Flexible Non-Autoregressive Extractive Summarization with Threshold: How to Extract a Non-Fixed Number of Summary Sentences

Ruipeng Jia,[1,2] **Yanan Cao,**[1,2] **Haichao Shi,**[1,2] **Fang Fang,**[1,2*] **Pengfei Yin,**[1,2] **Shi Wang**[3*]

[1] Institute of Information Engineering, Chinese Academy of Sciences
[2] School of Cyber Security, University of Chinese Academy of Sciences
[3] Institute of Computing Technology, Chinese Academy of Sciences
{jiaruipeng, caoyanan, shihaichao, fangfang0703, yinpengfei}@iie.ac.cn
wangshi@ict.ac.cn

Abstract

Sentence-level extractive summarization is a fundamental yet challenging task, and recent powerful approaches prefer to pick sentences sorted by the predicted probabilities until the length limit is reached, a.k.a. "Top-K Strategy". This length limit is fixed based on the validation set, resulting in the lack of flexibility. In this work, we propose a more flexible and accurate non-autoregressive method for single document extractive summarization, extracting a non-fixed number of summary sentences without the sorting step. We call our approach **ThresSum** as it picks sentences simultaneously and individually from the source document when the predicted probabilities exceed a threshold. During training, the model enhances sentence representation through iterative refinement and the intermediate latent variables receive some weak supervision with soft labels, which are generated progressively by adjusting the temperature with a knowledge distillation algorithm. Specifically, the temperature is initialized with high value and drops along with the iteration until a temperature of 1. Experimental results on CNN/DM and NYT datasets have demonstrated the effectiveness of ThresSum, which significantly outperforms BERTSUMEXT with a substantial improvement of 0.74 ROUGE-1 score on CNN/DM.

Introduction

Encoder-decoder mechanism is widely used for single document extractive summarization. The encoder is to encode one sentence into vector representation, while the popular decoder with top-k strategy can be divided into three steps: ***predict*** the probability scores of those sentence vectors, ***sort*** sentences in descending order in line with the probability scores, and ***pick*** sentences until exceeding the length limit. (Nallapati, Zhai, and Zhou 2017; Xiao and Carenini 2019; Liu and Lapata 2019; Xu et al. 2020). However, there are still two inherent obstacles for sentence-level extractive summarization:

1) Redundant phrases between selected sentences. The naive approach for the first prediction step of decoder is to make independent binary decisions for each sentence, leading to the absence of overlap or redundancy modeling between the selected target sentences (Xu et al. 2020; Zhong et al. 2020). The first natural solution is to introduce an autoregressive decoder (Chen and Bansal 2018; Jadhav and Rajan 2018; Liu and Lapata 2019; Xu et al. 2020), extracting sentences one by one and allowing different sentences to influence each other. Secondly, reinforcement learning is introduced for decoder to consider the semantics of the entire target summary (Narayan, Cohen, and Lapata 2018; Dong et al. 2018; Bae et al. 2019), which combines the maximum-likelihood cross-entropy loss with the rewards from policy gradient to directly optimize the evaluation metric for the summarization task. The third popular solution is to build summarization system with a two-stage decoder (Aliguliyev 2009; Galanis and Androutsopoulos 2010; Zhang et al. 2019a; Zhong et al. 2020), with the first stage to extract some fragments of the original text and the second stage to select or modify on the basis of these fragments. Unfortunately, all these approaches with an autoregressive decoder are unstable, for there are the train-inference gap and error propagation when extracting sentences one by one.

2) Fixed number or proportion of summary sentences. SummaRuNNer (Nallapati, Zhai, and Zhou 2017) first sets a "sort" step after the "predict" step, and "pick" sorted sentences until the length limit is reached (a.k.a. "Top-K Strategy"), which has been a popular method employed by the following models (Narayan, Cohen, and Lapata 2018; Zhang et al. 2018; Zhang, Wei, and Zhou 2019; Liu and Lapata 2019; Xu et al. 2020; Wang et al. 2020). While a flexible extractor should generate a non-fixed number of summary sentences based on source document length, topics, or other aspects. The reasonable approach is to pick sentences whose predicted probability is over a threshold. However, it may not be an optimal stratety since the training data is very imbalanced in terms of summary-membership of sentences (Nallapati, Zhai, and Zhou 2017). Furthermore, Mendes et al. (2019) introduces the length variable into the decoder and Zhong et al. (2020) can choose any number of sentences by matching candidate summary in semantic space. But the decoders of these two solutions are either autoregressive or two-stage.

To address the above two obstacles, we introduce ThresSum, a heuristic approach to strengthen the encoder by enhancing sentence representation through iterative refinement and simplify the decoder by removing the "sort" step:

*Corresponding authors: Fang Fang and Shi Wang

- For the encoder, we first map textual tokens into hidden states by contextualized interactions. Secondly, the sentence embedding is extracted through hierarchical attention to adaptively aggregate information from its word elements. Finally, we fuse the redundant information between selected sentences by iterative refinement and this process is supervised by knowledge distillation.
- The decoder is non-autoregressive with low redundancy, for that our encoder has modeled the overlap information between the selected sentences. In that case, our decoder only consists of two steps instead of the former three steps: ***predict*** the probability scores of those sentence vectors, and ***pick*** sentences simultaneously and individually when the predicted probability exceeds a threshold.
- The key component of our extractive model includes a weak supervision for the intermediate latent variables of iterative refinement. We design a teacher algorithm of knowledge distillation to produce high entropy soft labels at a high temperature, and progressively reduce the temperature along with iteration until a temperature of 1. Therefore, the former iterative steps with high temperature are equivalent to minimize the square difference between ground-truth and prediction, and the latter steps with a lower temperature will pay much more attention to matching the positive elements (Hinton, Vinyals, and Dean 2015).

