A Joint Sentence Scoring and Selection Framework for Neural Extractive Document Summarization

Qingyu Zhou, Nan Yang, Furu Wei, Shaohan Huang, Ming Zhou, and Tiejun Zhao

***Abstract*—Extractive document summarization methods aim to extract important sentences to form a summary. Previous works perform this task by first scoring all sentences in the document then selecting most informative ones; while we propose to jointly learn the two steps with a novel end-to-end neural network framework. Specifically, the sentences in the input document are represented as real-valued vectors through a neural document encoder. Then the method builds the output summary by extracting important sentences one by one. Different from previous works, the proposed joint sentence scoring and selection framework directly predicts the relative sentence importance score according to both sentence content and previously selected sentences. We evaluate the proposed framework with two realizations: a hierarchical recurrent neural network based model; and a pre-training based model that uses BERT as the document encoder. Experiments on two datasets show that the proposed joint framework outperforms the state-of-the-art extractive summarization models which treat sentence scoring and selection as two subtasks.**

***Index Terms*—Summarization, natural language processing, neural networks.**

I. Introduction

TRADITIONAL approaches to automatic text summarization focus on identifying important content, usually at sentence level [1]. A summarization system forms an output summary of the given document by extracting the identified important sentences. In recent decades, the *extractive document summarization* methods have proven effective in many systems [2]–[5]. In previous systems which use extractive methods, text document summarization is decomposed into two subtasks, i.e., *sentence scoring* and *sentence selection*.

Sentence scoring, as the first step of extractive summarization, has been investigated in many previous works. It aims to measure the importance of sentences and assign scores to each sentence. Feature-based methods have proven effective and are broadly used. Handcrafted features such as word probability, TF-IDF scores, the position of a sentence in a document and sentence length are used in many previous systems [6]–[8]. Based on weighted-graphs, graph-based methods like TextRank [3] and LexRank [9] are also effective for sentence scoring. In recent years, neural network models have proven very helpful in representation learning. Recent works [5], [8] adopt neural methods to represent sentences for calculating sentence importance scores.

As the second step, *sentence selection* adopts a particular strategy to choose the important sentences to form the output summary. Carbonell and Goldstein [2] propose Maximal Marginal Relevance method, which is also applied as a sentence selection strategy in summarization. It chooses the sentence which has the maximal importance score while having minimal redundant content with previously selected sentences. McDonald [4] proposes to treat the sentence selection task as an optimization problem under certain constraints (e.g., summary length limit). Therefore, Integer Linear Programming is used to solve this optimization problem. Furthermore, Lin and Bilmes [10] argue that the submodular functions can also be used to solve this optimization problem.

In this paper, we propose a joint sentence scoring and selection framework for neural extractive document summarization (NeuSum). Different from previous works which decompose extractive document summarization task into sentence scoring and sentence selection steps, our method integrates them into one end-to-end trainable neural network model. Specifically, NeuSum learns to identify the relative sentence importance without any handcrafted features. The relative sentence importance is measured by the gain over the current partial output summary. Therefore, at each time step, NeuSum scores the remaining sentences by considering both their content and previously selected sentences. Through the joint learning process, NeuSum learns to predict the relative gain of each sentence given the current extraction state and the partial output summary.

The proposed NeuSum framework consists of two modules, i.e., the document encoder and the sentence extractor. In this paper, we introduce the NeuSum$_{\text{BERT}}$ model which realizes the NeuSum framework. In detail, we employ the BERT [11] model as the document encoder in NeuSum$_{\text{BERT}}$ to represent the sentences in the given document. After representing the input sentences, we use a recurrent neural network (RNN) to build the sentence extractor. Specifically, a two-layer RNN with Layer Normalization [12] is used in the sentence extractor. The RNN-based sentence extractor has the following two main functionalities. On one hand, the RNN can remember the summary

Manuscript received June 27, 2019; revised November 5, 2019; accepted December 28, 2019. Date of publication January 6, 2020; date of current version January 22, 2020. This work was supported in part by the project of National Natural Science Foundation of China under Grant 91520204 and in part by the project of National Key Research and Development Program of China under Grant 2017YFB1002102. The work of Qingyu Zhou was funded by the Harbin Institute of Technology Scholarship Fund. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Eric Fosler-Lussier. *(Corresponding author: Qingyu Zhou.)*

Q. Zhou and T. Zhao are with the School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China (e-mail: qyzhou@hit.edu.cn; tjzhao@hit.edu.cn).

N. Yang, F. Wei, S. Huang, and M. Zhou are with Microsoft Research Asia, Beijing 100080, China (e-mail: nanya@microsoft.com; fuwei@microsoft.com; shaohanh@microsoft.com; mingzhou@microsoft.com).

Digital Object Identifier 10.1109/TASLP.2020.2964427

output history by feeding the previously selected sentences into it. On the other hand, its hidden state is used as a sentence extraction state which helps to score the remaining sentences with their representations. When selecting the t-th output sentence, the sentence extractor reads the representation of the previously extracted sentence $t-1$. It outputs a new sentence extraction state and then uses it to measure the relative importance of the remaining sentences.

We evaluate the proposed framework with two realizations, i.e., NEUSUM$_{\text{RNN}}$ (a hierarchical RNN based model which realizes NEUSUM in our preliminary work [13]) and NEUSUM$_{\text{BERT}}$ model on the *CNN/Daily Mail* and the *New York Times* datasets. The experimental results show that the proposed joint sentence scoring and selection framework NEUSUM (both NEUSUM$_{\text{RNN}}$ and NEUSUM$_{\text{BERT}}$) improves the performance compared with previous separated methods. The contributions of this paper are summarized as follows:

- We propose a joint sentence scoring and selection framework for extractive neural document summarization.
- The proposed framework can be end-to-end trained without handcrafted features.
- Two architectures, i.e., NEUSUM$_{\text{RNN}}$ and NEUSUM$_{\text{BERT}}$, which realize the proposed framework outperforms state-of-the-art methods on the *CNN/Daily Mail* and *New York Times* datasets.

This article presents an extended version of our preliminary work [13]. We add several new state-of-the-art baseline methods and experimental results. We provide novel algorithmic enhancements that a new architecture of NEUSUM framework, i.e., NEUSUM$_{\text{BERT}}$, is presented. More details about our framework and a comprehensive description of converting abstractive summary to extractive supervision are provided. We conduct more in-depth analyses and ablations to analyze the proposed NEUSUM framework.

II. Problem Formulation

Extractive document summarization aims to extract informative sentences to represent the important meanings of a given document. Given an input document $\mathcal{D} = (S_1, S_2, \ldots, S_L)$ with L sentences, an extractive summarization system should find a sentence subset $\mathcal{S}$ to produce the output summary: $\mathcal{S} = \{\hat{S}_i | \hat{S}_i \in \mathcal{D}\}$. In the training phase, the reference summary $\mathcal{S}^*$ is available so that we can calculate the score of an output summary $\mathcal{S}$ under a certain evaluation function $r(\mathcal{S}|\mathcal{S}^*)$, that $r(\cdot)$ is a function for summary quality evaluation. The goal of training a summarization system is to learn a scoring function $g(\mathcal{S})$ from the training data so that it can be used to produce an output summary during testing:

$$\begin{aligned} \arg\max_{\mathcal{S}} \quad & f(\mathcal{S}) \\ \text{s.t.} \quad & \mathcal{S} = \{\hat{S}_i | \hat{S}_i \in \mathcal{D}\} \\ & |\mathcal{S}| \leq l \end{aligned}$$

where l is the limit of the number of sentences.

