A Set Prediction Network For Extractive Summarization

Xiaoxia Cheng, Yongliang Shen, Weiming Lu[†]
College of Computer Science and Technology, Zhejiang University
{zjucxx, syl, luwm}@zju.edu.cn

Abstract

Extractive summarization focuses on extracting salient sentences from the source document and incorporating them in the summary without changing their wording or structure. The naive approach for extractive summarization is sentence classification, which makes independent binary decisions for each sentence, resulting in the model cannot detect the dependencies between sentences in the summary. Recent approaches introduce an autoregressive decoder to detect redundancy relationship between sentences by step-by-step sentence selection, but bring train-inference gap. To address these issues, we formulate extractive summarization as a salient sentence set recognition task. To solve the sentence set recognition task, we propose a set prediction network (**SetSum**), which sets up a fixed set of learnable queries to extract the entire sentence set of the summary, while capturing the dependencies between them. Different from previous methods with an auto-regressive decoder, we employ a non-autoregressive decoder to predict the sentences within the summary in parallel during both the training and inference process, which eliminates the train-inference gap. Experimental results on both single-document and multi-document extracted summary datasets show that our approach outperforms previous state-of-the-art models.

1 Introduction

Extractive summarization is the process of extracting a brief set of sentences from the source document to cover the salient information of it. Compared with abstractive summarization (Liu and Liu, 2021; Wu et al., 2021), extractive summarization is less likely to deviate from the source document, as well as more efficient in execution. Due to these advantages, it has become widely utilized for automatic summarization tasks.

Recently, most of the approaches (Nallapati et al., 2017; Liu and Lapata, 2019; Xu et al., 2020)

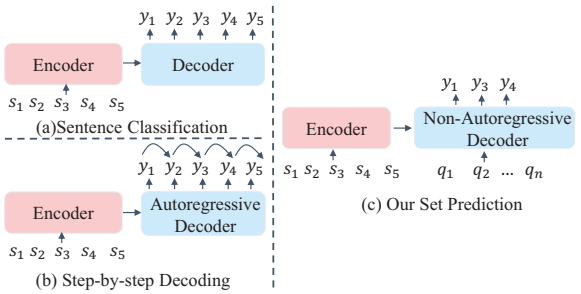

Figure 1: The Encoder-Decoder Framework for Extractive Summarization. (a) For a document, the methods based on sentence classification make decisions for each sentence individually, resulting in the inability to detect the dependencies between sentences in the summary. (b) The methods with an autoregressive decoder, which select sentences step by step, can capture some of the dependencies but lead to inefficiency and a training-inference gap. (c) In contrast, our set prediction method can get the sentences within the summary in parallel while capturing the dependencies between them and eliminating the training-inference gap.

formulate extractive summarization as a sequence labeling task, following the encoder-decoder framework. Sentence classification (Cheng and Lapata, 2016) is a naive solution for the task, which makes independent binary decisions for each sentence in the source document, leading to high redundancy of the summary, as shown in Figure 1 (a). To tackle the redundancy problem, Zhou et al. (2018); Narayan et al. (2020) introduce an autoregressive decoder, which select sentences step by step to construct the summary, until the length limit of the summary is reached, as shown in Figure 1 (b). Trigram Blocking (Paulus et al., 2018), as a plug-and-play method during inference, has the same motivation, which allows the model to consider the sentences already selected when making binary decisions for the current sentence. Besides, some methods (Narayan et al., 2018b; Bae et al., 2019; Gu et al., 2022) incorporate reinforcement learning to optimize the final evaluation metric with a

[†] Corresponding author.

Findings of the Association for Computational Linguistics: ACL 2023, pages 4766–4777
July 9-14, 2023 ©2023 Association for Computational Linguistics

step-by-step selection strategy. Zhong et al. (2020); Chen et al. (2021) propose a method to construct a summary with two steps, where step one is for constructing candidates summary, step two is for selecting a summary from the candidates. However, these methods not only decrease the efficiency of training and inference but also suffer from the training-inference gap.

To address the above issues, we formulate extractive summarization as a salient sentence set recognition task, which treats summary as the salient set of sentences, rather than a set of salient sentences. To solve the sentence set recognition task, we propose a set prediction network based on the encoder-decoder framework, which sets up a fixed set of learnable queries to extract the entire sentence set of the summary, while capturing the dependencies between them. As shown in Figure 1 (c), different from previous approaches, we employ a non-autoregressive decoder to extract the sentence set of the summary in parallel during both training and inference. The non-autoregressive decoder receives the sentence representation and a set of learnable vectors called sentence queries to decode the final sentence set in the summary. To measure the difference between predictions and gold labels in an end-to-end manner, we employ a loss function based on bipartite matching, which can produce an optimal matching between predictions and gold labels with minimal assignment cost. Compared with the autoregressive approach, our set prediction network is efficient and can eliminate the gap between training and inference. Furthermore, the decoder is able to capture the dependencies between sentences through a self-attention mechanism between sentence queries and then make joint decisions on the entire sentence set. Experimental results on four single-document and one multi-document extracted summary datasets show that our approach outperforms previous state-of-the-art models.

