Neural Document Summarization by Jointly Learning to Score and Select Sentences

Qingyu Zhou[†,*] **Nan Yang**[‡], **Furu Wei**[‡], **Shaohan Huang**[‡], **Ming Zhou**[‡], **Tiejun Zhao**[†]
[†]Harbin Institute of Technology, Harbin, China
[‡]Microsoft Research, Beijing, China
{qyzhou,tjzhao}@hit.edu.cn
{nanya,fuwei,shaohanh,mingzhou}@microsoft.com

Abstract

Sentence scoring and sentence selection are two main steps in extractive document summarization systems. However, previous works treat them as two separated subtasks. In this paper, we present a novel end-to-end neural network framework for extractive document summarization by jointly learning to score and select sentences. It first reads the document sentences with a hierarchical encoder to obtain the representation of sentences. Then it builds the output summary by extracting sentences one by one. Different from previous methods, our approach integrates the selection strategy into the scoring model, which directly predicts the relative importance given previously selected sentences. Experiments on the CNN/Daily Mail dataset show that the proposed framework significantly outperforms the state-of-the-art extractive summarization models.

1 Introduction

Traditional approaches to automatic text summarization focus on identifying important content, usually at sentence level (Nenkova and McKeown, 2011). With the identified important sentences, a summarization system can extract them to form an output summary. In recent years, *extractive methods* for summarization have proven effective in many systems (Carbonell and Goldstein, 1998; Mihalcea and Tarau, 2004; McDonald, 2007; Cao et al., 2015a). In previous works that use extractive methods, text summarization is decomposed into two subtasks, i.e., sentence scoring and sentence selection.

Sentence scoring aims to assign an importance score to each sentence, and has been broadly studied in many previous works. Feature-based methods are popular and have proven effective, such as word probability, TF*IDF weights, sentence position and sentence length features (Luhn, 1958; Hovy and Lin, 1998; Ren et al., 2017). Graph-based methods such as TextRank (Mihalcea and Tarau, 2004) and LexRank (Erkan and Radev, 2004) measure sentence importance using weighted-graphs. In recent years, neural network has also been applied to sentence modeling and scoring (Cao et al., 2015a; Ren et al., 2017).

For the second step, *sentence selection* adopts a particular strategy to choose content sentence by sentence. Maximal Marginal Relevance (Carbonell and Goldstein, 1998) based methods select the sentence that has the maximal score and is minimally redundant with sentences already included in the summary. Integer Linear Programming based methods (McDonald, 2007) treat sentence selection as an optimization problem under some constraints such as summary length. Submodular functions (Lin and Bilmes, 2011) have also been applied to solving the optimization problem of finding the optimal subset of sentences in a document. Ren et al. (2016) train two neural networks with handcrafted features. One is used to rank sentences, and the other one is used to model redundancy during sentence selection.

In this paper, we present a neural extractive document summarization (NEUSUM) framework which jointly learns to score and select sentences. Different from previous methods that treat sentence scoring and sentence selection as two tasks, our method integrates the two steps into one end-to-end trainable model. Specifically, NEUSUM is a neural network model without any handcrafted features that learns to identify the relative importance of sentences. The relative importance is

[*] Contribution during internship at Microsoft Research.

Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Long Papers), pages 654–663
Melbourne, Australia, July 15 - 20, 2018. ©2018 Association for Computational Linguistics

measured as the gain over previously selected sentences. Therefore, each time the proposed model selects one sentence, it scores the sentences considering both sentence saliency and previously selected sentences. Through the joint learning process, the model learns to predict the relative gain given the sentence extraction state and the partial output summary.

The proposed model consists of two parts, i.e., the document encoder and the sentence extractor. The document encoder has a hierarchical architecture, which suits the compositionality of documents. The sentence extractor is built with recurrent neural networks (RNN), which provides two main functionalities. On one hand, the RNN is used to remember the partial output summary by feeding the selected sentence into it. On the other hand, it is used to provide a sentence extraction state that can be used to score sentences with their representations. At each step during extraction, the sentence extractor reads the representation of the last extracted sentence. It then produces a new sentence extraction state and uses it to score the relative importance of the rest sentences.

We conduct experiments on the *CNN/Daily Mail* dataset. The experimental results demonstrate that the proposed NEUSUM by jointly scoring and selecting sentences achieves significant improvements over separated methods. Our contributions are as follows:

- We propose a joint sentence scoring and selection model for extractive document summarization.
- The proposed model can be end-to-end trained without handcrafted features.
- The proposed model significantly outperforms state-of-the-art methods and achieves the best result on *CNN/Daily Mail* dataset.

2 Related Work

Extractive document summarization has been extensively studied for years. As an effective approach, extractive methods are popular and dominate the summarization research. Traditional extractive summarization systems use two key techniques to form the summary, sentence scoring and sentence selection. Sentence scoring is critical since it is used to measure the saliency of a sentence. Sentence selection is based on the scores of sentences to determine which sentence should be extracted, which is usually done heuristically.