In this paper, we conduct experiments on the *CNN/Daily Mail* and *New York Times* datasets, in which the reference summaries

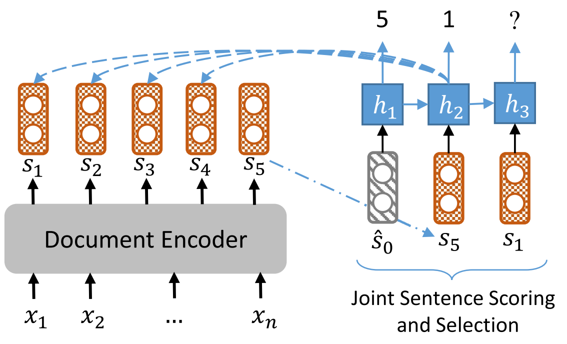

Fig. 1. Overview of the NEUSUM framework. In this example (5 sentences consisting of n words), it extracts S_5 and S_1 at the first two steps. At the first step, the model is fed with a special vector $\hat{s}_0$ to represent an empty partial output summary. In the following steps, the vector representations of previously selected sentences, i.e., s_5 and s_1, are fed into the extractor. At the second step, the model only needs to score the first four sentences since the 5th sentence is already extracted.

do not have a length limit. Following previous works [14]–[17], we employ ROUGE F1 [18] as the evaluation function $r(\cdot)$, and set the length limit l as a fixed number of sentences.

Therefore, the NEUSUM model is trained to learn a function $g(\cdot)$ to measure the ROUGE F1 gain given previously selected sentences at time step t. Specifically, function $g(\cdot)$ measures the relative importance of the remaining candidate sentences when choosing the t-th output sentence $\hat{S}_t$:

$$g(\hat{S}_t = S_i | \mathbb{S}_{t-1}) = r\left(\mathbb{S}_{t-1} \cup \{S_i\}\right) - r(\mathbb{S}_{t-1}) \tag{1}$$

where $\mathbb{S}_{t-1}$ is the set of previously selected sentences. We omit the condition $\mathcal{S}^*$ in the evaluation function $r(\cdot|\mathcal{S}^*)$ for simplicity. At each time t, the summarization system selects the sentence with maximal ROUGE F1 gain score until the number of output summary reaching the output length limit l. In this paper, we use S_i to represent sentence i in the document, and $\hat{S}_t$ to denote the extracted sentence at time step t.

III. Neural Document Summarization

Fig. 1 shows the overview of the proposed NEUSUM framework. NEUSUM framework consists of two main components, i.e., a document encoder, and a sentence extractor. The document encoder transforms the sentences in the input document into real-valued vector representations. Based on the sentence vectors, the sentence extractor measures the importance of sentences. It extracts one sentence and updates the extraction history at each step until reaching the output length limit. In our NEUSUM framework, the document encoder and the sentence extractor can have different architectures. In this paper, we employ the pre-trained BERT [11] based document encoder since it performs well on various downstream tasks. The sentence extractor is then designed as a multi-layer Gated Recurrent Unit (GRU) with Layer Normalization [12]. We denote this specific network structure as NEUSUM$_{\text{BERT}}$. In our preliminary work [13], we use a hierarchical RNN document encoder and a single layer RNN sentence extractor, which is denoted as NEUSUM$_{\text{RNN}}$ in this paper. In this section, we will describe the BERT-based document encoder, and present the joint sentence scoring and

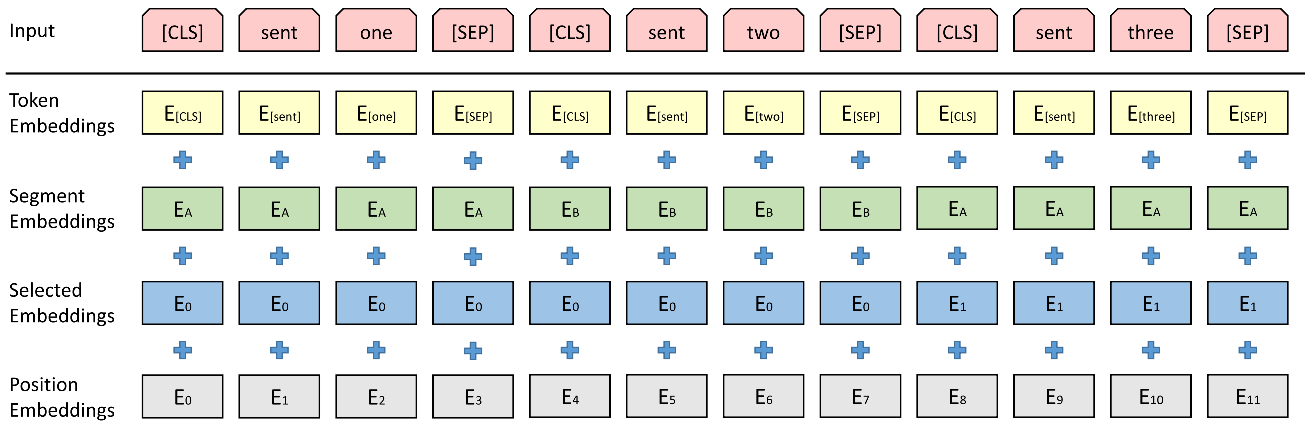

Fig. 2. The BERT-based document encoding input representation. Three sentences are fed into the model, and the third one is selected in the previous step.

selection model NeuSum$_{\text{BERT}}$. Finally, we introduce the training objective function of the proposed framework.

A. BERT-Based Document Encoding

Bidirectional Encoder Representations from Transformers (BERT) [11] has shown promising results on various downstream tasks, such as Question Answering and POS Tagging. Recently, Zhang *et al.* [19], Yang [20] and Ashutosh *et al.* [21] have shown that BERT performs well on document modeling. In this work, we use the pretrained BERT as the document encoder. As the basic block of BERT, Multi-Head Attention is proposed for Transformer [22], which is a new architecture for encoder-decoder models. Multi-Head Attention can be viewed as a feature extractor, which extracts information from a set of key-value pairs (K, V) given a query vector Q:

$$\text{MultiHead}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \text{Concat}(head_1,\ldots,head_A)\mathbf{W}^O \quad (2)$$

$$\text{where } head_i = \text{Attention}(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V) \quad (3)$$

where $\mathbf{W}_i^Q$, $\mathbf{W}_i^K$, $\mathbf{W}_i^V$ and $\mathbf{W}^O$ are parameter matrices. The particular attention mechanism used in BERT is called "Scaled Dot-Product Attention," which is defined as:

$$\text{Attention}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V} \quad (4)$$

where d_k is the dimension of the key column vector in $\mathbf{K}$.

Fig. 2 shows how the BERT-based encoder encodes the input document. The input of the BERT-based encoder is four types of embeddings, namely, the token embeddings, the segment embeddings, the selected embeddings and the position embeddings. The token, segment and position embeddings are pre-trained in BERT. In this work, we introduce an extra embedding, the selected embedding, as a task-specific input for extractive document summarization. It represents whether the sentence is selected in previous steps. If one sentence is selected, the selected embedding is set to E_1, otherwise E_0. For example, in Fig. 2, the third sentence is selected in the previous step. We then update its corresponding selected embedding from E_0 to E_1. We add the four types of embeddings together as the BERT input.

Following the previous practice of document encoding using BERT [11], [19], [20], we use vector of the `[CLS]` symbol of a sentence in the top layer as its representation. Then we can get the final sentence representations in the given document: $D_t = (\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_L)$. In this paper, we simplify D_t to D, and use sentence S_i and its representative vector $\mathbf{s}_i$ interchangeably.

B. Joint Sentence Scoring and Selection

Separate sentence scoring and selection in previous works cannot utilize the information of each other. By jointly learning to score and select sentences, the proposed NeuSum model can make these two steps benefit each other. We couple these two steps together so that: a) sentence scoring can leverage the information of previously selected sentences to make better sentence importance measurement; b) sentence selection step can be simplified since the scoring function is learned to be the Rouge score gain over the currently selected sentences as described in Equation (1) in section II.