Experimental results validate the effectiveness of ThresSum, which significantly outperforms BERTSUMEXT by 0.74 ROUGE-1 score on CNN/DM. The human evaluation also shows that our model is better in relevance compared with others. Our contributions in this work are concluded as follows:

1) Instead of extracting sentences one by one to form a top-k summary, we formulate a non-autoregressive decoder, which can extract a non-fixed number of summary sentences simultaneously and individually.

2) We propose iterative refinement to strengthen encoder and enhance the sentence representation. Simultaneously, we introduce and expand the knowledge distillation algorithm to progressively supervise the iterative refinement.

3) Our proposed framework has achieved superior performance compared with strong baselines. Moreover, we conduct an analysis to investigate where the performance gain of our model comes from.

Related Work

Extractive Summarization

There are two main lines of summarization: abstractive and extractive. The abstractive paradigm (Celikyilmaz et al. 2018; Sharma et al. 2019) focuses on generating a summary word-by-word after encoding the full document. The extractive approach (Cheng and Lapata 2016) directly selects sentences from the document to assemble into a summary. The abstractive approach is more flexible and generally produces less redundant summaries, while the extractive approach enjoys better factuality and efficiency (Cao et al. 2018).

Recent research work on extractive summarization spans a large range of approaches. These work usually instantiate their encoder-decoder architecture by choosing RNN (Nallapati, Zhai, and Zhou 2017; Zhou et al. 2018), Transformer (Wang et al. 2019; Zhong et al. 2019b; Liu and Lapata 2019; Zhang, Wei, and Zhou 2019) or GNN (Wang et al. 2020; Jia et al. 2020b) as encoder, autoregressive (Jadhav and Rajan 2018; Liu and Lapata 2019) or RL-based (Narayan, Cohen, and Lapata 2018; Arumae and Liu 2018; Bae et al. 2019) decoders. Despite the effectiveness, these models are with the top-k strategy essentially.

For two-stage summarization, Chen and Bansal (2018) and Bae et al. (2019) follow a hybrid extract-then-rewrite architecture, with policy-based RL to bridge the two networks together. Lebanoff et al. (2019), Xu and Durrett (2019) and Mendes et al. (2019) focus on the extract-then-compress learning paradigm, which will first train an extractor for content selection. Zhong et al. (2020) introduces extract-then-match framework, which employs BERTSUMEXT (Liu and Lapata 2019) as first-stage to prune unnecessary information. However, these two-stage approaches are inherently with error propagation and difficult to train.

Knowledge Distillation

The common formulation of knowledge distillation (KD) is proposed in (Buciluundefined, Caruana, and Niculescu-Mizil 2006; Hinton, Vinyals, and Dean 2015; Kim and Rush 2016), where a smaller student model is trained on soft probability labels provided by a larger teacher model (with temperature T for final softmax). More recently, (Tan et al. 2019) applied KD to multilingual NMT, and (Sun et al. 2019) proposed patient KD for BERT model compression. Our distillation focuses on using KD to generalize student model for fitting the cumbersome inter-relationship of sentences, while these previous work mostly focused on model compression.

Methodology

Problem Definition

Given a document D consisting of a sequence of M sentences $(s_1, s_2, ..., s_M)$ and a sentence s_i consisting of a sequence of N words $(w_{i1}, w_{i2}, ..., w_{iN})$. We denote by h_i and h_{ij} the embedding of sentences and words in a continuous space. The extractive summarizer aims to produce a summary $\mathcal{S}$ by selecting m sentences from D (where $m \leq M$). For each sentence $s_i \in D$, there is ground-truth $y_i \in \{0, 1\}$ and we will predict a label $\hat{y}_i \in \{0, 1\}$, where 1 means that s_i should be included in the summary. We assign a score $p(\hat{y}_i|s_i, D, \theta)$ to quantify s_i's relevance to the summary, where θ is the parameters of neural network model. Finally, we assemble a summary $\mathcal{S}$ by selecting m sentences, in which $p(1|s_i, D, \theta)$ exceeds the threshold.

Overview of Architecture

ThresSum consists of a powerful encoder and an easy-adjustable decoder, as shown in Figure 1(a).

Encoder: In order to learn the contextual representation of words, we utilize the pre-trained ALBERT (enhanced version of BERT) (Lan et al. 2020). The output of ALBERT contains

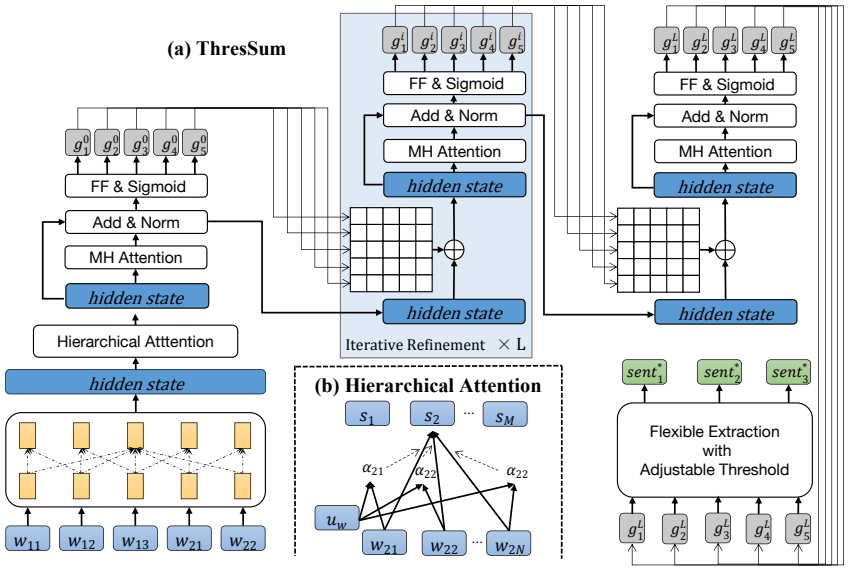


Figure 1: Overview of ThresSum.