Many techniques have been proposed to model and score sentences. Unsupervised methods do not require model training or data annotation. In these methods, many surface features are useful, such as term frequency (Luhn, 1958), TF*IDF weights (Erkan and Radev, 2004), sentence length (Cao et al., 2015a) and sentence positions (Ren et al., 2017). These features can be used alone or combined with weights.

Graph-based methods (Erkan and Radev, 2004; Mihalcea and Tarau, 2004; Wan and Yang, 2006) are also applied broadly to ranking sentences. In these methods, the input document is represented as a connected graph. The vertices represent the sentences, and the edges between vertices have attached weights that show the similarity of the two sentences. The score of a sentence is the importance of its corresponding vertex, which can be computed using graph algorithms.

Machine learning techniques are also widely used for better sentence modeling and importance estimation. Kupiec et al. (1995) use a Naive Bayes classifier to learn feature combinations. Conroy and O'leary (2001) further use a Hidden Markov Model in document summarization. Gillick and Favre (2009) find that using bigram features consistently yields better performance than unigrams or trigrams for ROUGE (Lin, 2004) measures.

Carbonell and Goldstein (1998) proposed the Maximal Marginal Relevance (MMR) method as a heuristic in sentence selection. Systems using MMR select the sentence which has the maximal score and is minimally redundant with previous selected sentences. McDonald (2007) treats sentence selection as an optimization problem under some constraints such as summary length. Therefore, he uses Integer Linear Programming (ILP) to solve this optimization problem. Sentence selection can also be seen as finding the optimal subset of sentences in a document. Lin and Bilmes (2011) propose using submodular functions to find the subset.

Recently, deep neural networks based approaches have become popular for extractive document summarization. Cao et al. (2015b) develop a novel summary system called PriorSum, which applies enhanced convolutional neural networks to capture the summary prior features derived from length-variable phrases. Ren et al. (2017) use

a two-level attention mechanism to measure the contextual relations of sentences. Cheng and Lapata (2016) propose treating document summarization as a sequence labeling task. They first encode the sentences in the document and then classify each sentence into two classes, i.e., extraction or not. Nallapati et al. (2017) propose a system called SummaRuNNer with more features, which also treat extractive document summarization as a sequence labeling task. The two works are both in the separated paradigm, as they first assign a probability of being extracted to each sentence, and then select sentences according to the probability until reaching the length limit. Ren et al. (2016) train two neural networks with handcrafted features. One is used to rank the sentences to select the first sentence, and the other one is used to model the redundancy during sentence selection. However, their model of measuring the redundancy only considers the redundancy between the sentence that has the maximal score, which lacks the modeling of all the selection history.

3 Problem Formulation

Extractive document summarization aims to extract informative sentences to represent the important meanings of a document. Given a document $\mathcal{D} = (S_1, S_2, \ldots, S_L)$ containing L sentences, an extractive summarization system should select a subset of $\mathcal{D}$ to form the output summary $\mathcal{S} = \{\hat{S}_i | \hat{S}_i \in \mathcal{D}\}$. During the training phase, the reference summary $\mathcal{S}^*$ and the score of an output summary $\mathcal{S}$ under a given evaluation function $r(\mathcal{S}|\mathcal{S}^*)$ are available. The goal of training is to learn a scoring function $f(\mathcal{S})$ which can be used to find the best 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 length limit of the output summary. In this paper, l is the sentence number limit.

Previous state-of-the-art summarization systems search the best solution using the learned scoring function $f(\cdot)$ with two methods, MMR and ILP. In this paper, we adopt the MMR method. Since MMR tries to maximize the relative gain given previous extracted sentences, we let the model to learn to score this gain. Previous works adopt ROUGE recall as the evaluation $r(\cdot)$ considering the DUC tasks have byte length limit for summaries. In this work, we adopt the *CNN/Daily Mail* dataset to train the neural network model, which does not have this length limit. To prevent the tendency of choosing longer sentences, we use ROUGE F1 as the evaluation function $r(\cdot)$, and set the length limit l as a fixed number of sentences.

Therefore, the proposed model is trained to learn a scoring function $g(\cdot)$ of the ROUGE F1 gain, specifically:

$$g(S_t|\mathbb{S}_{t-1}) = r\left(\mathbb{S}_{t-1} \cup \{S_t\}\right) - r(\mathbb{S}_{t-1}) \quad (1)$$

where $\mathbb{S}_{t-1}$ is the set of previously selected sentences, and we omit the condition $\mathcal{S}^*$ of $r(\cdot)$ for simplicity. At each time t, the summarization system chooses the sentence with maximal ROUGE F1 gain until reaching the sentence number limit.