When extracting the t-th output sentence, the sentence extractor is given the last extracted sentence $\hat{S}_{t-1}$ and chooses $\hat{S}_t$ by scoring the remaining sentences. Therefore, the model is able to leverage the extraction history. Concretely, the model should have two key abilities: 1) remembering the information of previously extracted sentences; 2) scoring the importance of document sentences based on both their content and the previously selected sentences. Therefore, we use a GRU as the recurrent unit to remember the sentence extraction history. To make the training process more stable, we apply the Layer Normalization [12] on the GRU cell. At time step t, the update of the GRU cell is calculated as:

$$\begin{pmatrix}\mathbf{z_t}\\ \mathbf{r}_t\end{pmatrix} = LN(\mathbf{W}_h\mathbf{h}_{t-1};\boldsymbol{\alpha}_1,\boldsymbol{\beta}_1) + LN(\mathbf{W}_x\mathbf{x}_t;\boldsymbol{\alpha}_2,\boldsymbol{\beta}_2) \quad (5)$$

$$\hat{\mathbf{h}}_t = \tanh(LN(\mathbf{W}\mathbf{x}_t;\boldsymbol{\alpha}_3,\boldsymbol{\beta}_3) + \sigma(\mathbf{r}_t)\odot LN(\mathbf{U}\mathbf{h}_{t-1};\boldsymbol{\alpha}_4,\boldsymbol{\beta}_4)) \quad (6)$$

$$\mathbf{h}_t = (1-\sigma(\mathbf{z}_t))\mathbf{h}_{t-1} + \sigma(\mathbf{z}_t)\hat{\mathbf{h}}_t \quad (7)$$

where $\mathbf{W}_h$, $\mathbf{W}_x$, $\mathbf{W}$ and $\mathbf{U}$ are parameters in the GRU cell. $LN(\mathbf{x};\boldsymbol{\alpha},\boldsymbol{\beta})$ represents the Layer Normalization operation with parameter $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ in [12]. In this paper, we use a two-layer GRU with Layer Normalization.

The sentence scorer is built with a bilinear layer which takes the GRU states and sentence representations as input. Specifically, the GRU takes the document level sentence vector $\mathbf{s}_{t-1}$ of the last extracted sentence $\hat{S}_{t-1}$ as input to produce its hidden state $\mathbf{h}_t$ at step t. To measure the relative importance of sentence S_i given the current extraction history, the sentence scorer takes its representation vector $\mathbf{s}_i$ and the current hidden state $\mathbf{h}_t$ of the top GRU layer, to predict its score $\delta(S_i)$:

$$\mathbf{h}_t = \text{GRU}(\mathbf{s}_{t-1}, \mathbf{h}_{t-1}) \tag{8}$$

$$\delta(S_i) = \frac{\mathbf{h}_t \mathbf{W}_s \mathbf{s}_i + \mathbf{b}_s}{\sqrt{d_s}} \tag{9}$$

where $\mathbf{W}_s$ and $\mathbf{b}_s$ are the parameters in the bilinear layer. $\frac{1}{\sqrt{d_s}}$ (d_s is the dimension of $\mathbf{s}_i$) is the scaling factor similar to [22], which is used to prevent dot products growing too large in magnitude. We found that using this scaling factor makes the training process more stable.

We initialize the GRU hidden state $\mathbf{h}_0$ with a single-layer MLP using the sentence vectors in the document:

$$\mathbf{h}_0 = \tanh(\mathbf{W}_m \mathbf{d}_{\text{avg}} + \mathbf{b}_m) \tag{10}$$

$$\mathbf{d}_{\text{avg}} = \frac{1}{L}\sum_{i=1}^{L}(\mathbf{s}_i) \tag{11}$$

$$\mathbb{S}_0 = \varnothing \tag{12}$$

$$\hat{\mathbf{s}}_0 = \mathbf{v} \tag{13}$$

where $\mathbf{W}_m$ and $\mathbf{b}_m$ are learnable parameters. The vector $\mathbf{d}_{\text{avg}}$ is the average of all the sentence vectors in the input document, and we use it to represent the information of the entire document. We learn a special parameter, $\mathbf{v}$, to represent the previously extracted sentence at time step $t = 1$, i.e., $\hat{\mathbf{s}}_0 = \mathbf{v}$, since the model has not extracted any sentences yet. The parameter $\mathbf{v}$ is updated during training.

After scoring all the remaining sentences at time t, we can choose the sentence with the maximal gain score:

$$\hat{S}_t = \underset{S_i \in \mathcal{D} \backslash \mathbb{S}_{t-1}}{\arg\max}\ \delta(S_i) \tag{14}$$

C. Training Objective

Inspired by [23], we optimize the Kullback-Leibler (KL) divergence of the model prediction P and the labeled data distribution Q during training. To get the model prediction distribution P, we normalize the model predicted sentence score $\delta(S_i)$ with softmax function at time step t:

$$P(\hat{S}_t = S_i) = \frac{\exp(\delta(S_i))}{\sum_{k=1}^{L}\exp(\delta(S_k))} \tag{15}$$

As previously formulated in Equation (1), the model is expected to learn the relative ROUGE F1 gain at time step t with respect to previously selected sentences $\mathbb{S}_{t-1}$. Since that the F1 gain values are in different ranges, we follow previous works [8] to use Min-Max Normalization to rescale it to $[0, 1]$:

$$g(\hat{S}_t = S_i) = r(\mathbb{S}_{t-1} \cup \{S_i\}) - r(\mathbb{S}_{t-1}) \tag{16}$$

$$\widetilde{g}(\hat{S}_t = S_i) = \frac{g(S_i) - \min_j(g(S_j))}{\max_j(g(S_j)) - \min_j(g(S_j))}. \tag{17}$$

We then apply a softmax function with temperature τ [24][1] to produce the labeled data distribution Q as the training target. The temperature parameter τ.[2] is used as a smoothing factor to produce a smoothed label distribution Q:

$$Q(S_i) = \frac{\exp(\tau\widetilde{g}(S_i))}{\sum_{k=1}^{L}\exp(\tau\widetilde{g}(S_k))}. \tag{18}$$

With labeled data distribution Q and the model prediction distribution P, we minimize the *KL* loss function J:

$$J = D_{KL}(Q\|P) \tag{19}$$

IV. Experiments

A. Dataset

Large scale datasets with human-created summary are essential for training neural network-based summarization models. Therefore, we employ the current two largest summarization datasets, the *CNN/Daily Mail* dataset [17], [25] and *New York Times* [26] dataset, as the training and evaluation datasets in our experiments.

1) CNN/Daily Mail*:* The *CNN/Daily Mail* dataset [25] is constructed from the two famous newswire website, i.e., CNN news and Daily Mail news. The datasets contain articles and their corresponding highlights, where the highlights are written by human editors and therefore are abstractive summaries.

Following [14], we preprocess this dataset using the same method,[3] which includes sentence splitting and word tokenization. Some previous works [16], [17] employ the *anonymized* version of the *CNN/Daily Mail* data, where the named entities are replaced by a placeholder such as `entity2`. However, the current state-of-the-art named entity taggers may still produce inaccurate results. Moreover, most recent works [14], [19], [20], [27]–[29] on single-document summarization use the *non-anonymized* version *CNN/Daily Mail*. Therefore, we use the *non-anonymized* version so we can directly operate on the original text. Table I shows the data statistics of the *CNN/Daily Mail* dataset.

2) New York Times*:* The *New York Times* dataset is built from the LDC New York Times Annotated Corpus.[4] Durrett *et al.* [26] propose to use the news articles from 2003 to 2007 as a summarization training and test sets. We build the dataset with their released data split files and scripts.[5] The script results in document-summary pairs with sentence splitting and word tokenization. The data split used in [26] does not

[1]We set $\tau = 4$ empirically according to the model performance on the development set.