Our main contributions are as follows:

- We formulate extractive summarization as a salient sentence set recognition task, which treats summary as a salient set of sentences, rather than a set of salient sentences.
- To solve the sentence set recognition task, we propose a set prediction network, which not only enables capturing the dependencies between sentences in the summary but also eliminates the training-inference gap.
- Experiments show that our proposed model achieves state-of-the-art performance on several single-document and multi-document datasets.

2 Related Work

Traditional approaches (Nallapati et al., 2017; Liu and Lapata, 2019; Xu et al., 2020) formulate extractive summarization as a sequence labeling task following the encoder-decoder framework. To improve the performance of extractive summarization, some methods introduce autoregressive decoder (Liu and Lapata, 2019; Narayan et al., 2020) or reinforcement learning (Dong et al., 2018; Gu et al., 2022). These methods construct summary step-by-step until the length limit of the summary is reached. In addition to the above paradigm of sequence labeling, Zhong et al. (2020); An et al. (2022) formulate the extractive summarization task as a semantic text matching problem. Tang et al. (2022) formulates extractive summarization as an Optimal Transport (OT) problem from document to summary. Jia et al. (2021) selects sentences simultaneously from the source document when the predicted sentence probabilities exceed a threshold. Although the above methods have made different degrees of progress in extractive summarization, they also encounter various challenges, such as insufficient decoding efficiency, training, inference gap, unstable results, etc.

Recently, the query-based approach is employed in the summarization task. For example, (Xu and Lapata, 2022) unifies that all summaries are a response to a query. (Zhang et al., 2022) use learnable queries as a control signal to control summary generation. These query-based methods focus on generating a highly relevant summary for a given query, in which queries can be observable or latent. Besides, these methods restrict queries to a single sample and semantic space. In this paper, we propose a set prediction network for extractive summarization based on sample-independent queries, which uses a non-autoregressive decoder to improve the decoding efficiency and unify the training and inference processes.

3 Method

In this section, we first introduce the task formulation in §3.1 and then describe each component of our method in detail. As shown in Figure 2, Our method consists of three components, a doc-

ument encoder §3.2, a non-autoregressive set decoder §3.3, and a bipartite matching loss §3.4.

3.1 Task Formulation

Given a training sample $(\mathcal{D}, \mathcal{G})$, where $\mathcal{D} = (s_1, s_2, \ldots, s_n)$ denotes the original document with n sentences, $\mathcal{G}$ denotes reference summary. Our goal is to select $S^* = \{s_1^*, s_2^*, \ldots, s_m^*\}$ from $\mathcal{D}$ to cover the sailent information of it, where $m \leq n$. A set of gold sentences $Y = \{< y_i^l, y_i^r, y_i^t >\}_{i=0}^{m-1}$ is derived by a greedy selection strategy, where $y_i^l, y_i^r \in [0, n-1]$, $y_i^t \in \{0, 1\}$ represents the left boundary, right boundary, and label of the i-th sentence, respectively.

3.2 Document Encoder

The encoder is designed to get a contextual representation of the sentence in the document. Following Liu and Lapata (2019), we first concatenate the sentences together while inserting a [CLS] and a [SEP] token at the start and the end of each sentence, respectively, and then input them to BERT (Devlin et al., 2019) to obtain a contextual representation of each token. After BERT encoding, we further encode the document with a bidirectional LSTM layer. Then, we take the representation of all the [CLS] tokens as the representation of the sentences $h = \{h_1, h_2, \cdots, h_n\}$.

In order to improve the model's ability to capture the inter-sentence relationships, we use a 3-layer Transformer (Vaswani et al., 2017) to encode it, which can be formulated as:

$$h^s = \text{Transformer}(h) \tag{1}$$

Finally, we get contextual representation for sentences $h^s = \{h_1^s, h_2^s, \cdots, h_n^s\}$ in the source document.

3.3 Set Decoder

The purpose of the set decoder is to generate the entire sentence set of the summary in parallel based on the output of the document encoder. To achieve this purpose, we use a non-autoregressive decoder as the backbone of the set decoder.

Input The input of the set decoder consists of M learnable randomly initialized vectors $e^q \in \mathbb{R}^{M \times d}$ and the output $h^s \in \mathbb{R}^{n \times d}$ of the document encoder. Each query corresponds to a prediction, and for M queries the set decoder generates M predictions. In order to decode all sentences in the summary, we set M greater than the maximum number of sentences contained in the summary.

