written by human experts for each document in training set. We use an unsupervised method to convert the human-written summaries to gold label vector $\boldsymbol{y'} = \{y'_1, ..., y'_{n_s}\}$, where $y'_i \in \{0, 1\}$ denotes whether the i-th sentence is selected (1) or not (0). Next, during training process, the cross entropy loss is calculated between $\boldsymbol{y}$ and $\boldsymbol{y'}$, which is minimized to optimize $\boldsymbol{y}$. Finally, we select three sentences with the highest score according to $\boldsymbol{y}$ to be the extracted summary. We detail our model below.

3.2 Model Architecture

ITS is depicted in Fig.1. It consists of multiple iterations with one encoder, one decoder, and one iteration unit in each iteration. We combine the outputs of decoders in all iterations to generate the extracting probabilities in the final labeling module.

Our encoder is illustrated in the shaded region in the left half of Fig.1. It takes as input all sentences as well as the document representation from the previous unit $\boldsymbol{D_{k-1}}$, processes them through several neural networks, and outputs the final state to the iterative unit module which updates the document representation.

Our decoder takes the form of a bidirectional RNN. It takes the representation of sentence generated by the encoder as input, and its initial state is the polished document representation $\boldsymbol{D_k}$. Our last module, the sentence labeling module, concatenates the hidden states of all decoders together to generate an integrated score for each sentence.

As we apply supervised training, the objective is to maximize the likelihood of all sentence labels $y' = \{y'_1, ..., y'_{n_s}\}$ given the input document s and model parameters θ:

$$\log p(y'|s;\theta) = \sum_{i=1}^{n_s} \log p(y'_i|s;\theta) \qquad (1)$$

4 Our Model

4.1 Encoder

In this subsection, we describe the encoding process of our model. For brevity, we drop the superscript k when focusing on a particular layer. All the $\boldsymbol{W}$'s and $\boldsymbol{b}$'s in this section with different superscripts or subscripts are the parameters to be learned.

Sentence Encoder: Given a discrete set of sentences $s = \{s_1, \ldots, s_{n_s}\}$, we use a word embedding matrix $\boldsymbol{M} \in \mathbb{R}^{V \times D}$ to embed each word w^i in sentence s_i into continuous space $\hat{\boldsymbol{w}}^{\boldsymbol{i}}$, where V is the vocabulary size, D is the dimension of word embedding.

The sentence encoder can be based on a variety of encoding schemes. Simply taking the average of embeddings of words in a sentence will cause too much information loss, while using GRUs or Long Short-Term Memory (LSTM) requires more computational resources and is prone to overfitting. Considering above, we select *positional encoding* described in (Sukhbaatar et al., 2015) as our sentence encoding method. Each sentence representation $\hat{\boldsymbol{s}}_{\boldsymbol{i}}$ is calculated by $\hat{\boldsymbol{s}}_{\boldsymbol{i}} = \sum_{j=1}^{n_w} \boldsymbol{l_j} \circ \hat{\boldsymbol{w}}_{\boldsymbol{j}}^{\boldsymbol{i}}$, where $\circ$ is element-wise multiplication, $\boldsymbol{l_j}$ is a column vector computed as $l_{j,d} = (1 - \frac{j}{n_w}) - (\frac{d}{D})(1 - \frac{2j}{n_w})$, $l_{j,d}$ denotes the d-th dimension of $\boldsymbol{l_j}$.

Note that throughout this study, we use GRUs as our RNN cells since they can alleviate the overfitting problem as confirmed by our experiments. As our selective reading mechanism (which will be explained later) is a modified version of original GRU cell, we give the details of the GRU here. GRU is a gating mechanism in recurrent neural networks, introduced in (Cho et al., 2014). Their performance was found to be similar to that of LSTM cell but using fewer parameters as described in (Hochreiter and Schmidhuber, 1997). The GRU cell consists of an update gate vector