words hidden state h_{ij}, special tokens $h_{[\texttt{CLS}]}$ and $h_{[\texttt{SEP}]}$. Liu and Lapata (2019) simply choose the $h_{[\texttt{CLS}]}$ as sentence representation, while we think it is difficult for [CLS] to identify the boundary of sentences. Inspired by Yang et al. (2016), we employ the hierarchical attention mechanism, shown as Figure 1(b), where the context vector u_w can be seen as a high level representation of a fixed query "what is the informative word".

In the process of iterative refinement with L steps, there are different state representations for s_i: $(\mathbf{h}_i^0, ...\mathbf{h}_i^L)$, where $\mathbf{h}_i^l$ is the hidden state of s_i at l-th iteration. We also introduce intermediate random variables $(g_i^0, ..., g_i^{L-1})$ for each sentence s_i, where g_i^l is the importance for sentence state $\mathbf{h}_i^l$. With the assistance of these latent variables, the encoder can implicitly aggregate the redundant information between selected summary sentences into $\mathbf{h}_i$.

Decoder: For the final sentences state $(\mathbf{h}_1^L, ..., \mathbf{h}_M^L)$, our decoder predicts the probabilities $(g_1^L, ..., g_M^L)$ with feed-forward network and sigmoid. Then we adjust the threshold to pick flexible summary sentences.

Approximate Models

The standard conditional probability distribution of selecting sentence s_i from D is as below:

$$P(\hat{y}_i = 1|D, y_1, ..., y_{i-1}, y_{i+1}, ..., y_M) \tag{1}$$

where we must have the ground-truth labels of other sentences before calculating s_i. However, there is still no known polynomial algorithm to solve it exactly.

Autoregressive originates from the literature on time-series models, where observations from the previous time-steps are used to predict the value at the current time step, i.e.,

$$P(\hat{y}_t = 1|D, y_{<t}) \tag{2}$$

The autoregressive paradigm is with error propagation inherently, especially when there is misjudgment for the first element. In this paper, we introduce a non-autoregressive architecture,

$$P(\hat{y}_i = 1|D, \hat{y}'_1, ..., \hat{y}'_{i-1}, \hat{y}'_{i+1}, ..., \hat{y}'_M) \tag{3}$$

where $\hat{y}'_i$ is the pre-predicted label introduced to implicitly capture the bidirectional dependencies among target symbols.

Iterative Refinement

Our model iteratively updates the latent variables g_i^l by masking document semantic information with $(g_1^{l-1}, ..., g_M^{l-1})$, especially:

$$\begin{aligned} g_i^l &= P(\hat{y}_i^l = 1|D, \hat{y}_1^{l-1}, ..., \hat{y}_M^{l-1}) \\ &= \sigma(\text{FFN}(\text{LN}(\mathbf{H}^l + \text{MHAtt}(\mathbf{H}^l))))_i \end{aligned} \tag{4}$$

where FFN, LN, MHAtt are for feed-forward network, layer normalize, and multi-head attention; $\hat{y}_i^{l-1}$ is the pre-predicted label for $l-1$ iteration; $\mathbf{H}^l$ is a matrix which contains the sentences hidden state $(\mathbf{h}_1^l, ..., \mathbf{h}_M^l)$ at l-th iteration.

As shown in Figure 1(a), for l-th iterative refinement, a Transformer-like unit layer is stacked on the top of hidden state:

$$\begin{aligned} \tilde{\mathbf{H}}^l &= \text{LN}(\mathbf{H}^{l-1} + \text{MHAtt}(\mathbf{H}^{l-1})) \\ \mathbf{H}^l &= \mathbf{W}_c\tilde{\mathbf{H}}^l - \tilde{\mathbf{H}}^l \odot \mathbf{W}_r \tanh(\mathbf{R}^l) \end{aligned} \tag{5}$$

where $\tilde{\mathbf{H}}^l$ is the input of l-th iteration, and $\mathbf{H}^l$ gets updated by reducing redundancy $\mathbf{R}^l$, a dynamic matrix representation of redundancy for each sentence $(\mathbf{r}_1^l, ..., \mathbf{r}_M^l)$; $\odot$ means the dot product of the i-th $\tilde{\mathbf{h}}_i^l$ in $\tilde{\mathbf{H}}^l$ and the corresponding $\mathbf{r}_i^l$, i.e. $\tilde{\mathbf{h}}_i^l \mathbf{W}_r tanh(\mathbf{r}_i^l)$; $\odot$ operation returns a new M-dimensional

vector. $\mathbf{r}_i^l$ represents the redundant phrase information of i-th sentence, which is a weighted summation of all other sentence-level hidden states:

$$\mathbf{r}_i^l = \sum_{j \in \{1,...,M\} \setminus \{i\}} g_j^{l-1} \tilde{\mathbf{h}}_j^l \tag{6}$$

where $g_j^{l-1} \in [0,1]$ is a predicted probability to mask the sentence information $\tilde{\mathbf{h}}_j^l$.