Non-Autoregressive Decoder The set decoder is based on a non-autoregressive decoder. We use a N-stacked identical layer to construct the non-autoregressive decoder. Each layer incorporates a multi-headed self-attention mechanism to represent the relationship between queries e^q, and a multi-headed cross-attention mechanism to fuse information of the sentence h^s in the source document, which can be formulated as follows:

$$H^q = \text{Decoder}(e^q; h^s) \tag{2}$$

where e^q, h^s denote initialized query vectors and sentence representation in the source document, respectively.

Through the non-autoregressive decoder, M sentence queries are transformed into M query embeddings, which are denoted as $H^q \in \mathbb{R}^{M \times d}$. In contrast to the autoregressive decoder that needs to adopt the mask mechanism to prevent information leakage, the non-autoregressive decoder has no need to adopt the mask strategy to prevent the earlier decoding steps from obtaining information from the subsequent steps. Therefore, we do not add any causal mask in the multi-head self-attention mechanism.

Set Prediction Each query embedding h_i^q in H^q predicts one sentence from document total M in parallel. Set prediction is a joint decision of boundary and label.

To get the boundary for each query h_i^q, we first interact the query with each sentence of the document by two linear layers. The fusion representation of the i-th query and j-th sentence is computed as:

$$h_{i,j}^{f^{r/l}} = \text{Tanh}(h_i^q w_i + h_j^s w_j) \tag{3}$$

where $w_i, w_j \in \mathbb{R}^{d \times d}$ are trainable projection parameters, r/l denotes left or right. Then we get fuse representation of the i-th query with all sentence $h_i^{f^{r/l}} = [h_{i0}^{f^{r/l}}, h_{i1}^{f^{r/l}}, \cdots, h_{in}^{f^{r/l}}]$.

According to the fuse representation, we calculate the distribution of the left or right boundary:

$$p_i^{r/l} = \text{Softmax}(h_i^{f^{r/l}}) \tag{4}$$

Furthermore, we can get the label probability of the query by the i-th query belonging to label c:

$$p_i^c = \frac{exp(h_i^q w_i^c + b_i^c)}{\sum_{c' \in C} exp(h_i^q w_i^{c'} + b_i^{c'})} \tag{5}$$

where $w_i^{c'}$ and $b_i^{c'}$ are learnable parameter.

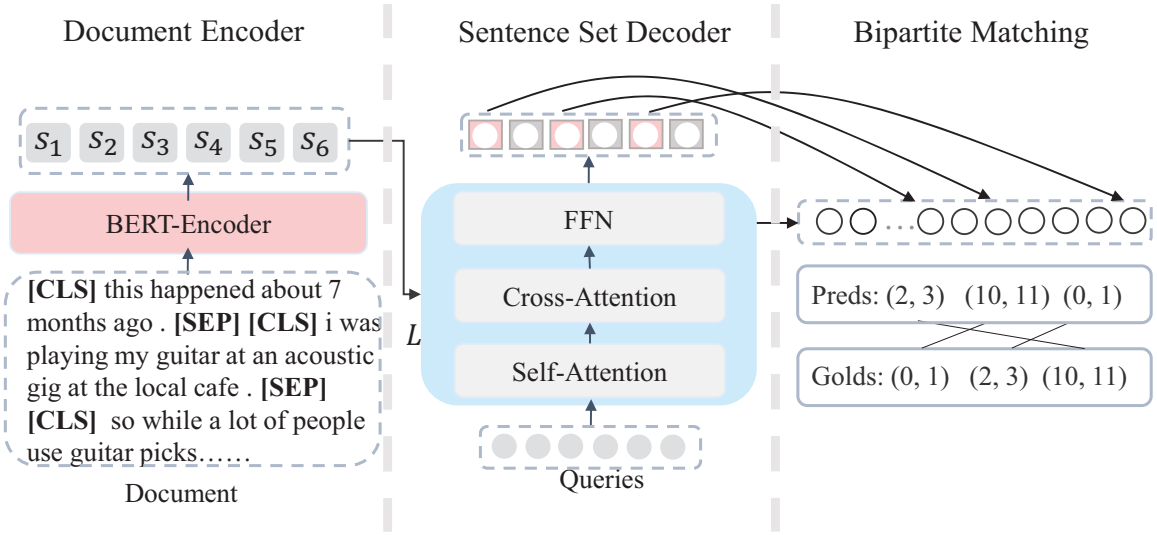

Figure 2: The architecture of our set prediction network. The document encoder get sentence representation from the document, and the sentence set decoder predicts the final sentence set of the summary with a non-autoregressive decoder. Then our method trains through a loss function based on bipartite matching, which can produce an optimal matching between gold labels and predicted results with minimal assignment cost.

Finally, the i-th query predicts result is $(\tau_i^l, \tau_i^r, \tau_i^c)$. $\tau_i^l = \text{argmax}(p_i^l)$ and $\tau_i^r = \text{argmax}(p_i^r)$ are the left and right boundary, $\tau_i^c = \text{argmax}_c(p_i^c)$ is the sentence label. Note that a special predicate label ϕ is included to indicate no sentence.