[2]The temperature τ here is the inverse of the temperature in [24]

[3][Online]. Available: https://github.com/abisee/cnn-dailymail

[4]LDC Catalog No.: LDC2008T19

[5][Online]. Available: http://nlp.cs.berkeley.edu/projects/summarizer.shtml

TABLE I
DATA STATISTICS OF *CNN/DAILY MAIL* DATASET

CNN/Daily Mail	Training	Dev	Test
#(Document)	287,227	13,368	11,490
#(Ref / Document)	1	1	1
Doc Len (Sentence)	31.58	26.72	27.05
Doc Len (Word)	791.36	769.26	778.24
Ref Len (Sentence)	3.79	4.11	3.88
Ref Len (Word)	55.17	61.43	58.31

TABLE II
DATA STATISTICS OF *NEW YORK TIMES* DATASET

New York Times	Training	Dev	Test
#(Document)	98,826	2,000	9,706
#(Ref / Document)	1	1	1
Doc Len (Sentence)	40.41	40.39	40.09
Doc Len (Word)	975.40	977.35	983.50
Ref Len (Sentence)	2.88	2.92	3.07
Ref Len (Word)	51.46	52.01	54.42

have a development set. Therefore, we further sample 2000 document-summary pairs as the development set. The final data statistics of the *New York Times* are shown in Table II.

3) Creation of the Training Supervision: To train an extractive neural summarization model, the system should be given the supervision that which sentences should be extracted. However, both summaries of the *CNN/Daily Mail* and *New York Times* are written by human editors and abstractive. Therefore, these human-written summaries cannot be directly used as the training supervision for extractive summarization systems since they do not appear exactly in the corresponding documents.

To create the supervision label, we propose a simple method to search the sentences which should be extracted. We apply this algorithm on both *CNN/Daily Mail* and *New York Times* datasets. Algorithm 1 shows the pseudo-code of converting abstractive summarization to extractive training supervision. Specifically, Algorithm 1 finds the sentence combination which has the maximum ROUGE-2 F1 score among all possible sentence combinations. However, it is computationally expensive to search the global optimal solution of sentence combination, we propose to use a greedy strategy. In detail, given a document with n sentences, the algorithm enumerates the combination of candidates from 1-combination $\binom{n}{1}$ to n-combination $\binom{n}{n}$ (In this paper, we use $\binom{n}{k}$ to represent the set of all the k-combination sentence sets. The number of combination is represented as the cardinal number: $|\binom{n}{i}|$). For the k-combination sentence candidates, we calculate the ROUGE-2 F1 scores of all candidates. The algorithm picks the best k-combination, comb-(k), which has the highest ROUGE-2 F1 score. The algorithm stops searching if the ROUGE-2 F1 score of comb-(k) is less than the previous best sentence combination candidate comb-$(k-1)$. We implemented this algorithm with Python and release it with the model source code. In our experiments, we set the max_oracle_len to 5 and max_comb to 100,000[6] to limit the search space so

Algorithm 1: Converting Abstractive Summarization to Extractive Training Supervision.

```
  Input: Document $\mathcal{D}$ with reference summary $\mathcal{S}^*$.
 max_oracle_len: the max sentence number in the output.
 max_comb: the max combination number to search.
  Output: Extractive training supervision for NeuSum
 1: procedure SearchOracle
 2:   $best = \emptyset$, $best\_score = 0$, $n = |\mathcal{D}|$
 3:   for $i \leftarrow 1$ to max_oracle_len do
 4:     $comb\_num = |\binom{n}{i}|$
 5:     if $comb\_num >$ max_comb then
 6:       break
 7:     scores = [ (ROUGE(comb)) for comb in $\binom{n}{i}$]
 8:     $max\_score = \max(scores)$
 9:     if $max\_score < best\_score$ then
10:       break
11:     $best = \arg\max_{\text{comb}}(\text{scores})$
12:     $best\_score = max\_score$
13:   return $best$
```

that the algorithm could run in a reasonable time. The program is efficient and needs less than 1 hour using 36 CPU cores to process the *CNN/Daily Mail* data with an Intel(R) Xeon(R) E5-2630 v4 2.20 GHz CPU.

B. Implementation Details

We introduce the implementation details of NEUSUM$_{\text{BERT}}$ in this section. More details about the implementation of NEUSUM$_{\text{RNN}}$ can be found in [13].

1) Model Parameters: We use the 'bert-base-uncased'[7] model to initialize the BERT-based encoder. We choose to use the BERT-base model because our GPU cannot fit the BERT-large model even with batch size of 1. The 'bert-base-uncased' model has 12 layers of Multi-Head Attention [22] with hidden size $H = 768$ and self-attention heads $A = 12$. The total parameter number of 'bert-base-uncased' is 110 M. The selected embedding is implemented as a look-up table. The embedding dimension is 768 which is the same as the other embeddings in the 'BERT-base' model. Since this is a new parameter matrix in the BERT model, we initialize it with the parameter of the segment embedding. The sentence extractor is a two-layer Layer-Normalized GRU (LNGRU), and its hidden size is set to 768. A dropout layer is added after the first LNGRU layer with dropout probability set to 0.15. The dimension of the learned parameter $\mathbf{v}$ representing the previously extracted sentence at the first step in Equation (13) is set to 768.

2) Model Training: The parameter matrices in the BERT-based encoder are initialized using the pre-trained 'bert-base-uncased' model. Other parameters in the NEUSUM$_{\text{BERT}}$ model are randomly initialized using Xavier uniform distribution [30]. The BERT-based encoder has a fixed-size vocabulary which

[6]We empirically choose this number so that the algorithm could both terminate in a reasonable time and yield satisfying results.

[7]We use the PyTorch implmentation of BERT: [Online]. Available: https://github.com/huggingface/pytorch-pretrained-BERT

is collected from a large scale training data. We use the subword tokenizer which is the same as BERT to process the input document. Since the maximum position in the pre-trained position embedding of BERT is 512, we truncate the articles to 512 subwords. We implement the NeuSum$_{\text{BERT}}$ model with PyTorch [31]. The model is trained with 4 Nvidia P100 GPU for 50,000 steps. The mini-batch size is set to 35 to fully utilize the GPU memory. The source code and related resources will be released at https://res.qyzhou.me.

3) Optimizer: The NeuSum$_{\text{BERT}}$ model is trained with stochastic gradient descent. Specifically, we choose Adam [32] as the training optimizer. We set the hyper-parameters of Adam as follows: momentum parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. We use the learning rate scheduler in Transformer [22] for the Adam optimizer, according to following formula:

$$lr = \alpha \cdot \min\left(step\_num^{-0.5}, step\_num \cdot warmup\_steps^{-1.5}\right) \tag{20}$$

where $\alpha = 0.002$. This learning rate curve corresponds to increasing the learning rate linearly for $warmup\_steps$, and then decreasing it to the inverse square root of the training step number. We use $warmup\_steps = 10,000$.

4) Model Testing: LEAD3 is a strong and commonly used extractive summarization baseline. And recent works [15], [16], [19], [20], [29] which use *CNN/Daily Mail* dataset choose to extract 3 sentences. Therefore, we extract 3 sentences during test to make our method and the baseline systems comparable to each other.

C. Baseline

We compare NeuSum models (RNN and BERT) with the following baseline systems:

LEAD3 Since the newswire articles tend to present important information at the beginning, selecting the leading sentences is a strong and commonly used baseline method. Following previous works [14]–[16], [35], we select the first three sentences as the LEAD3 baseline.

TextRank An unsupervised algorithm based on weighted-graphs proposed by Mihalcea and Tarau [3]. This method represents each sentence as a vertex and then builds edges between them. We use the TextRank implementation in the Gensim toolkit [37].

PGN Pointer-Generator Network (PGN) is a state-of-the-art abstractive document summarization system proposed by See *et al.* [14]. PNG is a hybrid method which incorporates copying mechanism [38], [39] in to the sequence-to-sequence generation model.

DCA DCA is a state-of-the-art abstractive summarization system proposed by Celikyilmaz *et al.* [33]. It consists of multiple communicating agents to encode the document.

JECS Xu and Durrett [29] propose a two stage method which first selects sentences and then decides which of a set of compression options to apply to each selected sentence.

Bottom-Up Gehrmann *et al.* [28] propose a two stage method which constrains the copy probability in PGN.