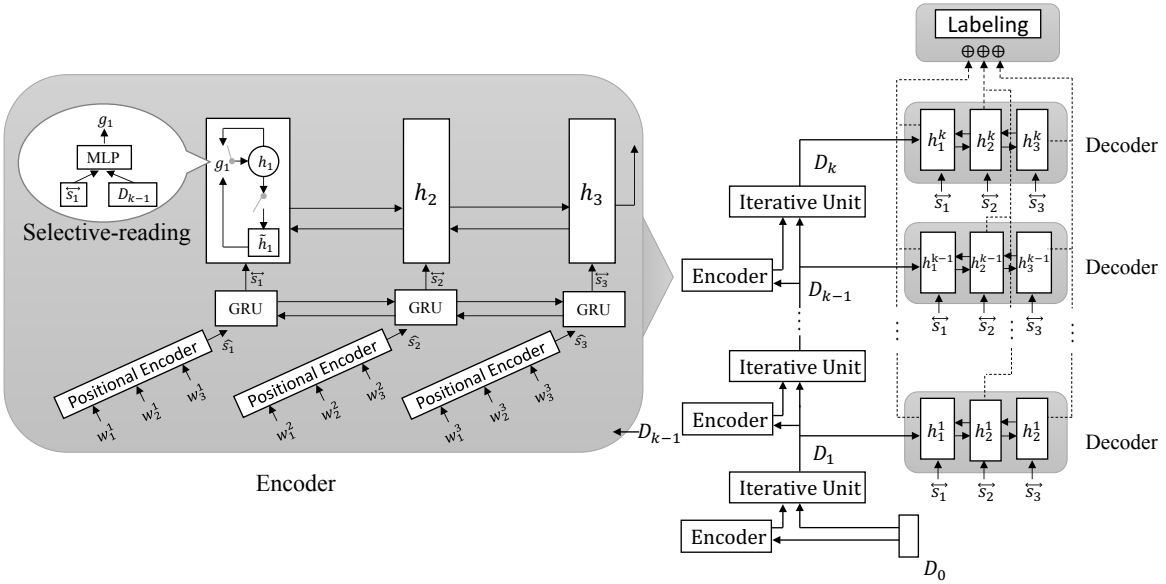


Figure 1: Model Structure: There is one encoder, one decoder and one iterative unit (which is used to polish document representation) in each iteration. The final labeling part is used to generating the extracting probabilities for all sentences combining hidden states of decoders in all iterations. We take a document consists of three sentences for example here.

$\boldsymbol{u_i}$, a reset gate vector $\boldsymbol{r_i}$, and an output vector $\boldsymbol{h_i}$. For each time step i with input $\boldsymbol{x_i}$ and previous hidden state $\boldsymbol{h_{i-1}}$, the updated hidden state $\boldsymbol{h_i} = \text{GRU}(\boldsymbol{x_i}, \boldsymbol{h_{i-1}})$ is computed by:

$$\boldsymbol{u_i} = \sigma(\boldsymbol{W^{(u)}}\boldsymbol{x_i} + \boldsymbol{U^{(u)}}\boldsymbol{h_{i-1}} + \boldsymbol{b^{(u)}}) \quad (2)$$

$$\boldsymbol{r_i} = \sigma(\boldsymbol{W^{(r)}}\boldsymbol{x_i} + \boldsymbol{U^{(r)}}\boldsymbol{h_{i-1}} + \boldsymbol{b^{(r)}}) \quad (3)$$

$$\tilde{\boldsymbol{h_i}} = \tanh(\boldsymbol{W^{(h)}}\boldsymbol{x_i} + \boldsymbol{r_i} \circ \boldsymbol{U}\boldsymbol{h_{i-1}} + \boldsymbol{b^{(h)}}) \quad (4)$$

$$\boldsymbol{h_i} = \boldsymbol{u_i} \circ \tilde{\boldsymbol{h_i}} + (1 - \boldsymbol{u_i}) \circ \boldsymbol{h_{i-1}} \quad (5)$$

where σ is the sigmoid activation function, $\boldsymbol{W^{(u)}}, \boldsymbol{W^{(r)}}, \boldsymbol{W^{(h)}} \in \mathbb{R}^{n_H \times n_I}, \boldsymbol{U^{(u)}}, \boldsymbol{U^{(r)}}, \boldsymbol{U} \in \mathbb{R}^{n_H \times n_H}, n_H$ is the hidden size, n_I is the size of input $\boldsymbol{x_i}$.