4 Neural Document Summarization

Figure 1 gives the overview of NEUSUM, which consists of a hierarchical document encoder, and a sentence extractor. Considering the intrinsic hierarchy nature of documents, that words form a sentence and sentences form a document, we employ a hierarchical document encoder to reflect this hierarchy structure. The sentence extractor scores the encoded sentences and extracts one of them at each step until reaching the output sentence number limit. In this section, we will first introduce the hierarchical document encoder, and then describe how the model produces summary by joint sentence scoring and selection.

4.1 Document Encoding

We employ a hierarchical document encoder to represent the sentences in the input document. We encode the document in two levels, i.e., sentence level encoding and document level encoding. Given a document $\mathcal{D} = (S_1, S_2, \ldots, S_L)$ containing L sentences. The sentence level encoder reads the j-th input sentence $S_j = (x_1^{(j)}, x_2^{(j)}, \ldots, x_{n_j}^{(j)})$ and constructs the basic sentence representation $\widetilde{s}_j$. Here we employ a bidirectional GRU (BiGRU) (Cho et al., 2014) as the recurrent unit, where GRU is defined as:

$$z_i = \sigma(\mathbf{W}_z[x_i, h_{i-1}]) \quad (2)$$

$$r_i = \sigma(\mathbf{W}_r[x_i, h_{i-1}]) \quad (3)$$

$$\widetilde{h}_i = \tanh(\mathbf{W}_h[x_i, r_i \odot h_{i-1}]) \quad (4)$$

$$h_i = (1 - z_i) \odot h_{i-1} + z_i \odot \widetilde{h}_i \quad (5)$$

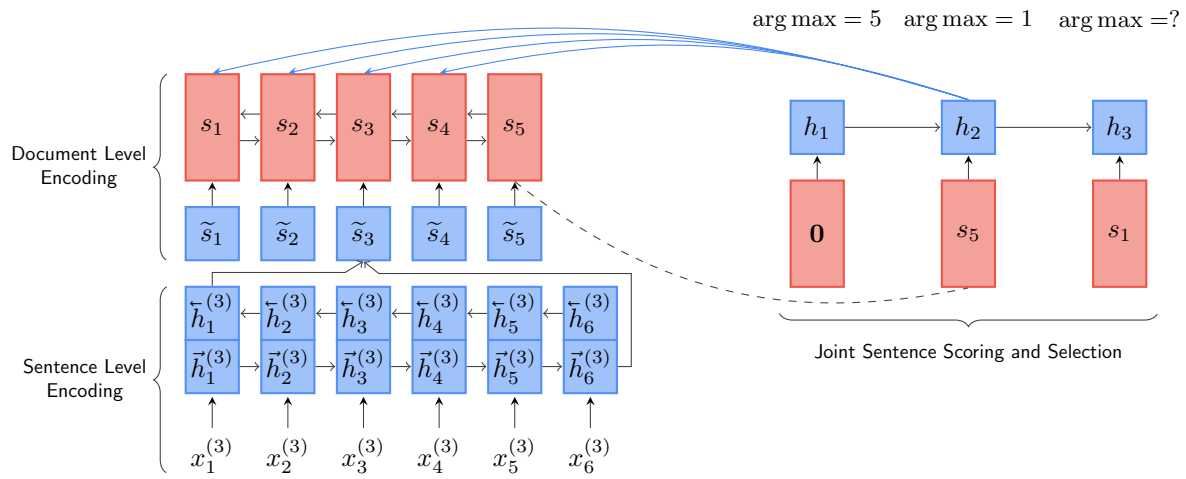

Figure 1: Overview of the NEUSUM model. The model extracts S_5 and S_1 at the first two steps. At the first step, we feed the model a zero vector $\mathbf{0}$ to represent empty partial output summary. At the second and third steps, the representations of previously selected sentences S_5 and S_1, i.e., s_5 and s_1, are fed into the extractor RNN. At the second step, the model only scores the first 4 sentences since the 5th one is already included in the partial output summary.

where $\mathbf{W}_z$, $\mathbf{W}_r$ and $\mathbf{W}_h$ are weight matrices.

The BiGRU consists of a forward GRU and a backward GRU. The forward GRU reads the word embeddings in sentence S_j from left to right and gets a sequence of hidden states, $(\vec{h}_1^{(j)}, \vec{h}_2^{(j)}, \ldots, \vec{h}_{n_j}^{(j)})$. The backward GRU reads the input sentence embeddings reversely, from right to left, and results in another sequence of hidden states, $(\overleftarrow{h}_1^{(j)}, \overleftarrow{h}_2^{(j)}, \ldots, \overleftarrow{h}_{n_j}^{(j)})$:

$$\vec{h}_i^{(j)} = \text{GRU}(x_i^{(j)}, \vec{h}_{i-1}^{(j)}) \quad (6)$$

$$\overleftarrow{h}_i^{(j)} = \text{GRU}(x_i^{(j)}, \overleftarrow{h}_{i+1}^{(j)}) \quad (7)$$

where the initial states of the BiGRU are set to zero vectors, i.e., $\vec{h}_1^{(j)} = 0$ and $\overleftarrow{h}_{n_j}^{(j)} = 0$.