SRC-ELMO Edunov *et al.* [34] propose using a pre-trained language model as the input features in the Transformer based seq2seq model.

CRSum CRSum is an extractive summarization model proposed by Ren *et al.* [8]. It considers the contextual information of a sentence. We implement this model and take it as a baseline method.

Refresh Narayan *et al.* [35] also view document summarization as a sequence labeling task. They argue that cross-entropy training is not optimal for extractive summarization. Thus they propose use globally optimizing the Rouge evaluation metric through a reinforcement learning objective.

SummaRuNNer Based on [15], [16] propose that using some interpretable lexical features such as absolute and relative sentence positions should improve the model performance.

BanditSum Dong *et al.* [27] propose to treat extractive summarization as contextual bandit problem. A policy gradient reinforcement learning algorithm is used to train the model to select sequences of sentences that maximize ROUGE score.

NN-SE NN-SE is a neural extractive summarization model proposed by Cheng and Lapata [15] which takes document summarization as a sequence labeling task.

BERT Similar to [15], [19] and [20] both propose finetuning BERT model by feeding the sentences into BERT and using the representation at `[CLS]` token to classify each sentence as a sequence labeling task.

Self-Supervised Wang *et al.* [36] propose a self-supervised learning method which introduces three auxiliary pre-training tasks to capture the document-level context.

HIBERT HIBERT is a pre-trained hierarchical multi-layer Transformer proposed by Zhang *et al.* [19]. It is pre-trained similar to BERT that it performs document masking and sentence prediction as the pre-training tasks.

D. Evaluation Metric

We use Rouge (v1.5.5) [18] as the automatic evaluation metric. Rouge evaluates the quality of a system summary by computing overlapping lexical units. The most commonly reported metrics are Rouge-1 (unigram), Rouge-2 (bigram), and Rouge-L (longest common subsequence, LCS). In this paper, we also report these three automatic evaluation results following previous works [14]–[16].

E. Results

Our main results on the *CNN/Daily Mail* dataset are shown in Table III. It includes the automatic evaluation results using full length Rouge-F1[8] metric. We report the results of two unsupervised baselines, i.e., LEAD3 and TextRank. We also report several state-of-the-art abstractive summarization models, i.e., Pointer-Generator Network (PGN), DCA, Bottom-Up, JECS and SRC-ELMO as a reference. We compare the NeuSum$_{\text{RNN}}$ model against the state-of-the-art extractive models without pre-training, namely, CRSum, NN-SE, Refresh, BanditSum and SummaRuNNer. For the pre-training based extractive models,

[8]The ROUGE evaluation option is, -m -n 2

TABLE III
EVALUATION RESULTS OF NEUSUM AND BASELINES ON *CNN/DAILY MAIL* TEST SET WITH FULL LENGTH ROUGE F1 METRICS (%). RESULTS WITH ‡ MARK ARE TAKEN FROM THE CORRESPONDING PAPERS

Model	ROUGE-1	ROUGE-2	ROUGE-L
Unsupervised / Abstractive System			
LEAD3 ([14], [13], [16], [19])	40.24	17.70	36.45
TEXTRANK (Mihalcea and Tarau [3])	40.20	17.56	36.44
PGN (See et al. [14])‡	39.53	17.28	36.38
DCA‡ (Celikyilmaz et al. [33])	41.69	19.47	37.92
JECS‡ (Xu and Durrett [29])	41.70	18.50	37.9
BOTTOM-UP (Gehrmann et al. [28])‡	41.22	18.68	38.34
SRC-ELMO (Edunov et al. [34])‡	41.56	18.94	38.47
Extractive System without Pre-training			
CRSUM (Ren et al. [8])	40.52	18.08	36.81
REFRESH (Narayan et al. [35])‡	40.00	18.20	36.60
SUMMARUNNER‡ (Nallapati et al. [16])	39.60	16.20	35.30
BANDITSUM‡ (Dong et al. [27])	41.50	18.70	37.60
NN-SE (Cheng and Lapata [15])	41.13	18.59	37.40
NEUSUM$_{\text{RNN}}$	**41.59**	**19.01**	**37.98**
Pre-training Based Extractive System			
BERT‡ ([19], [20])	42.57	19.96	39.04
SELF-SUPERVISED‡ (Wang et al. [36])	41.36	19.20	37.86
HIBERT‡ (Zhang et al. [19])	42.10	19.70	38.53
NEUSUM$_{\text{BERT}}$	**43.51**	**20.43**	**39.89**

TABLE IV
HUMAN EVALUATION RESULTS OF BERT BASELINE AND NEUSUM$_{\text{BERT}}$ (LOWER IS BETTER)

Model	Informativeness	Redundancy	Overall
BERT	1.36	1.38	1.36
NEUSUM$_{\text{BERT}}$	1.14	1.08	1.12

TABLE V
EVALUATION RESULTS OF NEUSUM AND BASELINES ON *NEW YORK TIMES* TEST SET WITH FULL LENGTH ROUGE F1 METRICS (%)

Model	ROUGE-1	ROUGE-2	ROUGE-L
LEAD3	31.17	15.59	27.86
TEXTRANK	32.38	16.27	28.93
CRSUM	31.46	15.50	27.96
NN-SE	36.66	19.88	32.97
NEUSUM$_{\text{RNN}}$	**37.71**	**20.49**	**33.92**
BERT	36.85	20.19	33.27
NEUSUM$_{\text{BERT}}$	**37.43**	**20.60**	**33.77**

we report the results of BERT, SELF-SUPERVISED and HIBERT and compare them with our NEUSUM$_{\text{BERT}}$ model.

As shown in Table III, our NEUSUM$_{\text{RNN}}$ model outperforms the unsupervised methods such as LEAD3 and TEXTRANK by a large margin. Specifically, NEUSUM$_{\text{RNN}}$ achieves 19.01 ROUGE-2 F1 score on the *CNN/Daily Mail* dataset. NEUSUM$_{\text{RNN}}$ also outperforms previously published extractive summarization methods such as CRSUM, NN-SE, REFRESH and SUMMARUNNER. The CRSUM and SUMMARUNNER baseline models leverage shallow features such as sentence position, which are have proven effective in extractive document summarization [8], [16]. Without any hand-crafted features, NEUSUM$_{\text{RNN}}$ performs better than these baseline systems. Compared to the RNN-based models whose sentence scoring and selection are separated, our NEUSUM$_{\text{RNN}}$ outperforms the previous state-of-the-art extractive method NN-SE, which indicates that joint sentence scoring and selection is useful in extractive document summarization. REFRESH [35] employ Reinforcement Learning to directly optimize the ROUGE scores. However, their method still falls into the separated sentence scoring and selection and paradigm. NEUSUM$_{\text{RNN}}$ outperforms REFRESH by +0.81 ROUGE-2 and +1.38 ROUGE-L points. As given by the 95% confidence interval in the official ROUGE script, NEUSUM$_{\text{RNN}}$ achieves statistically significant improvements over these baseline models without pre-training.

The performance of pre-training based models is also included in Table III. Three recent works [19], [20], [36] have discussed using pre-trained models for extractive document summarization. The ROUGE scores show that by leveraging pre-training methods, extractive summarization models can make large improvements. The pre-training based baseline methods, i.e., BERT, SELF-SUPERVISED and HIBERT, outperform the previous state-of-the-art systems without pre-training. By jointly scoring and selecting sentences, our NEUSUM$_{\text{BERT}}$ model surpasses these pre-training based methods. NEUSUM$_{\text{BERT}}$ outperforms BERT by +0.94 ROUGE-1, +0.47 ROUGE-2 and +0.85 ROUGE-L F1 which are statistically significant according to the official ROUGE script. To the best of our knowledge, the proposed NEUSUM$_{\text{BERT}}$ model achieves the best results among the published extractive models on the *CNN/Daily Mail* dataset.