To further study the interactions and information exchanges between sentences, we establish a Bi-directional GRU (Bi-GRU) network taking the sentence representation as input:

$$\overrightarrow{\boldsymbol{s_i}} = \text{GRU}_{\text{fwd}}(\hat{\boldsymbol{s}}_i, \overrightarrow{\boldsymbol{s_{i-1}}}) \quad (6)$$

$$\overleftarrow{\boldsymbol{s_i}} = \text{GRU}_{\text{bwd}}(\hat{\boldsymbol{s}}_i, \overleftarrow{\boldsymbol{s_{i-1}}}) \quad (7)$$

$$\overleftrightarrow{\boldsymbol{s_i}} = \overrightarrow{\boldsymbol{s_i}} + \overleftarrow{\boldsymbol{s_i}} \quad (8)$$

where $\hat{\boldsymbol{s}}_i$ is the sentence representation input at time step i, $\overrightarrow{\boldsymbol{s_i}}$ is the hidden state of the forward GRU at time step i, and $\overleftarrow{\boldsymbol{s_i}}$ is the hidden state of the backward GRU. This architecture allows information to flow back and forth to generate new sentence representation $\overleftrightarrow{\boldsymbol{s_i}}$.

Document Encoder: We must initialize a document representation before polishing it. Generating the document representation from sentence representations is a process similar to generating the sentence representation from word embeddings. This time we need to compress the whole document, not just a sentence, into a vector. Because the information a vector can contain is limited, rather than to use another neural network, we simply use a non-linear transformation of the average pooling of the concatenated hidden states of the above Bi-GRU to generate the document representation, as written below:

$$\boldsymbol{D_0} = \tanh(\boldsymbol{W}\frac{1}{n_s}\sum_{i=1}^{n_s}[\overrightarrow{\boldsymbol{s_i}}; \overleftarrow{\boldsymbol{s_i}}] + \boldsymbol{b}) \quad (9)$$

where '$[\cdot;\cdot]$' is the concatenation operation.

Selective Reading module: Now we can formally introduce the selective reading module in Fig.1. This module is a bidirectional RNN consisting of modified GRU cells whose input is the sentence representation $\overleftrightarrow{\boldsymbol{s}} = \{\overleftrightarrow{\boldsymbol{s_1}}, ..., \overleftrightarrow{\boldsymbol{s_{n_s}}}\}$. In the original version of GRU, the update gate $\boldsymbol{u_i}$ in Equation 2 is used to decide how much of hidden state should be retained and how much should be updated. However, due to the way $\boldsymbol{u_i}$ is calculated, it is sensitive to the position and ordering of sentences, but loses information captured by the polished document representation.

Herein, we propose a modified GRU cell that

replace the $\boldsymbol{u_i}$ with the newly computed update gate $\boldsymbol{g_i}$. The new cell takes in two inputs, the sentence representation and the document representation from the last iteration, rather than merely the sentence representation. For each sentence, the selective network generates an update gate vector $\boldsymbol{g_i}$ in the following way:

$$\boldsymbol{f_i} = [\overleftrightarrow{\boldsymbol{s_i}} \circ \boldsymbol{D_{k-1}}; \overleftrightarrow{\boldsymbol{s_i}}; \boldsymbol{D_{k-1}}] \quad (10)$$

$$\boldsymbol{F_i} = \boldsymbol{W^{(2)}}\tanh(\boldsymbol{W^{(1)}}\boldsymbol{f_i} + \boldsymbol{b^{(1)}}) + \boldsymbol{b^{(2)}} \quad (11)$$

$$\boldsymbol{g_i} = \frac{\exp(\boldsymbol{F_i})}{\sum_{j=1}^{n_s}\exp(\boldsymbol{F_j})} \quad (12)$$

where $\overleftrightarrow{\boldsymbol{s_i}}$ is the i-th sentence representation, $\boldsymbol{D_{k-1}}$ is the document representation from last iteration. Equation 5 now becomes:

$$\boldsymbol{h_i} = \boldsymbol{g_i} \circ \tilde{\boldsymbol{h_i}} + (1 - \boldsymbol{g_i}) \circ \boldsymbol{h_{i-1}} \quad (13)$$