After reading the words of the sentence S_j, we construct its sentence level representation $\widetilde{s}_j$ by concatenating the last forward and backward GRU hidden vectors:

$$\widetilde{s}_j = \begin{bmatrix} \overleftarrow{h}_1^{(j)} \\ \vec{h}_{n_j}^{(j)} \end{bmatrix} \quad (8)$$

We use another BiGRU as the document level encoder to read the sentences. With the sentence level encoded vectors $(\widetilde{s}_1, \widetilde{s}_2, \ldots, \widetilde{s}_L)$ as inputs, the document level encoder does forward and backward GRU encoding and produces two list of hidden vectors: $(\vec{s}_1, \vec{s}_2, \ldots, \vec{s}_L)$ and $(\overleftarrow{s}_1, \overleftarrow{s}_2, \ldots, \overleftarrow{s}_L)$. The document level representation s_i of sentence S_i is the concatenation of the forward and backward hidden vectors:

$$s_i = \begin{bmatrix} \vec{s}_i \\ \overleftarrow{s}_i \end{bmatrix} \quad (9)$$

We then get the final sentence vectors in the given document: $D = (s_1, s_2, \ldots, s_L)$. We use sentence S_i and its representative vector s_i interchangeably in this paper.

4.2 Joint Sentence Scoring and Selection

Since the separated sentence scoring and selection cannot utilize the information of each other, the goal of our model is to make them benefit each other. We couple these two steps together so that: a) sentence scoring can be aware of previously selected sentences; b) sentence selection can be simplified since the scoring function is learned to be the ROUGE score gain as described in section 3.

Given the last extracted sentence $\hat{S}_{t-1}$, the sentence extractor decides the next sentence $\hat{S}_t$ by scoring the remaining document sentences. To score the document sentences considering both their importance and partial output summary, the model should have two key abilities: 1) remembering the information of previous selected sentences; 2) scoring the remaining document sentences based on both the previously selected sentences and the importance of remaining sentences. Therefore, we employ another GRU as the recurrent unit to remember the partial output summary, and use a Multi-Layer Perceptron (MLP) to score

the document sentences. Specifically, the GRU takes the document level representation s_{t-1} of the last extracted sentence $\hat{S}_{t-1}$ as input to produce its current hidden state h_t. The sentence scorer, which is a two-layer MLP, takes two input vectors, namely the current hidden state h_t and the sentence representation vector s_i, to calculate the score $\delta(S_i)$ of sentence S_i.

$$h_t = \text{GRU}(s_{t-1}, h_{t-1}) \tag{10}$$

$$\delta(S_i) = \mathbf{W}_s \tanh\left(\mathbf{W}_q h_t + \mathbf{W}_d s_i\right) \tag{11}$$

where $\mathbf{W}_s$, $\mathbf{W}_q$ and $\mathbf{W}_d$ are learnable parameters, and we omit the bias parameters for simplicity.

When extracting the first sentence, we initialize the GRU hidden state h_0 with a linear layer with tanh activation function:

$$h_0 = \tanh\left(\mathbf{W}_m \overleftarrow{s}_1 + b_m\right) \tag{12}$$

$$S_0 = \varnothing \tag{13}$$

$$s_0 = \mathbf{0} \tag{14}$$

where$\mathbf{W}_m$ and b_m are learnable parameters, and $\overleftarrow{s}_1$ is the last backward state of the document level encoder BiGRU. Since we do not have any sentences extracted yet, we use a zero vector to represent the previous extracted sentence, i.e., $s_0 = \mathbf{0}$.

With the scores of all sentences at time t, we choose the sentence with maximal gain score:

$$\hat{S}_t = \arg\max_{S_i \in \mathcal{D}} \delta(S_i) \tag{15}$$

4.3 Objective Function

Inspired by Inan et al. (2017), we optimize the Kullback-Leibler (KL) divergence of the model prediction P and the labeled training data distribution Q. We normalize the predicted sentence score $\delta(S_i)$ with softmax function to get the model prediction distribution P:

$$P(\hat{S}_t = S_i) = \frac{\exp\left(\delta(S_i)\right)}{\sum_{k=1}^{L} \exp\left(\delta(S_k)\right)} \tag{16}$$

During training, the model is expected to learn the relative ROUGE F1 gain at time step t with previously selected sentences $\mathbb{S}_{t-1}$. Considering that the F1 gain value might be negative in the labeled data, we follow previous works (Ren et al., 2017) to use Min-Max Normalization to rescale the gain value to $[0, 1]$:

$$g(S_i) = r(\mathbb{S}_{t-1} \cup \{S_i\}) - r(\mathbb{S}_{t-1}) \tag{17}$$

$$\widetilde{g}(S_i) = \frac{g(S_i) - \min\left(g(S)\right)}{\max\left(g(S)\right) - \min\left(g(S)\right)} \tag{18}$$