Finally, ThresSum is trained to predict the label of sentences and the overall training is equivalent to optimizing the following conditional probability:

$$\mathcal{L} = -\mathbb{E}_{l \sim \{0,...,L\}} \left[\frac{L+l}{2L} \mathbb{E}_{i \sim \{1,...,M\}} \phi(\hat{y}_i^l = y_i^l | s_i, D, \theta) \right] \tag{7}$$

where we gradually increase the proportion $\frac{L+l}{2L}$ of each refinement according to importance; l and i are the index of randomly sampled iteration step and sentence; ϕ is the regular binary cross-entropy loss with respect to the prediction $\hat{y}_i^l$ against soft ground-truth label y_i^l:

$$\phi(\hat{y}_i^l = y_i^l | s_i, D, \theta) = y_i^l \log(g_i^l) + (1 - y_i^l) \log(1 - g_i^l) \tag{8}$$

Knowledge Distillation for Soft Labels

Most summarization datasets only contain human written abstractive summaries as ground truth. Thus, a greedy approach (Nallapati, Zhai, and Zhou 2017; Liu and Lapata 2019) is employed that adds one sentence at a time incrementally to the summary, with maximizing the ROUGE score and stopping until none of remaining candidate sentences improves the score.

Theoretically, soft labels with high entropy will provide much more information than binary hard labels and much less variance in the gradient between training cases (Hinton, Vinyals, and Dean 2015). While, the binary labels can maximize the margin between positive and negative examples by extracting salient sentences and reduce redundancy.

Intuitively, the former iterative steps in our architecture serve as a small model with few iteration, and the latter steps are larger with more iterations. Therefore, the iterative refinement with g_i^l should be gradually trained with soft labels y_i^l, until the last step g_i^L with ground-truth binary label $\{0, 1\}$.

In this work, we modify the knowledge distillation (Jia et al. 2020a) to design more soft target labels $(y_i^0, ..., y_i^L)$ for the intermediate variables $(g_i^0, ..., g_i^L)$. As Algorithm 1, we design a teacher algorithm of knowledge distillation to produce high entropy soft labels at a high temperature, and progressively reduce the temperature along with the iterations until a temperature of 1. As a result, the former iterative steps with high temperature are equivalent to the regression approach and the latter steps with lower temperature will pay much more attention to matching positive units (Hinton, Vinyals, and Dean 2015).

We denote $r_i, r_2, ..., r_M$ as the individual ROUGE scores of each sentence against the human-written summary, especially:

Algorithm 1: Teacher Algorithm for Soft Labels

```
Initialize Sentence Set D = {s_1, ..., s_M} ;
Initialize ROUGE r_1, ...r_M, and Iteration Steps L ;
Sort D by r_1, ..., r_M in descending order ;
for l from 0 to L − 1 do
    Set the Temperature T as L − l ;
    for t from T to 1 do
        Temporary Sentence Set: D_temp ← {} ;
        Temporary ROUGE of D_temp: R_temp ← 0 ;
        for s_i from D[0] to D[end] do
            D_temp ← D_temp + s_i ;
            if R_temp is increasing then
                D ← D − s_i
            else
                D_temp ← D_temp − s_i ;
            end
        end
        Set the Sentence s in D_temp with Soft Label t/T ;
    end
    Set the Sentence s Remained in D with Label 0 ;
    Record these Soft Labels as (y_1^l, y_2^l, ..., y_M^l) ;
    Re-Initialize Sentence Set D = {s_1, ..., s_M} ;
    Re-Sort D by r_1, ..., r_M in descending order ;
end
```

Experiments

Datasets

As shown in Table 1, we employ two datasets widely-used with multiple sentences summary: CNN and Dailymail (CNN/DM) (Hermann et al. 2015) and New York Times (NYT) (Sandhaus 2008).

CNN/DM. We used the standard split (Hermann et al. 2015) for training, validation and test (90,266/1,220/1,093 for CNN and 196,96/12.148/10,397 for Daily Mail), with splitting sentences by Stanford CoreNLP (Manning et al. 2014) toolkit and pre-processing the dataset following (See, Liu, and Manning 2017) and (Xu et al. 2020). This dataset contains news articles and several associated abstractive highlights. We use the un-anonymized version as in previous summarization work and each document is truncated to 768 BPE tokens.

NYT. Following previous work (Zhang, Wei, and Zhou 2019; Xu and Durrett 2019), we use 137,778, 17,222 and 17,223 samples for training, validation and test, respectively. Input documents were truncated to 768 BPE tokens too. Note that there are different divisions for NYT (Durrett, Berg-Kirkpatrick, and Klein 2016; Liu and Lapata 2019) and several models are not evaluated on NYT officially. e.g. See, Liu, and Manning (2017) and Mendes et al. (2019), so we re-train and evaluate them on NYT with the source code from Github.

Parameters & Metrics

Our code is based on Pytorch (Paszke et al. 2019) and the pre-trained model employed in ThresSum is 'albert-xxlarge-

Datasets	# docs (train / val / test)	avg.doc length		avg.summary length	
		words	sentences	words	sentences
CNN	90,266 / 1,220 / 1,093	760.50	33.98	45.70	3.59
DailyMail	196,961 / 12,148 / 10,397	653.33	29.33	54.65	3.86
NYT	137,778 / 17,222 / 17,223	800.04	35.55	45.54	2.44

Table 1: Data Statistics: CNN/Daily Mail and NYT.

v2' (huggingface/transformers[1]). We train ThresSum (with about 400M parameters) two days for 100,000 steps on 2GPUs(Nvidia Tesla V100, 32GB) with gradient accumulation every two steps. Adam with $\beta_1 = 0.9, \beta_2 = 0.999$ is used as optimizer. Learning rate schedule follows the strategy with warming-up on first 10,000 steps.