We use this "selective reading module" to automatically decide to which extent the information of each sentence should be updated based on its relationship with the polished document. In this way, the modified GRU network can grasp more accurate information from the document.

4.2 Iterative Unit

After each sentence passes through the selective reading module, we wish to update the document representation $\boldsymbol{D_{k-1}}$ with the newly constructed sentence representations. The iterative unit (also depicted above in Fig.1) is designed for this purpose. We use a GRU_{iter} cell to generate the polished document representation, whose input is the final state of the selective reading network from the previous iteration, $\boldsymbol{h_{n_s}}$ and whose initial state is set to the document representation of the previous iteration, $\boldsymbol{D_{k-1}}$. The updated document representation is computed by:

$$\boldsymbol{D_k} = \text{GRU}_{\text{iter}}(\boldsymbol{h_{n_s}}, \boldsymbol{D_{k-1}}) \quad (14)$$

4.3 Decoder

Next, we describe our decoders, which are depicted shaded in the right part of Fig.1. Following most sequence labeling task (Xue and Palmer, 2004; Carreras and Màrquez, 2005) where they learn a feature vector for each sentence, we use a bidirectional GRU_{dec} network in each iteration to output features so as to calculate extracting probabilities. For k-th iteration, given the sentence representation $\overleftrightarrow{\boldsymbol{s}}$ as input and the document representation $\boldsymbol{D_k}$ as the initial state, our decoder encodes the features of all sentences in the hidden state $\boldsymbol{h^k} = \{\boldsymbol{h_0^k}, ..., \boldsymbol{h_{n_s}^k}\}$:

$$\boldsymbol{h_i^k} = \text{GRU}_{\text{dec}}(\overleftrightarrow{\boldsymbol{s}}, \boldsymbol{h_{i-1}^k}) \quad (15)$$

$$\boldsymbol{h_0^k} = \boldsymbol{D_k} \quad (16)$$

4.4 Sentence Labeling Module

Next, we use the feature of each sentence to generate corresponding extracting probability. Since we have one decoder in each iteration, if we directly transform the hidden states in each iteration to extracting probabilities, we will end up with several scores for each sentence. Either taking the average or summing them together by specific weights is inappropriate and inelegant. Hence, we concatenate hidden states of all decoders together and apply a multi-layer perceptron to them to generate the extracting probabilities:

$$\boldsymbol{y} = \boldsymbol{W^{(4)}}\tanh(\boldsymbol{W^{(3)}}[\boldsymbol{h^1}; ...; \boldsymbol{h^k}] + \boldsymbol{b^{(3)}}) + \boldsymbol{b^{(4)}} \quad (17)$$

where $\boldsymbol{y} = \{y_1, ..., y_{n_s}\}$, y_i is the extracting probability for each setence. In this way, we let the model learn by itself how to utilize the outputs of all iterations and assign to each hidden state a reliable weight. In section 6, we will show that this labeling method outperforms other methods.

5 Experiment Setup

In this section, we present our experimental setup for training and estimating our summarization model. We first introduce the datasets used for training and evaluation, and then introduce our experimental details and evaluation protocol.

5.1 Datasets

In order to make a fair comparison with our baselines, we used the CNN/Dailymail corpus which was constructed by Hermann et al. (2015). We used the standard splits for training, validation and testing in each corpus (90,266/1,220/1,093 documents for CNN and 196,557/12,147/10,396 for DailyMail). We followed previous studies in using the human-written story highlight in each article as a gold-standard abstractive summary. These highlights were used to generate gold labels when training and testing our model using the greedy search method similar to (Nallapati et al., 2016a).