We then apply a softmax operation with temperature τ (Hinton et al., 2015) [1] to produce the labeled data distribution Q as the training target. We apply the temperature τ as a smoothing factor to produce a smoothed label distribution Q:

$$Q(S_i) = \frac{\exp\left(\tau \widetilde{g}(S_i)\right)}{\sum_{k=1}^{L} \exp\left(\tau \widetilde{g}(S_k)\right)} \tag{19}$$

Therefore, we minimize the *KL* loss function J:

$$J = D_{KL}(P \parallel Q) \tag{20}$$

5 Experiments

5.1 Dataset

A large scale dataset is essential for training neural network-based summarization models. We use the *CNN/Daily Mail* dataset (Hermann et al., 2015; Nallapati et al., 2016) as the training set in our experiments. The *CNN/Daily Mail* news contain articles and their corresponding highlights. The highlights are created by human editors and are abstractive summaries. Therefore, the highlights are not ready for training extractive systems due to the lack of supervisions.

We create an extractive summarization training set based on *CNN/Daily Mail* corpus. To determine the sentences to be extracted, we design a rule-based system to label the sentences in a given document similar to Nallapati et al. (2017). Specifically, we construct training data by maximizing the ROUGE-2 F1 score. Since it is computationally expensive to find the global optimal combination of sentences, we employ a greedy approach. Given a document with n sentences, we enumerate the candidates from 1-combination $\binom{n}{1}$ to n-combination $\binom{n}{n}$. We stop searching if the highest ROUGE-2 F1 score in $\binom{n}{k}$ is less than the best one in $\binom{n}{k-1}$. Table 1 shows the data statistics of the *CNN/Daily Mail* dataset.

We conduct data preprocessing using the same method[2] in See et al. (2017), including sentence splitting and word tokenization. Both Nallapati et al. (2016, 2017) use the *anonymized* version of the data, where the named entities are replaced by identifiers such as `entity4`. Following See et al. (2017), we use the *non-anonymized* version so we can directly operate on the original text.

[1] We set $\tau = 20$ empirically according to the model performance on the development set.

[2] https://github.com/abisee/cnn-dailymail

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 1: Data statistics of *CNN/Daily Mail* dataset.

5.2 Implementation Details

Model Parameters The vocabulary is collected from the *CNN/Daily Mail* training data. We lowercase the text and there are 732,304 unique word types. We use the top 100,000 words as the model vocabulary since they can cover 98.23% of the training data. The size of word embedding, sentence level encoder GRU, document level encoder GRU are set to 50, 256, and 256 respectively. We set the sentence extractor GRU hidden size to 256.

Model Training We initialize the model parameters randomly using a Gaussian distribution with Xavier scheme (Glorot and Bengio, 2010). The word embedding matrix is initialized using pre-trained 50-dimension GloVe vectors (Pennington et al., 2014)[3]. We found that larger size GloVe does not lead to improvement. Therefore, we use 50-dim word embeddings for fast training. The pre-trained GloVe vectors contain 400,000 words and cover 90.39% of our model vocabulary. We initialize the rest of the word embeddings randomly using a Gaussian distribution with Xavier scheme. The word embedding matrix is not updated during training. We use Adam (Kingma and Ba, 2015) as our optimizing algorithm. For the hyperparameters of Adam optimizer, we set the learning rate $\alpha = 0.001$, two momentum parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$ respectively, and $\epsilon = 10^{-8}$. We also apply gradient clipping (Pascanu et al., 2013) with range $[-5, 5]$ during training. We use dropout (Srivastava et al., 2014) as regularization with probability $p = 0.3$ after the sentence level encoder and $p = 0.2$ after the document level encoder. We truncate each article to 80 sentences and each sentence to 100 words during both training and testing. The model is implemented with PyTorch (Paszke et al., 2017). We release the source code and related resources at https://res.qyzhou.me.

Model Testing At test time, considering that LEAD3 is a commonly used and strong extractive baseline, we make NEUSUM and the baselines extract 3 sentences to make them all comparable.

5.3 Baseline

We compare NEUSUM model with the following state-of-the-art baselines:

LEAD3 The commonly used baseline by selecting the first three sentences as the summary.

TEXTRANK An unsupervised algorithm based on weighted-graphs proposed by Mihalcea and Tarau (2004). We use the implementation in Gensim (Řehůřek and Sojka, 2010).

CRSUM Ren et al. (2017) propose an extractive summarization system which considers the contextual information of a sentence. We train this baseline model with the same training data as our approach.

NN-SE Cheng and Lapata (2016) propose an extractive system which models document summarization as a sequence labeling task. We train this baseline model with the same training data as our approach.

SUMMARUNNER Nallapati et al. (2017) propose to add some interpretable features such as sentence absolute and relative positions.

PGN Pointer-Generator Network (PGN). A state-of-the-art abstractive document summarization system proposed by See et al. (2017), which incorporates copying and coverage mechanisms.