3.4 Bipartite Matching Loss

The main challenge of training is measuring the difference between the M decoding results $\hat{Y}$ and gold sentence set Y in an end-to-end manner. We introduce a bipartite matching loss to overcome this challenge. The calculation of the loss can be broken down into two stages: finding the optimal matching and then calculating the loss based on the optimal matching.

Finding the Optimal Matching. We find the optimal matching between gold set Y and the model output $\hat{Y}$ by minimizing the cost between them. Notably, a query only can assign one instance in gold set, and vice versa. Since the model predicts results size M larger than the gold sentence set size, we first pad Y to the size M with ϕ. Then the cost of assigning $\hat{Y}_i$ with Y_j is defined as:

$$C_{match}(\hat{Y}_i, Y_j) = -\mathbb{1}_{\{c_i \neq \phi\}}[p_j^r(r_i) + p_j^l(l_i) + p_j^c(c_i)] \quad (6)$$

Finally, we get the optimal permutation element of o^* with the lowest cost, which is defined as:

$$o^* = \underset{o \in \mathcal{O}_M}{\text{argmin}} \sum_i^M C_{match}(\hat{Y}_{o(i)}, Y_j) \quad (7)$$

where $\mathcal{O}_M$ is the space of all M-length permutations and $\mathcal{O}_M$ increases as M increases, resulting in computational efficiency challenges. To obtain the optimal assignment o^* efficiently, we use the Hungarian algorithm (Kuhn, 1955). With this algorithm, the optimal matching o^* can be easily computed in polynomial time $(O(M^3))$.

Calculating the Loss. After obtaining the optimal matching o^*, we then calculate the loss for all matched pairs in o^*. We define the loss as:

$$\mathcal{L}(\hat{Y}, Y) = \sum_i^M \{-logp_{o^*(i)}^c(c_i) + \mathbb{1}_{\{c_i \neq \phi\}}[-logp_{o^*(i)}^l(l_i) - logp_{o^*(i)}^r(r_i)]\} \quad (8)$$

4 Experiment

4.1 Datasets and Evaluation Metrics

To demonstrate the effectiveness of our model, we conduct experiments on five single-document datasets and a multi-document dataset: **CNN/DailyMail** (Hermann et al., 2015) is a widely used single-document news summarization dataset

Datasets	Source	Type	#Pairs			#Tokens (avg)		#Ext
			Train	Valid	Test	Doc.	Sum.	
CNN/DM	News	SDS	287,084	13,367	11,489	766.1	58.2	3
XSum	News	SDS	203,028	11,273	11,332	430.2	23.3	2
Reddit	Social Media	SDS	41,675	645	645	482.2	28.0	2
WikiHow	Knowledge Base	SDS	168,126	6,000	6,000	580.8	62.6	4
PubMed	Scientific Paper	SDS	83,233	4,646	5,025	444.0	209.5	7
Multi-News	News	MDS	44,972	5,622	5,622	487.3	262.0	9

Table 1: Details statics information of datasets we used in the experiment. SDS and MDS represent single-document and multi-document summarization respectively. #EXT denotes the number of sentences that should extract from datasets.

containing article-highlight pairs. **XSum** (Narayan et al., 2018a) concludes one-sentence summaries of online articles from BBC. **Reddit** (Kim et al., 2019) is collected from social media platforms with weak lead bias and strong abstractive features. **WikiHow** (Koupaee and Wang, 2018) is a dataset extracted and constructed from an online knowledge base covering a wide range of topics and with high diversity styles. **PubMed** (Cohan et al., 2018) is a long-form dataset of scientific papers, and we use the truncated version like (Gu et al., 2022). **Multi-News** (Fabbri et al., 2019) is a multi-document news summarization dataset. More statistical information about the datasets we used in the experiment is shown in Table 1.

We evaluate the quality of generated summaries using the popular automatic evaluation method ROUGE (Lin, 2004). In ROUGE, unigram and bigram overlap (ROUGE-1, 2) is used to measure informativeness and the longest common subsequence (ROUGE-L) is used to measure fluency. For simplicity, ROUGE-1, ROUGE-2, and ROUGE-L are abbreviated as R-1, R-2, and R-L, respectively. In addition, we also apply human evaluation, as a complement to the automatic evaluation.

4.2 Baselines

Basic Extractive Methods: **LEAD** selects the first several sentences as a summary from the source document. **ORACLE** extracts sentences as a summary from the source document according to the gold labels. **BERTEXT** (Liu and Lapata, 2019) utilizes pre-trained BERT (Devlin et al., 2019) to get the sentence representation and assign a label to a sentence to decide whether a sentence is in the summary.