We also tested ITS on an out-of-domain corpus, DUC2002, which consists of 567 documents. Documents in this corpus belong to 59 various clusters and each cluster has a unique topic. Each document has two gold summaries written by human experts of length around 100 words.

5.2 Implementation Details

We implemented our model in Tensorflow (Abadi et al., 2016). The code for our models is available online[1]. We mostly followed the settings in (Nallapati et al., 2016a) and trained the model using the Adam optimizer (Kingma and Ba, 2014) with initial learning rate 0.001 and anneals of 0.5 every 6 epochs until reaching 30 epochs. We selected three sentences with highest scores as summary. After preliminary exploration, we found that arranging them according to their scores consistently achieved the best performance. Experiments were performed with a batch size of 64 documents. We used 100-dimension GloVe (Pennington et al., 2014) embeddings trained on Wikipedia 2014 as our embedding initialization with a vocabulary size limited to 100k for speed purposes. We initialized out-of-vocabulary word embeddings over a uniform distribution within [-0.2,0,2]. We also padded or cut sentences to contain exactly 70 words. Each GRU module had 1 layer with 200-dimensional hidden states and with either an initial state set up as described above or a random initial state. To prevent overfitting, we used dropout after each GRU network and embedding layer, and also applied L2 loss to all unbiased variables. The iteration number was set to 5 if not specified. A detailed discussion about iteration number can be found in section 7.

5.3 Baselines

On all datasets we used the Lead-3 method as a baseline, which simply chooses the first three sentences in a document as the gold summary. On DailyMail datasets, we report the performance of SummaRuNNer in (Nallapati et al., 2016a) and the model in (Cheng and Lapata, 2016), as well as a logistic regression classifier (LReg) that they used as a baseline. We reimplemented the Hybrid MemNet model in (Singh et al., 2017) as one of our baselines since they only reported the performance of 500 samples in their paper. Also, Narayan et al. (2018) released their code[2] for the REFRESH model, we used their code to produce Rouge recall scores on the DailyMail dataset as they only reported results on CNN/DailyMail joint dataset. Baselines on CNN dataset are similar. On DUC2002 corpus, we compare our model with several baselines such as Integer Linear Programming (ILR) and LReg. We also report the performance of the newest neural networks model including (Nallapati et al., 2016a; Cheng and Lapata, 2016; Singh et al., 2017).

5.4 Evaluation

In the evaluation procedure, we used the Rouge scores, i.e. Rouge-1, Rouge-2, and Rouge-L, corresponding to the matches of unigram, bigrams, and Longest Common Subsequence (LCS) respectively, to estimate our model. We obtained our Rouge scores using the standard pyrouge package[3]. To compare with other related works, we used full-length F1 score on the CNN corpus, limited length of 75 bytes and 275 bytes recall score on DailyMail corpus. As for the DUC2002 corpus, following the official guidelines, we examined the Rouge recall score at the length of 75 words. All results in our experiment are statistically significant using 95% confidence interval as estimated by Rouge script.

Schluter (2017) noted that only using the Rouge metric to evaluate summarization quality can be misleading. Therefore, we also evaluated our model using human evaluation. Five highly educated participants were asked to rank 40 summaries produced by four models: the Lead-3 baseline, Hybrid MemNet, ITS, and human-authored highlights. We chose Hybrid MemNet as one of the human evaluation baselines since its performance is relatively high compared to other baselines. Judging criteria included informativeness and coherence. Test cases were randomly sampled from DailyMail test set.