We have tried the iteration steps of $[1, 3, 5, 7]$ for knowledge distillation, and $L = 5$ is the best choice based on the validation set. In comparison, we have tried to replace binary cross-entropy with regression objective, but the result indicates that regression can't achieve the performance of cross-entropy. The final threshold of extraction is 0.73 for CNN/DM and 0.78 for NYT, which are tuned on the validation set to get the highest ROUGE-1 score. A higher threshold is for a more concise summary and the lower threshold will return more information.

We report the F1 ROUGE score of ThresSum by *ROUGE-1.5.5.pl* (Lin 2004), which calculates the overlap lexical units between extracted sentences and ground-truth. Our source code will be available on Github.[2]

Baselines

Abstractive Methods: ABS is the normal architecture with RNN-based encoder and decoder. **PGC** augments the standard Seq2Seq attentional model with pointer and coverage mechanisms. **TransformerABS** employs Transformer in text summarization. **T5**, **BART**, and **ProphetNet** are pretrained on large unlabeled data and perform excellent performance with Transformer architecture. **PEGASUS** proposes Transformer-based models with extracted gap-sentences for abstractive summarization.

Extractive Methods: Oracle Summary is the extracted summary according to the ground-truth labels. Specifically, the oracle summary is essential to reveal the upper bound performance of the extractive paradigm. **Lead-3** is a base method for extractive text summarization that chooses first three sentences as a summary. **SummaRuNNer** takes content, salience, novelty, and position of each sentence into consideration when deciding if a sentence should be included in the extractive summary. **Exconsumm** first extracts sentences from a document and then compresses them. **PNBERT** tries to employ the unsupervised transferable knowledge. **DiscoBERT** extracts sub-sentential discourse units as candidates for extractive selection on a finer granularity. **BERTSUMEXT** applies pre-trained BERT in text summarization and proposes a general framework for both extractive and abstractive models. **MATCHSUM** is a two stage method for extract-then-match, and the first-stage is BERTSUMEXT.

[1]https://github.com/huggingface/transformers

[2]https://github.com/coder352/ThresSum

Result & Analysis

ROUGE Score

The experiment results of ROUGE are shown in Table 2. These scores are in accordance with original papers and the missing ones (only for NYT) are calculated with source code on Github by ourselves. It is obvious that our ThresSum outperforms all the baseline models, demonstrating that our enhanced encoder can help to model the relationships across source sentences and selected sentences. Specifically, our model outperforms MATCHSUM by 0.18 ROUGE-1, 0.29 ROUGE-2 and 0.21 ROUGE-L on CNN/DM. For more in-depth performance analysis, we note that: 1) Pre-trained BERT-like model is so powerful for that it can capture bidirectional dependencies by applying deep architecture; 2) Flexibility of the summary length is essential, for the large margin between ThresSum/MATCHSUM and BERTSUMEXT.

Ablation Studies

We propose several strategies to improve the performance of extractive summarization, including knowledge distillation(vs. binary labels), pre-trained ALBERT(vs. BERT), and iterative refinement(vs. None). To investigate the influence of these factors, we conduct the experiments and list the results in Table 3. Significantly, 1) Iterative refinement is more important than ALBERT, for the reason that the redundant information in selected sentences are difficult for ALBERT to model; 2) Knowledge distillation mechanism enlarges the advantage of extractive method, with high entropy for the soft labels (Hinton, Vinyals, and Dean 2015).

Human Evaluation for Summarization

It is not enough to only rely on the ROUGE evaluation for a summarization system, although the ROUGE correlates well with human judgments (Owczarzak et al. 2012). Therefore, we design an Amazon Mechanical Turk experiment based on ranking method. Following (Cheng and Lapata 2016), (Narayan, Cohen, and Lapata 2018) and (Zhang, Wei, and Zhou 2019), firstly, we randomly select 40 samples from CNN/DM test set. Then the human participants are presented with one original document and a list of corresponding summaries produced by different model systems. Participants are requested to rank these summaries (ties allowed) by taking informativeness (Can the summary capture the important information from the document) and fluency (Is the summary

Models	CNN/DM			NYT		
	R-1	R-2	R-L	R-1	R-2	R-L
Abstractive						
ABS (2015)	35.46	13.30	32.65	42.78	25.61	35.26
PGC (2017)	39.53	17.28	36.38	43.93	26.85	38.67
TransformerABS (2017)	40.21	17.76	37.09	45.36	27.34	39.53
$T5_{Large}$ (2020)	43.52	21.55	40.69	-	-	-
$BART_{Large}$ (2019b)	44.16	21.28	40.90	48.73	29.25	44.48
$PEGASUS_{Large}$ (2019b)	44.17	21.47	41.11	-	-	-
$ProphetNet_{Large}$ (2020)	44.20	21.17	41.30	-	-	-
Extractive						
Oracle (Sentence)	55.61	32.84	51.88	64.22	44.57	57.27
Lead-3 †	40.42	17.62	36.67	41.80	22.60	35.00
SummaRuNNer † ⋆ (2017)	39.60	16.20	35.30	42.37	23.89	38.74
Exconsumm ‡ ⋆ (2019)	41.7	18.6	37.8	43.18	24.43	38.92
$PNBERT_{Base}$ † ⋆(2019a)	42.69	19.60	38.85	-	-	-
$DiscoBERT_{Base}$ (2020)	43.77	20.85	40.67	-	-	-
$BERTSUMEXT_{Large}$ † ⋆(2019)	43.85	20.34	39.90	48.51	30.27	44.65
$MATCHSUM_{Base}$ ‡ ⋆(2020)	44.41	20.86	40.55	-	-	-
ThresSum$_{Large}$ ‡ •(Ours)	**44.59**	**21.15**	**40.76**	**50.08**	**31.77**	**45.21**

Table 2: Automatic Evaluation of ROUGE F1.