5.4 Evaluation Metric

We employ ROUGE (Lin, 2004) as our evaluation metric. ROUGE measures the quality of summary by computing overlapping lexical units, such as unigram, bigram, trigram, and longest common subsequence (LCS). It has become the standard evaluation metric for DUC shared tasks and popular for summarization evaluation. Following previous work, we use ROUGE-1 (unigram), ROUGE-2 (bigram) and ROUGE-L (LCS) as the evaluation metrics in the reported experimental results.

[3] https://nlp.stanford.edu/projects/glove/

5.5 Results

We use the official ROUGE script[4] (version 1.5.5) to evaluate the summarization output. Table 2 summarizes the results on *CNN/Daily Mail* data set using full length ROUGE-F1[5] evaluation. It includes two unsupervised baselines, LEAD3 and TEXTRANK. The table also includes three state-of-the-art neural network based extractive models, i.e., CRSUM, NN-SE and SUMMARUNNER. In addition, we report the state-of-the-art abstractive PGN model. The result of SUMMARUNNER is on the *anonymized* dataset and not strictly comparable to our results on the *non-anonymized* version dataset. Therefore, we also include the result of LEAD3 on the *anonymized* dataset as a reference.

Models	**ROUGE-1**	**ROUGE-2**	**ROUGE-L**
LEAD3	40.24^{-}	17.70^{-}	36.45^{-}
TEXTRANK	40.20^{-}	17.56^{-}	36.44^{-}
CRSUM	40.52^{-}	18.08^{-}	36.81^{-}
NN-SE	41.13^{-}	18.59^{-}	37.40^{-}
PGN‡	39.53^{-}	17.28^{-}	36.38^{-}
LEAD3‡ *	39.2	15.7	35.5
SUMMARUNNER‡ *	39.6	16.2	35.3
NEUSUM	**41.59**	**19.01**	**37.98**

Table 2: Full length ROUGE F1 evaluation (%) on *CNN/Daily Mail* test set. Results with ‡ mark are taken from the corresponding papers. Those marked with * were trained and evaluated on the anonymized dataset, and so are not strictly comparable to our results on the original text. All our ROUGE scores have a 95% confidence interval of at most $\pm$0.22 as reported by the official ROUGE script. The improvement is statistically significant with respect to the results with superscript $^{-}$ mark.

NEUSUM achieves 19.01 ROUGE-2 F1 score on the *CNN/Daily Mail* dataset. Compared to the unsupervised baseline methods, NEUSUM performs better by a large margin. In terms of ROUGE-2 F1, NEUSUM outperforms the strong baseline LEAD3 by 1.31 points. NEUSUM also outperforms the neural network based models. Compared to the state-of-the-art extractive model NN-SE (Cheng and Lapata, 2016), NEUSUM performs significantly better in terms of ROUGE-1, ROUGE-2 and ROUGE-L F1 scores. Shallow features, such as sentence position, have proven effective in document summarization (Ren et al., 2017; Nallapati et al., 2017). Without any hand-crafted features, NEUSUM performs better than the CRSUM and SUMMARUNNER baseline models with features. As given by the 95% confidence interval in the official ROUGE script, our model achieves statistically significant improvements over all the baseline models. To the best of our knowledge, the proposed NEUSUM model achieves the best results on the *CNN/Daily Mail* dataset.

Models	Info	Rdnd	Overall
NN-SE	1.36	1.29	1.39
NEUSUM	**1.33**	**1.21**	**1.34**

Table 3: Rankings of NEUSUM and NN-SE in terms of informativeness (Info), redundancy (Rdnd) and overall quality by human participants (lower is better).

We also provide human evaluation results on a sample of test set. We random sample 50 documents and ask three volunteers to evaluate the output of NEUSUM and the NN-SE baseline models. They are asked to rank the output summaries from best to worst (with ties allowed) regarding informativeness, redundancy and overall quality. Table 3 shows the human evaluation results. NEUSUM performs better than the NN-SE baseline on all three aspects, especially in redundancy. This indicates that by jointly scoring and selecting sentences, NEUSUM can produce summary with less content overlap since it re-estimates the saliency of remaining sentences considering both their contents and previously selected sentences.