Auoregressive selection Methods: **BERTEXT + RL** (Bae et al., 2019) directly maximizes summary-level ROUGE scores through reinforcement learning based on BERTEXT. **BERTEXT + Tri-Blocking** (Liu and Lapata, 2019) introduces trigram blocking during inference based on BERTEXT. **DiscoBERT** (Xu et al., 2020) focuses on capturing a more semantically rich representation of sentences based on the RST graph to improve the quality of the summary. **Stepwise ETCSum** (Narayan et al., 2020) enable stepwise summarization by injecting the previously planned summary content into the structured transformer.

Parallel Prediction Methods: **MatchSum** (Zhong et al., 2020) is a summary-level approach, which selects the best one from the candidate summaries to form the final summary. **OTExtSum** (Tang et al., 2022) is a non-learning-based approach, which formulates text summarization as an Optimal Transport (OT) problem from document to summary. **ThresSum**$_{large}$ (Jia et al., 2021) picks up sentences simultaneously by a non-autoregressive decoder when predicted sentence probabilities exceed a threshold.

4.3 Implementation Details

We use pre-trained BERT (Devlin et al., 2019) in the encoder. In order to make a fair comparison with other methods, we use the BERT-base version on all datasets in the experiments. The number of stacked transformer blocks in the encoder is set to 2-4, and the batch size is set to 12. The number of stacked transformer blocks in the non-autoregressive decoder is set to 6. We initialize all queries using the normal distribution $\mathcal{N}(0.0, 0.02)$. We apply a linear warmup-decay learning rate scheduler. All experiments are conducted on an NVIDIA GeForce RTX 3090.

Models	R-1	R-2	R-L
LEAD	40.43	17.62	36.67
ORACLE	52.59	31.23	48.87
BERTEXT	42.57	19.96	39.04
Auoregressive Selection Methods			
BERTEXT + RL	42.76	19.87	39.11
BERTEXT + TriBlk	43.23	20.22	39.60
DiscoBERT	43.77	20.85	40.67
ETCSum	43.80	20.80	39.77
Parallel Prediction Methods			
MatchSum	44.22	20.62	40.38
OTExtSum	34.50	12.80	27.80
ThresSum$^{*}_{large}$	44.59	21.15	40.76
SetSum	**44.62**	**20.81**	**40.76**

Table 2: Results on CNN/DailyMail dataset. TriBlk is an abbreviation for Trigram Blocking. ThresSum$^{*}_{large}$ builds on BERT large architectures (24 layers), whereas ours build on BERT base architectures (12 layers).

Models	XSum		
	R-1	R-2	R-L
LEAD	19.58	2.38	14.74
ORACLE	32.33	9.33	23.86
BERTEXT	22.86	4.48	17.16
MatchSum	24.48	4.58	18.31
SetSum	**24.80**	**4.59**	**18.52**
Models	**Reddit**		
	R-1	R-2	R-L
LEAD	17.01	2.72	13.76
ORACLE	35.76	13.44	28.45
BERTEXT	23.86	5.85	19.11
MatchSum	24.90	5.91	20.03
SetSum	**25.49**	**6.39**	**20.33**

Table 3: Results on XSum and Reddit datasets. BERTEXT here has been added to Trigram Blocking strategy.

5 Results and Analysis

5.1 Overall Performances

Results on CNN/DailyMail Table 2 shows the evaluation results of proposed methods and baselines on the CNN/DailyMail dataset. Compared with the best auto-regressive methods ETCSum, our method achieves +0.82, +0.01, and +1.01 improvements on R-1, R-2, and R-L, respectively. Compared with the best summary-level method MatchSum, our method has +0.40, +0.19, and +0.38 improvements on R-1, R-2, and R-L, respectively.

Besides, we can see that the introduction of RL-based autoregressive decoding slightly improves the quality of the summary, but is less effective than a simple method like Trigram Blocking. Furthermore, the summary-level methods are still remarkably effective extractive summarization methods at present, which consider the entire summary and can be seen as a special non-autoregressive method.

Results on Datasets with Short Summaries We conduct experiments on XSum and Reddit to prove the effectiveness of our method on short datasets. As shown in Table 3, we can see there is a pretty apparent improvement achieved on the XSum and Reddit datasets than baseline methods. In particular, on the Reddit dataset, our method achieves +0.59, +0.48, and +0.3 improvements on R-1, R-2, and R-L, respectively. We think this is due to the fact that the token length of the source documents in the two dataset are shorter than other datasets, which enable the model is easy to get contextual sentence representation.

Results on Datasets with Long Summaries Table 4 shows the results on WikiHow, Pubmed, and Multi-News, where Multi-News is a multi-document dataset. According to our intuition, the summary of these datasets has a larger set of sentences compared with other datasets, which requires more challenges to the model to find the correct set of sentences to construct the summary. From the results, we can see that our method has slight improvements on PubMed and Multi-News datasets, which is consistent with our intuition. For example, on the Multi-News dataset, compared with the summary-level method MatchSum our approach achieves +0.13, +0.29, and +0.11 on R-1, R-2, and R-L, respectively. On the WikiHow dataset, we achieve comparable results to MatchSum. We think there are two reasons for the result. First, WikiHow has longer summaries compared to Reddit dataset, even though they are both datasets with abstract characteristics. Second, unlike CNN/DailyMail where the main information is more concentrated, that in WikiHow is more scattered.