We further conduct human evaluation with 50 randomly sampled documents from NEUSUM test set to manually check the quality of the output summaries of BERT baseline and our NEUSUM$_{\text{BERT}}$ model. The human judges are asked to rank the model output summaries from best to worst (with ties allowed) regarding the informativeness, redundancy, and overall quality. The evaluation results are listed in Table IV. The results show that NEUSUM$_{\text{BERT}}$ performs better than BERT on all the evaluation aspects, which shows similar trends as the human evaluation of NEUSUM$_{\text{RNN}}$ and NN-SE in our preliminary work [13]. The results indicate that NEUSUM$_{\text{BERT}}$ can choose more important sentences while extracting less redundant sentences. We credit this improvement to the joint sentence scoring and selection since it enables NEUSUM$_{\text{BERT}}$ to re-estimate the importance of remaining sentences considering both their contents and previously selected sentences.

Results on the *New York Times* corpus show similar trends. The evaluation metric is full length ROUGE F1[9] score. Table V summarizes the evaluation results of NEUSUM and several baselines. NEUSUM$_{\text{RNN}}$ outperforms NN-SE by +0.61 ROUGE-2 and +0.95 ROUGE-L F1. In the meanwhile, NEUSUM$_{\text{BERT}}$ performs better than BERT by +0.41 ROUGE-2 and +0.50 ROUGE-L F1. The results show that the joint sentence scoring and selection models (NEUSUM$_{\text{RNN}}$ and NEUSUM$_{\text{BERT}}$) outperforms their separate counterparts (NN-SE and BERT).

V. DISCUSSION

A. The Effect of the KL Training Objective

To show the effectiveness of the proposed training objective for NEUSUM framework, we conduct an additional test that

[9]The ROUGE evaluation option is, -m -n 2

TABLE VI
EVALUATION RESULTS OF NEUSUM$_{\text{BERT}}$ AND NEUSUM$_{\text{BERT}}$ WITH ONE-HOT CROSS ENTROPY LOSS (NEUSUM$_{\text{BERT}}$-XENT) ON *CNN/DAILY MAIL* TEST SET WITH FULL LENGTH ROUGE F1 METRICS (%)

Model	ROUGE-1	ROUGE-2	ROUGE-L
BERT‡ ([19], [20])	42.57	19.96	39.04
NEUSUM$_{\text{BERT}}$	**43.51**	**20.43**	**39.89**
NEUSUM$_{\text{BERT}}$-XENT	43.34	20.29	39.73

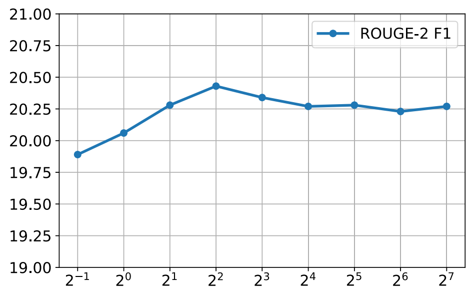

Fig. 3. ROUGE-2 F1 score with respect to temperature τ (logarithmic scale applied to the x-axis) in the NEUSUM$_{\text{BERT}}$ model.

we replace the KL objective with softmax cross entropy with one-hot distribution (XENT) objective. Specifically, at each step, the training supervision is a one-hot vector instead of the probability in the KL divergence. For example, if the KL training supervision is $[0.1, 0.1, 0.2, 0.1, 0.5]$, the training label of the XENT will be $[0, 0, 0, 0, 1]$. In fact, cross entropy with one-hot distribution is a variant of the KL objective. Detailed analysis in Section V-B shows the relationship between the one-hot cross entropy loss and KL divergence. We conduct experiment on the *CNN/Daily Mail* benchmark dataset and Table VI shows the ROUGE evaluation results.

By replacing the KL objective with XENT objective, the performance drops with -0.14 ROUGE-2 F1, -0.17 ROUGE-1 F1 and -0.16 ROUGE-L F1 compared with NEUSUM$_{\text{BERT}}$. The NEUSUM$_{\text{BERT}}$ with XENT objective (NEUSUM$_{\text{BERT}}$-XENT) still outperforms the BERT baseline. This shows the effectiveness of the proposed NEUSUM framework since NEUSUM$_{\text{BERT}}$-XENT adopts the joint sentence scoring and selection method while BERT baseline does not.

This experiment also shows that the KL divergence objective benefits the NEUSUM framework. Different from the cross entropy loss function which uses "should select" or "should not select" labels, the KL loss function assigns each sentence an importance score. This enables NEUSUM to learn a better sentence scorer without ignoring any potentially important sentences. Therefore, NEUSUM$_{\text{BERT}}$ performs better than NEUSUM$_{\text{BERT}}$-XENT.

B. The Effect of Temperature τ

We also study the effect of the hyper-parameter, temperature τ, in Equation (18). We train several NEUSUM$_{\text{BERT}}$ models with nine τ values, i.e., $[2^{-1}, 2^0, 2^1, \ldots, 2^7]$. We pick the τ value according to the performance on the development set, and the best τ value is $2^2 = 4$. The relationship is shown in Fig. 3, a plot of $\log_2 \tau$ and the NEUSUM$_{\text{BERT}}$ performance in terms of ROUGE-2 F1.

We observe that the performance of NEUSUM$_{\text{BERT}}$ increases when $\tau = 2^{-1}$ to $\tau = 2^2$, and then slowly decreases after $\tau \geq 2^3$. We therefore assume that the best performance appears when $2 \leq \tau \leq 8$. Here we give an intuitive explanation about why the performance changes with respect to temperature τ. With a smaller value for τ, the training label Q in Equation (18) is a softer distribution which might be a weak supervision. One extreme situation is $\tau = 0$, that Q becomes a uniform distribution which provides no supervision at all. When τ is large, the softmax distribution becomes sharp which might ignore the sentences with small scores. One extreme situation is $\tau \to \infty$, that the supervision becomes a one-hot vector like $[0, 0, 0, 0, 1]$ in the XENT loss. Therefore, a proper value of τ is needed to construct a good training supervision distribution.

There also exists a lower bound of the model performance when increasing the temperature τ, which is equal to the model performance using XENT loss. It can be observed in Fig. 3 that the ROUGE score almost remains unchanged after $\tau \geq 2^4$. Since the Q and P in Equation (19) are both discrete variables, the KL divergence is defined as:

$$\begin{aligned} D_{\text{KL}}(Q \parallel P) &= -\sum_{x \in \mathcal{X}} Q(x) \log \left(\frac{P(x)}{Q(x)} \right) \\ &= \sum_x Q(x) \log Q(x) - \sum_x Q(x) \log P(x) \\ &= -H(Q) + H(Q, P) \end{aligned}$$

where $H(Q)$ is the entropy of Q, $H(Q, P)$ is the cross entropy for Q and P, and Q is the training label in Equation (18):

$$Q(S_i) = \frac{\exp\left(\tau \widetilde{g}(S_i)\right)}{\sum_{k=1}^{L} \exp\left(\tau \widetilde{g}(S_k)\right)}. \tag{21}$$

As τ approaches infinity, the output of softmax function converges to a one-hot distribution (vector). The entropy $H(Q)$ is zero and $D_{\text{KL}}(Q \parallel P)$ then equals to the cross entropy of Q and P. We already analyzed the model performance of cross entropy objective in Section V-A. The result of NEUSUM$_{\text{BERT}}$-XENT is very close to the performance in Fig. 3 when τ is large.

VI. RELATED WORK

Automatic text summarization has been extensively studied for years. Dating back to the 1950 s, Luhn [6] uses lexical frequency to determine the importance of a sentence and then extracts sentences to form summaries. As an effective method, extractive summarization approaches are popular and dominate the summarization research. Previous extractive document summarization systems decompose this task into two steps, i.e., sentence scoring and sentence selection. Sentence scoring is the first step and critical since it is used to measure the importance of a sentence. Based on the output of the sentence scoring step, sentence selection determines the sentences to be extracted, which is usually done heuristically.