6 Experiment analysis

Table 1 shows the performance comparison of our model with other baselines on the DailyMail dataset with respect to Rouge score at 75 bytes and 275 bytes of summary length. Our model performs consistently and significantly better than other models on 75 bytes, while on 275 bytes, the improvement margin is smaller. One possible interpretation is that our model has high precision on top rank outputs, but the accuracy is lower for lower rank sentences. In addition, (Cheng and Lapata, 2016) used additional supervised training

[1] https://github.com/yingtaomj/Iterative-Document-Representation-Learning-Towards-Summarization-with-Polishing

[2] https://github.com/EdinburghNLP/Refresh

[3] https://pypi.python.org/pypi/pyrouge/0.1.0

DailyMail	b75			b275		
	Rouge-1	Rouge-2	Rouge-L	Rouge-1	Rouge-2	Rouge-L
Lead-3	21.9	7.2	11.6	40.5	14.9	32.6
LReg(500)	18.5	6.9	10.2	-	-	-
Cheng *et.al*'16	22.7	8.5	12.5	42.2	17.3	**34.8**
SummaRuNNer	26.2	10.8	14.4	42	16.9	34.1
REFRESH	24.1	11.5	12.5	40.3	15.1	32.9
Hybrid MemNet	26.3	11.2	15.5	41.4	16.7	33.2
ITS	**27.4**	**11.9**	**16.1**	**42.4**	**17.4**	34.1

Table 1: Comparison with other baselines on **DailyMail** test dataset using Rouge recall score with respect to the abstractive ground truth at 75 bytes and at 275 bytes.

CNN	Rouge-1	Rouge-2	Rouge-L
Lead-3	29.1	11.1	25.9
Cheng *et.al*'16	28.4	10.0	25.0
Hybrid MemNet	29.9	11.3	26.4
REFRESH	30.4	11.7	26.9
ITS	**30.8**	**12.6**	**27.6**

Table 2: Comparison with other baselines on **CNN** test dataset using full-length F1 variants of Rouge.

to create sentence-level extractive labels to train their model, while our model uses an unsupervised greedy approximation instead.

We also examined the performance of our model on CNN dataset as listed in Table 2. To compare with other models, we used full-length Rouge F1 metric as reported by Narayan et al. (2018). Results demonstrate that our model has a consistently best performance on different datasets.

In Table 3, we present the performance of ITS on the out of domain DUC dataset. Our model outperforms or matches other basic models including LReg and ILR as well as neural network baselines such as SummaRuNNer with respect to the ground truth at 75 bytes, which shows that our model can be adapted to different copora maintaining high accuracy.

In order to explore the impact of internal structure of ITS, we also conducted an ablation study in Table 4. The first variation is the same model without the selective reading module. The second one sets the iteration number to one, that is, a model without iteration process. The last variation is to apply MLP on the output from the last iteration instead of concatenating the hidden states of all decoders. All other settings and parameters are the same. Performances of these models are worse than that of ITS in all metrics, which demonstrates

DUC2002	Rouge-1	Rouge-2	Rouge-L
Lead-3	43.6	21.0	40.2
LReg	43.8	20.7	40.3
ILP	45.4	21.3	42.8
Cheng *et.al*'16	47.4	23.0	**43.5**
SummaRuNNer	46.6	23.1	43.0
Hybrid MemNet	46.9	23.0	43.1
ITS	**47.6**	**23.4**	**43.5**

Table 3: Comparison with other baselines on **DUC2002** dataset using Rouge recall score with respect to the abstractive ground truth at 75 bytes.

Variations	Rouge-1	Rouge-2	Rouge-L
ITS	27.4	11.9	16.1
w/o selective reading	27.1	11.6	15.4
w/o iteration	26.9	11.6	15.8
w/o concatenation	27.2	11.7	15.9

Table 4: Ablation study on DailyMail test dataset with respect to the abstractive ground truth at 75 bytes.

the preeminence of ITS. More importantly, by this controlled experiment, we can verify the contribution of different module in ITS.