Methods	WikiHow			PubMed			Multi-News		
	R-1	R-2	R-L	R-1	R-2	R-L	R-1	R-2	R-L
LEAD	24.97	5.83	23.24	37.58	12.22	33.44	43.08	14.27	38.97
ORACLE	35.59	12.98	32.68	45.12	20.33	40.19	49.06	21.54	44.27
BERTEXT	30.31	8.71	28.24	41.05	14.88	36.57	45.80	16.42	41.53
BERTEXT+TriBlk	30.37	8.45	28.28	38.81	14.62	34.52	44.94	15.47	40.63
MatchSum	31.85	8.98	29.58	41.21	14.91	36.75	46.20	16.51	41.89
SetSum	31.66	8.72	29.36	**41.53**	**15.11**	**36.88**	**46.33**	**16.80**	**42.00**

Table 4: Results on WikiHow, PubMed, and Multi-News datasets.

5.2 Ablation Studies & Analysis

To demonstrate the effectiveness of the components in the model, we performed a series of ablation experiments on the Reddit dataset.

Effects of Model Architecture To verify the effectiveness of the model architecture of SetSum, we remove the parallel decoder and find a significant drop of model results. Specifically, the results shown in Table 5 dropped by 1.38, 0.64, and 1.01 at R-1, R-2, and R-L, respectively, which means that the encoder-based sentence level classifier is prone to select the incorrect sentences. After applying Triam Blocking on an encoder-based sentence-level classifier, the results have improved slightly, which indicates that there is redundant in the results. This occurs because when the parallel decoder is removed, the sentences selected by the method are independent without any dependency. This demonstrates that the parallel decoder plays an extremely important role in the SetSum model.

Effects of Bipartite Matching Loss In our experiments, we introduce bipartite matching loss to address the challenge of measuring the difference between the decoding results and the gold sentence set in an end-to-end manner. We investigate its effect in detail. Specifically, we compare bipartite matching loss with widely used cross-entropy loss. However, there is an issue that the number of decoder prediction M is larger than the gold label N, which makes it impossible to use the cross-entropy loss directly. To address the problem, we adopt two strategies to sort gold sentence labels: Fix Order and Random Order. Fix Order means we keep the original order of sentences in the source document and keep unchanged during training then pad to query size M. Random Order means we randomly sort gold sentence labels for each document before training then pad to query size M. From the results in Table 5, we find: (1) Compared with the Fix Order strategy, simply shuffling (Random Order) will not improve the performance. (2) Compared with Fix Order and Random Order, introducing bipartite matching loss gains 1.44, 0.67, and 1.22 in R-1, R-2, and R-L, respectively, which verifies the effectiveness of bipartite matching loss.

Settings	R-1	R-2	R-L
Full model	**25.49**	**6.39**	**20.33**
w/o Decoder	24.11	5.75	19.32
w/o Decoder + TriBlk	24.20	5.78	19.43
CELoss+Fix	24.52	5.92	19.77
CELoss+Randm	24.50	5.91	19.76
query freeze	24.21	5.51	19.31

Table 5: Ablation studies for the learnable sentence queries and bipartite matching loss.

Effects of Sentence Query The sentence query is the most important part of the SetSum, and we conduct a series of experiments to demonstrate its effectiveness. To explore the learning ability of entity queries, we freeze the parameter of sentence queries during training. The results are shown in Table 5 dropped by 1.31, 0.60, and 1.06 at R-1, R-2, and R-L, respectively, which indicates that the sentence queries do learn the patterns of summary. To verify the effect of query size M on the model, we conduct comparison experiments in Table 6. We can see that the effect of the model does not always increase as M is added, but first increases and then decreases. This occurs most likely because when M is small, the query cannot fully learn the pattern of summary, but as M becomes larger and larger, the query learns the noise instead. Eventually, the query number M in our experiments is set to 60.

Query Size (M)	R-1	R-2	R-L
40	24.75	6.05	19.75
60	**25.49**	**6.39**	**20.33**
80	24.82	5.99	19.93
100	24.90	5.96	19.90
200	24.70	6.13	19.76

Table 6: The performances with different numbers of sentence queries.

Effect of Different Decoder Layers To investigate the importance of the decoder, we explore the effect of decoder layers on the results, as shown in Table 7. We can see that the model performs better as the number of decoder layers increases. In general, the model is more capable of learning as the number of decoder layers is stacked. Furthermore, the results with the sentence level interaction are always better than the interaction at the token level, regardless of the number of layers, and the effect is more obvious as the number of layers increases. For example, when we change the interaction from Q-S to Q-T at layer 6, R-1 drops by 0.75, but at layer 4, R-1 only drops by 0.22.