Many previous works focus on the modeling and scoring of document sentences. Unsupervised methods have been studied for decades since they do not need large-scale annotated datasets or model training. In these methods, some surface lexical features are proven useful, such as word term frequency [6], TF-IDF weights [9], sentence positions and lengths [5], [7], [8]. These surface features can be used alone or combined with weights.

Graph-based approaches [3], [9], [40] are also widely applied to sentence scoring. In these approaches, the input document is converted to a weighted-graph. In this graph, the vertices represent the document sentences, and the edges between two vertices have attached weights which show the similarity of the corresponding two sentences. Therefore, the importance score is indicated by its corresponding vertex. Different algorithms are proposed to calculate the importance scores of vertices, such as TextRank [3] and LexRank [9].

Approaches based on machine learning techniques are also studied since they can learn better sentence modeling based on surface features. Kupiec *et al.* [41] propose to use a Naive Bayes classifier to learn feature combination weights. Conroy and Oleary [42] use a Hidden Markov Model which has two states, i.e., summary sentence and non-summary sentence, in extractive document summarization. Gillick and Favre [43] find that for ROUGE [18] metrics, using bigram features yields better results than unigram or trigram features.

Having the sentence importance scores, sentence selection aims to produce summary under certain constraints. Maximal Marginal Relevance (MMR) [2] is proposed for document reordering in Information Retrieval and passage selection in document summarization. Summarization systems adopting the MMR strategy select the sentence which has the maximal importance score and is minimally redundant with previously selected sentences. As the main advantage of MMR is to reduce redundancy, it is widely used in multi-document summarization. Yogatama *et al.* [44] propose to maximize semantic volume instead of bigram coverage. McDonald [4] treats extractive document summarization as an optimization problem subject to some constraints such as the summary length limit. Therefore, sentence selection can also be seen as the problem of searching the optimal sentence subset. McDonald [4] proposes to solve this optimization problem using Integer Linear Programming. However, solving this optimization problem might be very slow since it is NP-hard. Lin and Bilmes [10] prove that ROUGE recall is monotone submodular so that they propose to apply submodular functions to find the subset.

Recently, deep neural network methods [8], [15], [16], [27], [29], [45], [46] have proven effective for extractive document summarization. Cao *et al.* [45] propose PriorSum, which uses convolutional neural networks to encode the summary prior features from length-variable phrases. Ren *et al.* [8] apply a two-level attention mechanism [47] to model the contextual relation between sentences. Cheng and Lapata [15] treat extractive document summarization as a sequence labeling task following [42]. They first encode the document sentences and then predict an *extract* or *not-extract* label for each sentence. Nallapati *et al.* [16] follow this line and propose SummaRuNNer with more lexical features. These two works are both in the separated paradigm, since they first predict an extraction probability score to each sentence (sentence scoring), and then sort the sentences to select the top sentences (sentence selection). Ren *et al.* [48] propose using two separate neural networks based on handcrafted features. One network scores the sentences to extract the first sentence, and the other one is used to model the redundancy in the sentence selection step. However, their redundancy model only considers the sentence that has the maximal score, which makes it unable to track all the selection history.

VII. CONCLUSION

In this work, we present a joint sentence scoring and selection framework for extractive document summarization. The most distinguishing feature of the proposed framework from previous approaches is that it combines sentence scoring and selection into one phase. Every time it selects a sentence, it scores the sentences according to the current extraction state and the partial output summary. Experiments with two network architectures on two datasets show that the proposed joint framework improves the performance of extractive summarization system compared with previous separated methods.

ACKNOWLEDGMENT

We would like to thank Danqing Huang, Fenfei Guo, Zhirui Zhang, Shuangzhi Wu, Chuanqi Tan, and Wei Jia for the helpful discussions. We thank all the anonymous reviewers for their helpful comments.

REFERENCES

[1] A. Nenkova and K. McKeown, “Automatic summarization,” *Found. Trends Inf. Retriev.*, vol. 5, no. 2–3, pp. 103–233, 2011.
[2] J. Carbonell and J. Goldstein, “The use of MMR, diversity-based reranking for reordering documents and producing summaries,” in *Proc. 21st Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retriev.*, 1998, pp. 335–336.
[3] R. Mihalcea and P. Tarau, “Textrank: Bringing order into text,” in *Proc. Conf. Empir. Methods Natural Lang. Process.*, 2004, pp. 404–411.
[4] R. McDonald, “A study of global inference algorithms in multi-document summarization,” in *Proc. Eur. Conf. Inf. Retriev.*, 2007, pp. 557–564.
[5] Z. Cao, F. Wei, L. Dong, S. Li, and M. Zhou, “Ranking with recursive neural networks and its application to multi-document summarization.” in *Proc. AAAI Conf. Artif. Intell.*, 2015, pp. 2153–2159.
[6] H. P. Luhn, “The automatic creation of literature abstracts,” *IBM J. Res. Develop.*, vol. 2, no. 2, pp. 159–165, 1958.
[7] E. Hovy and C.-Y. Lin, “Automated text summarization and the summarist system,” in *Proc. Workshop Held Baltimore, Maryland*, 1998, pp. 197–214.
[8] P. Ren, Z. Chen, Z. Ren, F. Wei, J. Ma, and M. de Rijke, “Leveraging contextual sentence relations for extractive summarization using a neural attention model,” in *Proc. 40th Int. ACM SIGIR Conf. Res. Develop. Inf. Retriev.*, New York, NY, USA, 2017, pp. 95–104.
[9] G. Erkan and D. R. Radev, “LexRank: Graph-based lexical centrality as salience in text summarization,” *J. Artif. Intell. Res.*, vol. 22, pp. 457–479, 2004.
[10] H. Lin and J. Bilmes, “A class of submodular functions for document summarization,” in *Proc. 49th Annu. Meet. Assoc. Comput. Linguist.: Human Lang. Technol. Volume 1*, 2011, pp. 510–520.
[11] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding,” in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguist.: Human Lang. Technol., Volume 1 (Long and Short Papers)*, Minneapolis, Minnesota, Jun. 2019, pp. 4171–4186.
[12] J. Lei Ba, J. R. Kiros, and G. E. Hinton, “Layer normalization,” in *Proc. NIPS Deep Learning Symp.*, 2016.
[13] Q. Zhou, N. Yang, F. Wei, S. Huang, M. Zhou, and T. Zhao, “Neural document summarization by jointly learning to score and select sentences,” in *Proc. 56th Annu. Meet. Assoc. Comput. Linguist. (Volume 1: Long Papers)*, 2018, pp. 654–663.

[14] A. See, P. J. Liu, and C. D. Manning, "Get to the point: Summarization with pointer-generator networks," in *Proc. 55th Annu. Meet. Assoc. Comput. Linguist. (Volume 1: Long Papers)*, Vancouver, Canada, Jul. 2017, pp. 1073–1083.

[15] J. Cheng and M. Lapata, "Neural summarization by extracting sentences and words," in *Proc. 54th Annu. Meet. Assoc. Comput. Linguist. (Volume 1: Long Papers)*, Berlin, Germany, Aug. 2016, pp. 484–494.

[16] R. Nallapati, F. Zhai, and B. Zhou, "Summarunner: A recurrent neural network based sequence model for extractive summarization of documents." in *Proc. AAAI Conf. Artif. Intell.*, 2017, pp. 3075–3081.

[17] R. Nallapati, B. Zhou, Ç. Gulçehre, and B. Xiang, "Abstractive text summarization using sequence-to-sequence RNNs and beyond," in *Proc. 20th SIGNLL Conf. Comput. Natural Lang. Learn.*, 2016, pp. 280–290.

[18] C.-Y. Lin, "Rouge: A package for automatic evaluation of summaries," in *Proc. ACL Workshop Text Summarization Branches Out*, Barcelona, Spain, 2004, vol. 8, pp. 74–81.