6 Discussion

6.1 Precision at Step-t

We analyze the accuracy of sentence selection at each step. Since we extract 3 sentences at test time, we show how NEUSUM performs when extracting each sentence. Given a document D in test set $\mathfrak{T}$, NEUSUM predicted summary $\mathcal{S}$, its reference summary $\mathcal{S}^*$, and the extractive oracle summary $\mathcal{O}$ with respect to D and $\mathcal{S}^*$ (we use the method described in section 5.1 to construct $\mathcal{O}$), we define the precision at step t as $p(@t)$:

$$p(@t) = \frac{1}{|\mathfrak{T}|} \sum_{D \in \mathfrak{T}} \mathbf{1}_{\mathcal{O}}(\mathcal{S}[t]) \qquad (21)$$

[4] http://www.berouge.com/

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

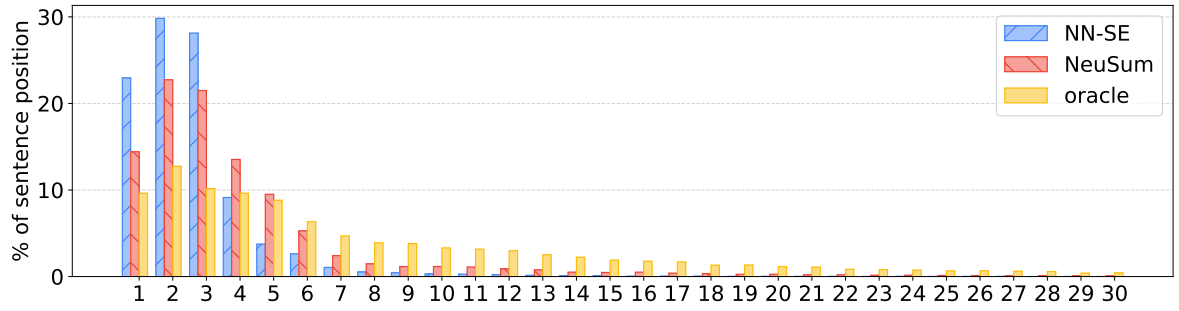

Figure 2: Position distribution of selected sentences of the NN-SE baseline, our NEUSUM model and oracle on the test set. We only draw the first 30 sentences since the average document length is 27.05.

where $\mathcal{S}[t]$ is the sentence extracted at step t, and $\mathbf{1}_{\mathcal{O}}$ is the indicator function defined as:

$$\mathbf{1}_{\mathcal{O}}(x) = \begin{cases} 1 & \text{if } x \in \mathcal{O} \\ 0 & \text{if } x \notin \mathcal{O} \end{cases} \quad (22)$$

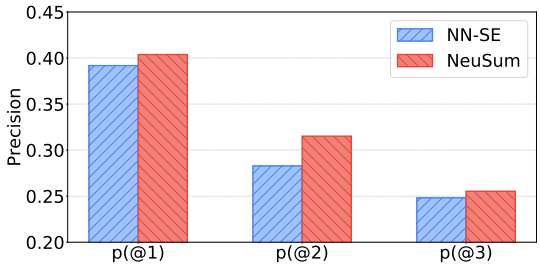

Figure 3: Precision of extracted sentence at step t of the NN-SE baseline and the NEUSUM model.

Figure 3 shows the precision at step t of NN-SE baseline and our NEUSUM. It can be observed that NEUSUM achieves better precision than the NN-SE baseline at each step. For the first sentence, both NEUSUM and NN-SE achieves good performance. The NN-SE baseline has 39.18% precision at the first step, and NEUSUM outperforms it by 1.2 points. At the second step, NEUSUM outperforms NN-SE by a large margin. In this step, the NEUSUM model extracts 31.52% sentences correctly, which is 3.24 percent higher than 28.28% of NN-SE. We think the second step selection benefits from the first step in NEUSUM since it can remember the selection history, while the separated models lack this ability.

However, we can notice the trend that the precision drops fast after each selection. We think this is due to two main reasons. First, we think that the error propagation leads to worse selection for the third selection. As shown in Figure 2, the $p(@1)$ and $p(@2)$ are 40.38% and 31.52% respectively, so the history is less reliable for the third selection. Second, intuitively, we think the later selections are more difficult compared to the previous ones since the most important sentences are already selected.

6.2 Position of Selected Sentences

Early works (Ren et al., 2017; Nallapati et al., 2017) have shown that sentence position is an important feature in extractive document summarization. Figure 2 shows the position distributions of the NN-SE baseline, our NEUSUM model and oracle on the *CNN/Daily Mail* test set. It can be seen that the NN-SE baseline model tends to extract large amount of leading sentences, especially the leading three sentences. According to the statistics, about 80.91% sentences selected by NN-SE baseline are in leading three sentences.

In the meanwhile, our NEUSUM model selects 58.64% leading three sentences. We can notice that in the oracle, the percentage of selecting leading sentences (sentence 1 to 5) is moderate, which is around 10%. Compared to NN-SE, the position of selected sentences in NEUSUM is closer to the oracle. Although NEUSUM also extracts more leading sentences than the oracle, it selects more tailing ones. For example, our NEUSUM model extracts more than 30% of sentences in the range of sentence 4 to 6. In the range of sentence 7 to 13, NN-SE barely extracts any sentences, but our NEUSUM model still extract sentences in this range. Therefore, we think this is one of the reasons why NEUSUM performs better than NN-SE.

We analyze the sentence position distribution and offer an explanation for these observations.