Layer	Interaction	Reddit		
		R-1	R-2	R-L
2	Q-S	25.14	6.37	20.12
2	Q-T	25.03	6.24	20.23
4	Q-S	24.95	6.18	20.13
4	Q-T	24.73	5.86	19.72
6	**Q-S**	**25.49**	**6.39**	**20.33**
6	Q-T	24.71	6.03	19.72

Table 7: The performance of the decoder with different layers and interaction types. Q-S and Q-T denote cross-attention interaction between queries and sentence representation, token representation, respectively.

5.3 Human Evaluation

We also conduct human evaluation following (Chen et al., 2021) on CNN/DailyMail dataset. We invite 2 volunteers who are major in journalism to review the output summaries of several representative models independently. Specifically, we select 100 samples from the CNN/DailyMail dataset, volunteers are asked to rank summaries produced by BERTEXT (Liu and Lapata, 2019), MatchSum (Zhong et al., 2020) and our SetSum according to the following criteria: (1) Informativeness: The summary should preserve the main meaning of the original document (2) Coherence: The sentences in the summary should be coherent with each other. All of the systems were ranked by 1, 2, and 3 with 3, 2, and 1 scores, respectively. Finally, we get a weighted average score for each system to measure the overall quality of the summary. Results are shown in Table 8. From the results, we can see that the summary obtained by our SetSum outperforms other methods in terms of informativeness and coherence. In addition, the 4.95% improvement in coherence is more obvious than the 4.39% improvement in informativeness, which indicates that our system can learn more dependencies between summary sentences in addition to improving the informativeness of summaries. The results of human evaluation further validate the effectiveness of our method.

Models	Informativeness			
	1st	2nd	3rd	Avg R.
BERTEXT	0.23	0.35	0.42	1.81
MatchSum	0.35	0.35	0.30	2.05
SetSum	0.42	0.30	0.28	**2.14**
Models	**Coherence**			
	1st	2nd	3rd	Avg R.
BERTEXT	0.26	0.34	0.40	1.86
MatchSum	0.34	0.34	0.32	2.02
SetSum	0.40	0.32	0.28	**2.12**

Table 8: Human evaluation results on CNN/DaiyMail dataset. Avg R denotes the weighted average ranking score. The larger ranking score denotes better summary quality.

6 Conclusion

In this paper, we propose a set prediction network for extractive summarization task. Compared with previous sequence labeling methods, our approach formulates extractive summarization as a sentence set prediction problem. In our approach, a set of sentence queries are fed into a non-autoregressive decoder, which then predicts all sentences within the summary in parallel. To measure the difference between the parallel prediction results and the gold labels, we apply a bipartite matching loss to train the model. To demonstrate the effectiveness of our approach, we conduct experiments on single-document and multi-document datasets. The experimental results demonstrate that our method outperforms the previous state-of-the-art models.

Limitations

We propose a set prediction network for the extractive summarization task, which has worked well on some datasets but still has some limitations. Firstly, due to the use of pre-train BERT in the document encoder, our method is inadequate for long text summarization tasks. In general, the text length of a long document is much longer, so the model needs to be more capable to capture the dependency. Next, we will extend the method to long document summarization tasks. Secondly, the queries in the decoder are initialized with a normal distribution. If we can initialize the queries with the prior knowledge, our method may be able to find the set of sentences of the summary more accurately, which is another direction we need to focus on in the future.

Acknowledgements

This work is supported by the Key Research and Development Program of Zhejiang Province, China (No. 2023C01152), the Fundamental Research Funds for the Central Universities (No. 226-2022-00143), and MOE Engineering Research Center of Digital Library.

References

Chenxin An, Ming Zhong, Zhiyong Wu, Qin Zhu, Xuanjing Huang, and Xipeng Qiu. 2022. CoLo: A contrastive learning based re-ranking framework for one-stage summarization. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 5783–5793, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Sanghwan Bae, Taeuk Kim, Jihoon Kim, and Sanggoo Lee. 2019. Summary level training of sentence rewriting for abstractive summarization. In *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, pages 10–20, Hong Kong, China. Association for Computational Linguistics.

Moye Chen, Wei Li, Jiachen Liu, Xinyan Xiao, Hua Wu, and Haifeng Wang. 2021. SgSum:transforming multi-document summarization into sub-graph selection. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4063–4074, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

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.

Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, Walter Chang, and Nazli Goharian. 2018. A discourse-aware attention model for abstractive summarization of long documents. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 615–621, New Orleans, Louisiana. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Yue Dong, Yikang Shen, Eric Crawford, Herke van Hoof, and Jackie Chi Kit Cheung. 2018. BanditSum: Extractive summarization as a contextual bandit. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3739–3748, Brussels, Belgium. Association for Computational Linguistics.