[19] X. Zhang, F. Wei, and M. Zhou, "HIBERT: Document level pre-training of hierarchical bidirectional transformers for document summarization," in *Proc. 57th Annu. Meet. Assoc. Comput. Linguist.*, Florence, Italy, Jul. 2019, pp. 5059–5069.

[20] Y. Liu, "Fine-tune BERT for extractive summarization," 2019, *arXiv:1903.10318*.

[21] A. Adhikari, A. Ram, R. Tang, and J. Lin, "Docbert: BERT for document classification," 2019, *arXiv:1904.08398*.

[22] A. Vaswani *et al.*, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.

[23] H. Inan, K. Khosravi, and R. Socher, "Tying word vectors and word classifiers: A loss framework for language modeling," in *Proc. 5th Int. Conf. Learn. Represent.*, 2017.

[24] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," in *Proc. NIPS Deep Learn. Represent. Learn. Workshop*, 2015.

[25] K. M. Hermann *et al.*, "Teaching machines to read and comprehend," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 1693–1701.

[26] G. Durrett, T. Berg-Kirkpatrick, and D. Klein, "Learning-based single-document summarization with compression and anaphoricity constraints," in *Proc. 54th Annu. Meet. Assoc. Comput. Linguist. (Volume 1: Long Papers)*, 2016, pp. 1998–2008.

[27] Y. Dong, Y. Shen, E. Crawford, H. van Hoof, and J. C. K. Cheung, "BanditSum: Extractive summarization as a contextual bandit," in *Proc. Conf. Empir. Methods Natural Lang. Process.*, Brussels, Belgium, Oct./Nov. 2018, pp. 3739–3748.

[28] S. Gehrmann, Y. Deng, and A. Rush, "Bottom-up abstractive summarization," in *Proc. Conf. Empir. Methods Natural Lang. Process.*, Brussels, Belgium, Oct./Nov. 2018, pp. 4098–4109.

[29] J. Xu and G. Durrett, "Neural extractive text summarization with syntactic compression," in *Proc. Conf. Empir. Methods Natural Lang. Process. 9th Int. Joint Conf. Natural Lang. Process.*, Hong Kong, China, Nov. 2019, pp. 3290–3301.

[30] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. Int. Conf. Artif. Intell. Statist.*, vol. 9, 2010, pp. 249–256.

[31] A. Paszke *et al.*, "Automatic differentiation in PyTorch," *NIPS 2017 Autodiff Workshop*, 2017.

[32] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Represent.*, San Diego, 2015.

[33] A. Celikyilmaz, A. Bosselut, X. He, and Y. Choi, "Deep communicating agents for abstractive summarization," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguist.: Human Lang. Technol., Volume 1 (Long Papers)*, New Orleans, Louisiana, Jun. 2018, pp. 1662–1675.

[34] S. Edunov, A. Baevski, and M. Auli, "Pre-trained language model representations for language generation," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguist.: Human Lang. Technol., Volume 1 (Long and Short Papers)*, Minneapolis, Minnesota, Jun. 2019, pp. 4052–4059.

[35] S. Narayan, S. B. Cohen, and M. Lapata, "Ranking sentences for extractive summarization with reinforcement learning," in *Proc. North Amer. Chapter Assoc. Comput Linguist.*, 2018, pp. 1747–1759.

[36] H. Wang *et al.*, "Self-supervised learning for contextualized extractive summarization," in *Proc. 57th Annu. Meet. Assoc. Comput. Linguist.*, Florence, Italy, Jul. 2019, pp. 2221–2227.

[37] R. Řehůřek and P. Sojka, "Software framework for topic modelling with large corpora," in *Proc. LREC Workshop New Challenges NLP Frameworks*, Valletta, Malta, May 2010, pp. 45–50.

[38] J. Gu, Z. Lu, H. Li, and V. O. Li, "Incorporating copying mechanism in sequence-to-sequence learning," in *Proc. Assoc. Comput. Linguist.*, Berlin, Germany, Aug. 2016, pp. 1631–1640.

[39] C. Gulcehre, S. Ahn, R. Nallapati, B. Zhou, and Y. Bengio, "Pointing the unknown words," in *Proc. Assoc. Comput. Linguist.*, Berlin, Germany, Aug. 2016, pp. 140–149.

[40] X. Wan and J. Yang, "Improved affinity graph based multi-document summarization," in *Proc. Human Lang. Technology Conf. NAACL, Companion Volume Short Papers*, 2006, pp. 181–184.

[41] J. Kupiec, J. Pedersen, and F. Chen, "A trainable document summarizer," in *Proc. 18th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retriev.*, 1995, pp. 68–73.

[42] J. M. Conroy and D. P. O'leary, "Text summarization via hidden markov models," in *Proc. 24th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retriev.*, 2001, pp. 406–407.

[43] D. Gillick and B. Favre, "A scalable global model for summarization," in *Proc. Workshop Integer Linear Program. Natural Lang. Process.*, 2009, pp. 10–18.

[44] D. Yogatama, F. Liu, and N. A. Smith, "Extractive summarization by maximizing semantic volume," in *Proc. Conf. Empir. Methods Natural Lang. Process.*, Lisbon, Portugal, Sep. 2015, pp. 1961–1966.

[45] Z. Cao, F. Wei, S. Li, W. Li, M. Zhou, and W. Houfeng, "Learning summary prior representation for extractive summarization," in *Proc. 53rd Annu. Meet. Assoc. Comput. Linguist. 7th Int. Joint Conf. Natural Lang. Process. (Volume 2: Short Papers)*, vol. 2, 2015, pp. 829–833.

[46] C. Kedzie, K. McKeown, and H. Daumé III, "Content selection in deep learning models of summarization," in *Proc. Conf. Empir. Methods Natural Lang. Process.*, Brussels, Belgium, Oct./Nov. 2018, pp. 1818–1828.

[47] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *Proc. 3rd Int. Conf. Learn. Represent.*, San Diego, 2015.

[48] P. Ren, F. Wei, C. Zhumin, M. Jun, and M. Zhou, "A redundancy-aware sentence regression framework for extractive summarization," in *Proc. COLING 26th Int. Conf. Comput. Linguist.: Tech. Papers*, 2016, pp. 33–43.

Qingyu Zhou received the master's degree in July 2016 from the Department of Computer Science, Harbin Institute of Technology, Harbin, China, where he is currently working toward the Ph.D. degree. His current research interests include text summarization, natural language generation, and natural language processing.

Nan Yang received the Ph.D. degree from the University of Science and Technology of China, Hefei, China, in June 2014. He is a Lead Researcher with Natural Language Computing Group, Microsoft Research Asia, Beijing, China. His research interests include natural language processing, machine translation, and question answering.

Furu Wei received the Ph.D. degree from the Department of Computer Science, Wuhan University, Wuhan, China, in June 2009. He is a Senior Researcher with Natural Language Computing Group, Microsoft Research Asia, Beijing, China. Before joining MSRA-NLC in November 2010, he has been a staff researcher with IBM Research-China (IBM CRL) since July 2009. His research interests include natural language processing, information retrieval, and machine learning.

Shaohan Huang received the B.S. and M.S. degrees from Beihang University, Beijing, China. He is currently an Associate Researcher with Natural Language Computing Group, Microsoft Research Asia. His research interests include natural language generation, summarization, and anomaly detection.

Ming Zhou received the Ph.D. degree in computer science from Harbin Institute of Technology, Harbin, China, in 1991. He is a Principal Researcher and Manager of Microsoft Research Asia Natural Language Computing Group, Microsoft Research, Beijing, China. He came to Microsoft in 1999 from his post as an Associate Professor with the Department of Computer Science, Tsinghua University. He is an Expert in the areas of machine translation and natural language processing.

Tiejun Zhao received the Ph.D. degree from the Department of Computer Science, Harbin Institute of Technology, Harbin, China, in 1992. He is a Full Professor with the Department of Computer Science, and the Director of Machine Intelligence and Translation, Harbin Institute of Technology. His research interests include machine translation and question answering.