† means Top-K strategy; ‡ means Dynamically Adjusting Summary Length.
⋆ means with Binary Labels / Hard Labels; • means with Soft Labels.

Models	R-1	R-2	R-L
ThresSum	44.59	21.15	40.76
ThresSum w/o Distillation	44.18	20.95	40.42
ThresSum w/o ALBERT	44.35	21.03	40.57
ThresSum w/o Iteration	43.98	20.74	40.19

Table 3: Ablation Study on CNN/DM.

Models	1st	2nd	3rd	4th	MeanR
BERTSUMEXT	0.20	0.28	0.30	0.22	2.54
MATCHSUM	0.23	0.32	0.27	0.18	2.40
ThresSum	0.47	0.28	0.18	0.07	1.85
Ground-Truth	0.70	0.20	0.08	0.02	1.42

Table 4: Human Evaluation on CNN/DM.

grammatical) into account. Each document is annotated by three different participants separately.

The input article and ground truth summaries are also shown to the human participants in addition to the three model summaries (BERTSUMEXT, MATCHSUM and ThresSum). From the results shown in Table 4, it is obvious that ThresSum is better in relevance compared with others.

Trigram-Blocking vs. Iterative Refinement

Trigram blocking (Paulus, Xiong, and Socher 2018; Liu and Lapata 2019) skips the sentence that has trigram overlaps with the previously selected sentences, bringing a remarkable performance improvement on CNN/DM. Whereas there is another statistic on the test set of CNN/DM:

- 7.35% of the oracle summaries have trigram overlaps within its sentences.
- 8.64% of the summaries extracted by our ThresSum (with iterative refinement) have trigram overlaps within its sentences.
- 21.47% of the summaries extracted by our ThresSum (without iterative refinement) have trigram overlaps within its sentences.
- 0% of the summaries extracted by BERTSUMEXT (with Trigram-Blocking) have trigram overlaps within its sentences.

The oracle summaries are the upper bound of the extractive paradigm and 7.35% of them still contain trigram overlaps, while it is 0% for BERTSUMEXT with Trigram-Blocking. Consequently, Trigram-Blocking is a straightforward yet not optimal approach. In this paper, the proposed iterative refinement is to model the overlaps between the selected sentences, and it can effectively avoid but not empty the overlaps. ThresSum with Iterative Refinement reduces the overlaps from 21.47% to 8.64%, showing superiority over Trigram-Blocking.

Autoregressive Decoders with Threshold

ThresSum extracts a non-fixed number of summary sentences with a threshold, but whether there is a barrier that prevents

Models	R-1	R-L
ThresSum	44.59	40.76
ThresSum (with Trigram-Blocking)	44.03	40.25
BERTSUMEXT	43.85	39.90
BERTSUMEXT (with Threshold)	41.17	36.52
SummaRuNNer	39.60	35.30
SummaRuNNer (with Threshold)	36.58	33.61

Table 5: Threshold Strategy on CNN/DM.

previous models like SummaRuNNer or BERTSUMEXT to do this? It has been explained by Nallapati, Zhai, and Zhou (2017), that picking all sentences by comparing the predicted probability with a threshold may not be an optimal strategy since the training data is very imbalanced in terms of summary-membership of sentences.

Table 5 further summarizes the performance gain of threshold strategy. The thresholds for BERTSUMEXT and SummaRuNNer are tuned on the validation set individually to get highest ROUGE-1 score. It is obvious that the threshold strategy is not suitable for BERTSUMEXT/SummaRuNNer, for that the independent binary decision is based on the overlaps modeling between selected sentences. On the other hand, the Trigram-Blocking strategy with fixed top-3 summary sentences will damage the flexibility of ThresSum.

Non-Fixed Number of Summary Sentences

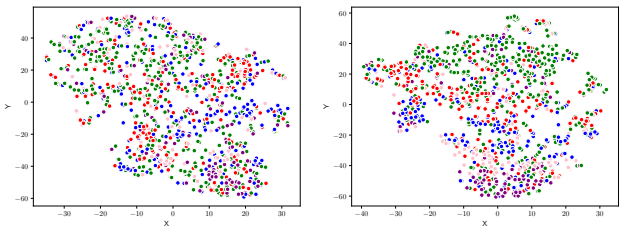
Since the extractive summarization requires sentence-level summary membership lables, Nallapati, Zhai, and Zhou (2017) first introduces a simple greedy approach to convert the abstractive summaries into extractive binary labels. Considering that ThresSum removes the restriction of the summary sentence number, it is necessary to discuss the distribution of summary sentence numbers on the test set of CNN/DM.

According to statistics, 5% / 27% / 68% of the test set examples are with 1- / 2- / 3-sentences summary. However, previous extractive approaches (such as SummaRuNNer and BERTSUMEXT) with top-k strategy only extract 3 sentences, which is not suitable for almost half of the examples.

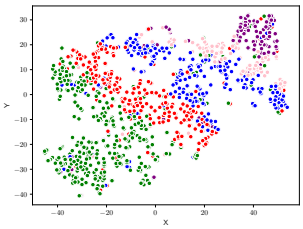
In this paper, there are about 6% / 35% / 59% for 1- / 2- / 3-sentences summary in CNN/DM test dataset, extracted by our ThresSum to get the highest ROUGE-1 score. Our threshold is still a hyper-parameter which need to be tuned. A higher threshold is for a more concise summary and the lower threshold will return more information.