Alexander Fabbri, Irene Li, Tianwei She, Suyi Li, and Dragomir Radev. 2019. Multi-news: A large-scale multi-document summarization dataset and abstractive hierarchical model. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1074–1084, Florence, Italy. Association for Computational Linguistics.

Nianlong Gu, Elliott Ash, and Richard Hahnloser. 2022. MemSum: Extractive summarization of long documents using multi-step episodic Markov decision processes. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6507–6522, Dublin, Ireland. Association for Computational Linguistics.

Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *NIPS*.

Ruipeng Jia, Yanan Cao, Haichao Shi, Fang Fang, Pengfei Yin, and Shi Wang. 2021. Flexible non-autoregressive extractive summarization with threshold: How to extract a non-fixed number of summary sentences. In *AAAI Conference on Artificial Intelligence*.

Byeongchang Kim, Hyunwoo Kim, and Gunhee Kim. 2019. Abstractive summarization of Reddit posts with multi-level memory networks. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics:*

Human Language Technologies, Volume 1 (Long and Short Papers), pages 2519–2531, Minneapolis, Minnesota. Association for Computational Linguistics.

Mahnaz Koupaee and William Yang Wang. 2018. Wikihow: A large scale text summarization dataset. *ArXiv*, abs/1810.09305.

Harold W. Kuhn. 1955. The hungarian method for the assignment problem. *Naval Research Logistics (NRL)*, 52.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *ACL 2004*.

Yang Liu and Mirella Lapata. 2019. Text summarization with pretrained encoders. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3730–3740, Hong Kong, China. Association for Computational Linguistics.

Yixin Liu and Pengfei Liu. 2021. SimCLS: A simple framework for contrastive learning of abstractive summarization. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 1065–1072, Online. Association for Computational Linguistics.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. *ArXiv*, abs/1611.04230.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018a. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807, Brussels, Belgium. Association for Computational Linguistics.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018b. Ranking sentences for extractive summarization with reinforcement learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1747–1759, New Orleans, Louisiana. Association for Computational Linguistics.

Shashi Narayan, Joshua Maynez, Jakub Adamek, Daniele Pighin, Blaz Bratanic, and Ryan McDonald. 2020. Stepwise extractive summarization and planning with structured transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4143–4159, Online. Association for Computational Linguistics.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. *ArXiv*, abs/1705.04304.

Peggy Tang, Kun Hu, Rui Yan, Lei Zhang, Junbin Gao, and Zhiyong Wang. 2022. OTExtSum: Extractive Text Summarisation with Optimal Transport. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 1128–1141, Seattle, United States. Association for Computational Linguistics.

Ashish Vaswani, Noam M. Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *ArXiv*, abs/1706.03762.

Wenhao Wu, Wei Li, Xinyan Xiao, Jiachen Liu, Ziqiang Cao, Sujian Li, Hua Wu, and Haifeng Wang. 2021. BASS: Boosting abstractive summarization with unified semantic graph. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6052–6067, Online. Association for Computational Linguistics.

Jiacheng Xu, Zhe Gan, Yu Cheng, and Jingjing Liu. 2020. Discourse-aware neural extractive text summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5021–5031, Online. Association for Computational Linguistics.

Yumo Xu and Mirella Lapata. 2022. Document summarization with latent queries. *Transactions of the Association for Computational Linguistics*, 10:623–638.

Yubo Zhang, Xingxing Zhang, Xun Wang, Si-Qing Chen, and Furu Wei. 2022. Latent prompt tuning for text summarization. *ArXiv*, abs/2211.01837.

Ming Zhong, Pengfei Liu, Yiran Chen, Danqing Wang, Xipeng Qiu, and Xuanjing Huang. 2020. Extractive summarization as text matching. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6197–6208, Online. Association for Computational Linguistics.

Qingyu Zhou, Nan Yang, Furu Wei, Shaohan Huang, M. Zhou, and Tiejun Zhao. 2018. Neural document summarization by jointly learning to score and select sentences. In *Annual Meeting of the Association for Computational Linguistics*.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
Limitations

☐ A2. Did you discuss any potential risks of your work?
Not applicable. Left blank.

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Abstract

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

Datasets and Evaluation Metrics

☑ B1. Did you cite the creators of artifacts you used?
Datasets and Evaluation Metrics

☒ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
Left blank.

☐ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
No response.

☐ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
Not applicable. Left blank.

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Datasets and Evaluation Metrics

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Table 1

C ☑ Did you run computational experiments?

Implementation Details

☒ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Our method is not a parametric sensitive method.

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Implementation Details

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Results and Analysis

☒ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
The package is open source

D ☑ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Human Evaluation

☑ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
Human Evaluation

☑ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
Human Evaluation

☑ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
Human Evaluation

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
Not applicable. Left blank.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
Not applicable. Left blank.