7 Further discussion

Analysis of iteration number: We did a broad sweep of experiments to further investigate the influence of iteration process on the generated summary quality. First, we studied the influence of iteration number. In order to make a fair comparison between models with different iteration number, we trained all models for same epochs without tuning. Fig.2 illustrates the relationship between iteration number and the Rouge score at 75 bytes of summary length on DailyMail test dataset. The result shows that the Rouge score increases with the number of iteration to begin with. After reaching the upper limit it begins to drop. Note that

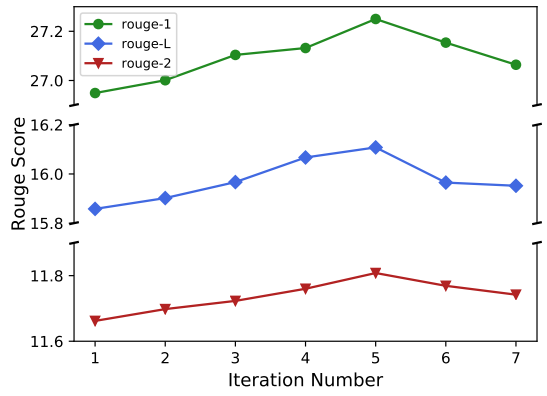
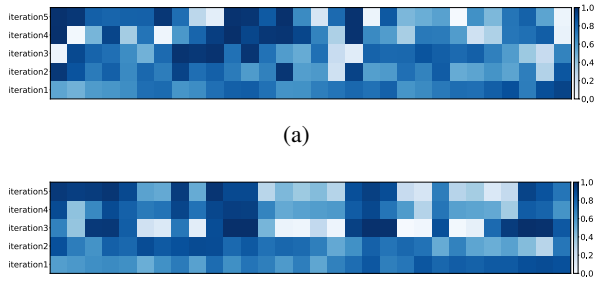


Figure 2: Relationship between number of iteration and Rouge score on DailyMail test dataset with respect to the ground truth at 75 bytes.



(a)

(b)

Figure 3: The predicted extracting probabilities for each sentence calculated by the output of each iteration.

the result of training the model for only one epoch outperforms the state-of-the-art in (Singh et al., 2017), which demonstrates that our selective reading module is effective. The fact that continuing this process increase the performance confirms that the iteration idea behind our model is useful in practice. Based on above observation, we set the default iteration number to be 5.

Analysis of polishing process: Next, to fully investigate how the iterative process influences the extracting results, we draw heatmaps of the extracting probabilities for each decoder at each iteration. We pick two representative cases in Fig.3, where the x-axis represents the sentence index and y-axis is the iteration number, x-axis labels are omitted. The darker the color is, the higher the extracting probability is. In Fig.3(a), it can be seen that when the iteration begins, most sentences have similar probabilities. As we increase the number of iteration, some probabilities begin to fall and others saturate. This means that the model already has preferred sentences to select. Another interesting feature we found is that there is a transitivity between iterations as shown in Fig.3(b). To be specific, the sentences which are not preferred by iteration 3 remain low probabilities in the next two iterations, while sentences with relatively high scores are still preferred by iteration 4 and 5.

Models	1st	2nd	3rd	4th
Lead-3	0.12	0.11	0.25	**0.52**
Hybrid MemNet	0.24	0.25	**0.28**	0.23
ITS	0.31	**0.34**	0.23	0.12
Gold	**0.33**	0.30	0.24	0.13

Table 5: System ranking comparison with other baselines on DailyMail corpus. Rank 1 is the best and Rank 4 is the worst. Each score represents the percentage of the summary under this rank.

Human Evaluation: We gave human evaluators three system-generated summaries, generated by Lead-3, Hybrid MemNet, ITS, as well as the human-written gold standard summary, and asked them to rank these summaries based on summary informativeness and coherence. Table 5 shows the percentages of summaries of different models under each rank scored by human experts. It is not surprising that gold standard has the most summaries of the highest quality. Our model has the most summaries under 2nd rank, thus can be considered 2nd best, following are Hybrid MemNet and Lead-3, as they are ranked mostly 3rd and 4th. By case study, we found that a number of summaries generated by Hybrid MemNet have two sentences the same as ITS out of three, however, the third distinct sentence from our model always leads to a better evaluation result considering overall informativeness and coherence. Readers can refer to the appendix to see our case study.

8 Conclusion

In this work, we introduce ITS, an iteration based extractive summarization model, inspired by the observation that it is often necessary for a human to read the article multiple times to fully understand and summarize it. Experimental results on CNN/DailyMail and DUC corpora demonstrate the effectiveness of our model.