Visualization

We visualize the three types of well-trained sentence representation of SummaRuNNer, BERTSUMEXT, and ThresSum by employing the t-SNE algorithm. T-SNE is used to visualize the representations of sentences learned by models, and a better extractive model should enlarge the distance between different clusters / different colors. Specifically, we randomly select 1000 sentences in test set and each sentence is represented as one point in the two-dimensional space.



(a) SummaRuNNer (b) BERTSUMEXT



(c) ThresSum

Figure 2: T-SNE Visualization on CNN/DM.

In Figure 2, there are five different colors, for five different soft labels in our ThresSum. It is obvious that: 1) Compared with BERTSUMEXT or SummaRuNNer, the sentence clusters in ThresSum are more distinguishable and meaningful; 2) The decoder of ThresSum is easy to score these sentences individually and extract summary sentences with an adjustable threshold; 3) That's why our ThresSum can extract summary sentences by threshold, while other models only extract top-3 sentences.

Conclusion

In this paper, to remove the restriction that the number of the summary sentences is fixed, we introduce three substantial improvements: strengthen the encoder by enhancing sentence representation through iterative refinement, simplify the decoder by removing the "sort" step, and weakly supervise the intermediate latent variables of iterative refinement by knowledge distillation. It is amazing that our ThresSum can extract each sentence separately only according to an adjustable threshold, which is a great improvement by fitting the distribution of sentence number in extractive summarization. Experimental results show that our method significantly outperforms previous models on the ROUGE score, the flexibility of summary sentence, and the proportion of the overlaps. Our future work will focus on extending the flexible summary-sentences mechanism to unsupervised summarization.

Acknowledgements

This research is supported by the National Key Research and Development Program of China (NO.2018YFB1004703) and National Natural Science Foundation of China (No.61902394). We thank all authors for their contributions and all anonymous reviewers for their constructive comments.

References

Aliguliyev, R. M. 2009. The two-stage unsupervised approach to multidocument summarization. *Automatic Control and Computer Sciences* 276–284.

Arumae, K.; and Liu, F. 2018. Reinforced Extractive Summarization with Question-Focused Rewards. In *ACL*, 105–111.

Bae, S.; Kim, T.; Kim, J.; and goo Lee, S. 2019. Summary Level Training of Sentence Rewriting for Abstractive Summarization. In *arXiv preprint arXiv:1909.08752*.

Buciluundefined, C.; Caruana, R.; and Niculescu-Mizil, A. 2006. Model Compression. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 535–541.

Cao, Z.; Wei, F.; Li, W.; and Li, S. 2018. Faithful to the Original: Fact Aware Neural Abstractive Summarization. In *AAAI*, 4784–4791.

Celikyilmaz, A.; Bosselut, A.; He, X.; and Choi, Y. 2018. Deep Communicating Agents for Abstractive Summarization. In *NAACL-HLT*, 1662–1675.

Chen, Y.-C.; and Bansal, M. 2018. Fast Abstractive Summarization with Reinforce-Selected Sentence Rewriting. In *ACL*, 675–686.

Cheng, J.; and Lapata, M. 2016. Neural summarization by extracting sentences and words. In *ACL*. doi:10.18653/v1/p16-1046.

Dong, Y.; Shen, Y.; Crawford, E.; van Hoof, H.; and Cheung, J. C. K. 2018. BanditSum: Extractive Summarization as a Contextual Bandit. *EMNLP* 3739–3748.

Durrett, G.; Berg-Kirkpatrick, T.; and Klein, D. 2016. Learning-based single-document summarization with compression and anaphoricity constraints. In *arXiv preprint arXiv:1603.08887*.

Galanis, D.; and Androutsopoulos, I. 2010. An extractive supervised two-stage method for sentence compression. In *NAACL-HLT*, 885–893.

Hermann, K. M.; Kocisky, T.; Grefenstette, E.; Espeholt, L.; Kay, W.; Suleyman, M.; and Blunsom, P. 2015. Teaching machines to read and comprehend. In *NIPS*, 1693–1701.

Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the Knowledge in a Neural Network. In *NIPS Deep Learning and Representation Learning Workshop*. URL http://arxiv.org/abs/1503.02531.

Jadhav, A.; and Rajan, V. 2018. Extractive summarization with swap-net: Sentences and words from alternating pointer networks. In *ACL*, 142–151.

Jia, R.; Cao, Y.; Shi, H.; Fang, F.; Liu, Y.; and Tan, J. 2020a. DistilSum: Distilling the Knowledge for Extractive Summarization. In *CIKM*, 2069–2072.

Jia, R.; Cao, Y.; Tang, H.; Fang, F.; Cao, C.; and Wang, S. 2020b. Neural Extractive Summarization with Hierarchical Attentive Heterogeneous Graph Network. In *EMNLP*, 3622–3631.

Kim, Y.; and Rush, A. M. 2016. Sequence-Level Knowledge Distillation. In *EMNLP*, 1317–1327.

Lan, Z.; Chen, M.; Goodman, S.; Gimpel, K.; Sharma, P.; and Soricut, R. 2020. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. In *ICLR*. OpenReview.net. URL https://openreview.net/forum?id=H1eA7AEtvS.

Lebanoff, L.; Song, K.; Dernoncourt, F.; Kim, D. S.; Kim, S.; Chang, W.; and Liu, F. 2019. Scoring Sentence Singletons and Pairs for Abstractive Summarization. *ACL* 2175–2189.

Lin, C.-Y. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, 74–81.

Liu, Y.; and Lapata, M. 2019. Text summarization with pretrained encoders. In *EMNLP*, 3728–3738.

Manning, C.; Surdeanu, M.; Bauer, J.; Finkel, J.; Bethard, S.; and McClosky, D. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *ACL*, 55–60.