Intuitively, leading sentences are important for a well-organized article, especially for newswire articles. It is also well known that LEAD3 is a very strong baseline. In the training data, we found that 50.98% sentences labeled as "should be extracted" belongs to the first 5 sentences, which may cause the trained model tends to select more leading sentences. One possible situation is that one sentence in the tail of a document is more important than the leading sentences, but the margin between them is not large enough. The models which separately score and select sentences might not select sentences in the tail whose scores are not higher than the leading ones. These methods may choose the safer leading sentences as a fallback in such confusing situation because there is no direct competition between the leading and tailing candidates. In our NEUSUM model, the scoring and selection are jointly learned, and at each step the tailing candidates can compete directly with the leading ones. Therefore, NEUSUM can be more discriminating when dealing with this situation.

7 Conclusion

Conventional approaches to extractive document summarization contain two separated steps: sentence scoring and sentence selection. In this paper, we present a novel neural network framework for extractive document summarization by jointly learning to score and select sentences to address this issue. The most distinguishing feature of our approach from previous methods is that it combines sentence scoring and selection into one phase. Every time it selects a sentence, it scores the sentences according to the partial output summary and current extraction state. ROUGE evaluation results show that the proposed joint sentence scoring and selection approach significantly outperforms previous separated methods.

Acknowledgments

We thank three anonymous reviewers for their helpful comments. We also thank Danqing Huang, Chuanqi Tan, Zhirui Zhang, Shuangzhi Wu and Wei Jia for helpful discussions. The work of this paper is funded by the project of National Key Research and Development Program of China (No. 2017YFB1002102) and the project of National Natural Science Foundation of China (No. 91520204). The first author is funded by the Harbin Institute of Technology Scholarship Fund.

References

Ziqiang Cao, Furu Wei, Li Dong, Sujian Li, and Ming Zhou. 2015a. Ranking with recursive neural networks and its application to multi-document summarization. In *AAAI*, pages 2153–2159.

Ziqiang Cao, Furu Wei, Sujian Li, Wenjie Li, Ming Zhou, and WANG Houfeng. 2015b. Learning summary prior representation for extractive summarization. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 829–833.

Jaime Carbonell and Jade Goldstein. 1998. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 335–336. ACM.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 484–494, Berlin, Germany. Association for Computational Linguistics.

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. In *Proceedings of EMNLP 2014*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

John M Conroy and Dianne P O'leary. 2001. Text summarization via hidden markov models. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 406–407. ACM.

Günes Erkan and Dragomir R Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, 22:457–479.

Dan Gillick and Benoit Favre. 2009. A scalable global model for summarization. In *Proceedings of the Workshop on Integer Linear Programming for Natural Langauge Processing*, pages 10–18. Association for Computational Linguistics.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Aistats*, volume 9, pages 249–256.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Eduard Hovy and Chin-Yew Lin. 1998. Automated text summarization and the summarist system. In *Proceedings of a workshop on held at Baltimore, Maryland: October 13-15, 1998*, pages 197–214. Association for Computational Linguistics.

Hakan Inan, Khashayar Khosravi, and Richard Socher. 2017. Tying word vectors and word classifiers: A loss framework for language modeling. In *Proceedings of 5th International Conference for Learning Representations*.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of 3rd International Conference for Learning Representations*, San Diego.

Julian Kupiec, Jan Pedersen, and Francine Chen. 1995. A trainable document summarizer. In *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 68–73. ACM.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out: Proceedings of the ACL-04 workshop*, volume 8. Barcelona, Spain.

Hui Lin and Jeff Bilmes. 2011. A class of submodular functions for document summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 510–520. Association for Computational Linguistics.

Hans Peter Luhn. 1958. The automatic creation of literature abstracts. *IBM Journal of research and development*, 2(2):159–165.

Ryan McDonald. 2007. A study of global inference algorithms in multi-document summarization. In *European Conference on Information Retrieval*, pages 557–564. Springer.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *AAAI*, pages 3075–3081.

Ramesh Nallapati, Bowen Zhou, Ça glar Gulçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*.

Ani Nenkova and Kathleen McKeown. 2011. Automatic summarization. *Foundations and Trends® in Information Retrieval*, 5(2–3):103–233.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. *ICML (3)*, 28:1310–1318.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch.

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

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA.

Pengjie Ren, Zhumin Chen, Zhaochun Ren, Furu Wei, Jun Ma, and Maarten de Rijke. 2017. Leveraging contextual sentence relations for extractive summarization using a neural attention model. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 95–104, New York, NY, USA. ACM.

Pengjie Ren, Furu Wei, CHEN Zhumin, MA Jun, and Ming Zhou. 2016. A redundancy-aware sentence regression framework for extractive summarization. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 33–43.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Xiaojun Wan and Jianwu Yang. 2006. Improved affinity graph based multi-document summarization. In *Proceedings of the human language technology conference of the NAACL, Companion volume: Short papers*, pages 181–184. Association for Computational Linguistics.