Acknowledgments

We would like to thank the anonymous reviewers for their constructive comments. We would also like to thank Jin-ge Yao and Zhengyuan Ma for their valuable advice on this project. This work was supported by the National Key Re-

search and Development Program of China (No. 2017YFC0804001), the National Science Foundation of China (NSFC No. 61876196, No. 61672058). Rui Yan was sponsored by CCF-Tencent Open Research Fund and Microsoft Research Asia (MSRA) Collaborative Research Program.

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283.

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the conll-2005 shared task: Semantic role labeling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, CONLL '05, pages 152–164, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words.

Kyunghyun Cho, Bart Van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *Computer Science*.

Kevin Bretonnel Cohen. 2002. Natural language processing for online applications: Text retrieval, extraction and categorization (review). *Language*, 80(3):510–511.

Karl Moritz Hermann, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. pages 1693–1701.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *Computer Science*.

Ramesh Nallapati, Igor Melnyk, Abhishek Kumar, and Bowen Zhou. 2017. Sengen: Sentence generating neural variational topic model.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2016a. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents.

Ramesh Nallapati, Bowen Zhou, Cicero Nogueira Dos Santos, Caglar Gulcehre, and Bing Xiang. 2016b. Abstractive text summarization using sequence-to-sequence rnns and beyond.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Ranking sentences for extractive summarization with reinforcement learning.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543.

Dragomir R Radev, Timothy Allison, Sasha Blair-Goldensohn, John Blitzer, Arda Celebi, Stanko Dimitrov, Elliott Drabek, Ali Hakim, Wai Lam, Danyu Liu, et al. 2004. Mead-a platform for multidocument multilingual text summarization. In *LREC*.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *Computer Science*.

Natalie Schluter. 2017. The limits of automatic summarisation according to rouge. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, volume 2, pages 41–45.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. *CoRR*, abs/1704.04368.

Abhishek Kumar Singh, Manish Gupta, and Vasudeva Varma. 2017. Hybrid memnet for extractive summarization. pages 2303–2306.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. *Computer Science*.

Kristian Woodsend and Mirella Lapata. 2010. Automatic generation of story highlights. In *Meeting of the Association for Computational Linguistics*, pages 565–574.

Yuxiang Wu and Baotian Hu. 2018. Learning to extract coherent summary via deep reinforcement learning. *arXiv preprint arXiv:1804.07036*.

Caiming Xiong, Stephen Merity, and Richard Socher. 2016. Dynamic memory networks for visual and textual question answering.

Nianwen Xue and Martha Palmer. 2004. Calibrating features for semantic role labeling. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*.

Rui Yan. 2016. i, poet: Automatic poetry composition through recurrent neural networks with iterative polishing schema. In *IJCAI*, pages 2238–2244.

Rui Yan, Cheng Te Li, Xiaohua Hu, and Ming Zhang. 2016. Chinese couplet generation with neural network structures. In *Meeting of the Association for Computational Linguistics*, pages 2347–2357.

Rui Yan, Jian Yun Nie, and Xiaoming Li. 2011a. Summarize what you are interested in: An optimization framework for interactive personalized summarization. In *Conference on Empirical Methods in Natural Language Processing, EMNLP 2011, 27-31 July 2011, John Mcintyre Conference Centre, Edinburgh, Uk, A Meeting of Sigdat, A Special Interest Group of the ACL*, pages 1342–1351.

Rui Yan, Xiaojun Wan, Mirella Lapata, Wayne Xin Zhao, Pu-Jen Cheng, and Xiaoming Li. 2012. Visualizing timelines: Evolutionary summarization via iterative reinforcement between text and image streams. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 275–284. ACM.

Rui Yan, Xiaojun Wan, Jahna Otterbacher, Liang Kong, Xiaoming Li, and Yan Zhang. 2011b. Evolutionary timeline summarization: a balanced optimization framework via iterative substitution. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 745–754.