Mendes, A.; Narayan, S.; Miranda, S.; Marinho, Z.; Martins, A. F.; and Cohen, S. B. 2019. Jointly Extracting and Compressing Documents with Summary State Representations. In *NAACL-HLT*, 3955–3966.

Nallapati, R.; Zhai, F.; and Zhou, B. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *AAAI*, 3075–3081.

Narayan, S.; Cohen, S. B.; and Lapata, M. 2018. Ranking sentences for extractive summarization with reinforcement learning. In *NAACL-HLT*, 1747–1759.

Owczarzak, K.; Conroy, J. M.; Dang, H. T.; and Nenkova, A. 2012. An assessment of the accuracy of automatic evaluation in summarization. In *Proceedings of Workshop on Evaluation Metrics and System Comparison for Automatic Summarization*, 1–9.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; Desmaison, A.; Kpf, A.; Yang, E.; DeVito, Z.; Raison, M.; Tejani, A.; Chilamkurthy, S.; Steiner, B.; Fang, L.; Bai, J.; and Chintala, S. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *NIPS*, 8024–8035.

Paulus, R.; Xiong, C.; and Socher, R. 2018. A Deep Reinforced Model for Abstractive Summarization. In *ICLR*.

Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; and Liu, P. J. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.* 140:1–140:67.

Rush, A. M.; Chopra, S.; and Weston, J. 2015. A Neural Attention Model for Abstractive Sentence Summarization. In *EMNLP*, 379–389.

Sandhaus, E. 2008. The new york times annotated corpus. In *Linguistic Data Consortium, Philadelphia*.

See, A.; Liu, P. J.; and Manning, C. D. 2017. Get To The Point: Summarization with Pointer-Generator Networks. In *ACL*, 1073–1083.

Sharma, E.; Huang, L.; Hu, Z.; and Wang, L. 2019. An Entity-Driven Framework for Abstractive Summarization. In *EMNLP*, 3278–3289.

Sun, S.; Cheng, Y.; Gan, Z.; and Liu, J. 2019. Patient Knowledge Distillation for BERT Model Compression. In *EMNLP*, 4322–4331.

Tan, X.; Ren, Y.; He, D.; Qin, T.; Zhao, Z.; and Liu, T.-Y. 2019. Multilingual Neural Machine Translation with Knowledge Distillation. In *ICLR*. OpenReview.net. URL https://openreview.net/forum?id=S1gUsoR9YX.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *NIPS*, 5998–6008.

Wang, D.; Liu, P.; Zheng, Y.; Qiu, X.; and Huang, X. 2020. Heterogeneous Graph Neural Networks for Extractive Document Summarization. In *ACL*, 6209–6219.

Wang, D.; Liu, P.; Zhong, M.; Fu, J.; Qiu, X.; and Huang, X. 2019. Exploring Domain Shift in Extractive Text Summarization. In *arXiv preprint arXiv:1908.11664*.

Xiao, W.; and Carenini, G. 2019. Extractive Summarization of Long Documents by Combining Global and Local Context. In *EMNLP*, 3009–3019.

Xu, J.; and Durrett, G. 2019. Neural Extractive Text Summarization with Syntactic Compression. *EMNLP* 3290–3301.

Xu, J.; Gan, Z.; Cheng, Y.; and Liu, J. 2020. Discourse-Aware Neural Extractive Text Summarization. In *ACL*, 5021–5031.

Yan, Y.; Qi, W.; Gong, Y.; Liu, D.; Duan, N.; Chen, J.; Zhang, R.; and Zhou, M. 2020. ProphetNet: Predicting Future N-gram for Sequence-to-Sequence Pre-training. In *arXiv preprint arXiv:2001.04063*, 2401–2410.

Yang, Z.; Yang, D.; Dyer, C.; He, X.; Smola, A.; and Hovy, E. 2016. Hierarchical Attention Networks for Document Classification. In *NAACL-HLT*, 1480–1489.

Zhang, H.; Gong, Y.; Yan, Y.; Duan, N.; Xu, J.; Wang, J.; Gong, M.; and Zhou, M. 2019a. Pretraining-Based Natural Language Generation for Text Summarization. In *CoNLL*, 789–797.

Zhang, J.; Zhao, Y.; Saleh, M.; and Liu, P. J. 2019b. PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization. In *arXiv preprint arXiv:1912.08777*, 11328–11339.

Zhang, X.; Lapata, M.; Wei, F.; and Zhou, M. 2018. Neural Latent Extractive Document Summarization. In *EMNLP*, 779–784.

Zhang, X.; Wei, F.; and Zhou, M. 2019. HIBERT: Document Level Pre-training of Hierarchical Bidirectional Transformers for Document Summarization. In *ACL*, 5059–5069.

Zhong, M.; Liu, P.; Chen, Y.; Wang, D.; Qiu, X.; and Huang, X. 2020. Extractive Summarization as Text Matching. In *ACL*, 6197–6208.

Zhong, M.; Liu, P.; Wang, D.; Qiu, X.; and Huang, X. 2019a. Searching for Effective Neural Extractive Summarization: What Works and Whats Next. In *ACL*, 1049–1058.

Zhong, M.; Wang, D.; Liu, P.; Qiu, X.; and Huang, X. 2019b. A Closer Look at Data Bias in Neural Extractive Summarization Models. In *arXiv preprint arXiv:1909.13705*.

Zhou, Q.; Yang, N.; Wei, F.; Huang, S.; Zhou, M.; and Zhao, T. 2018. Neural document summarization by jointly learning to score and select sentences. In *ACL*, 654–663.