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 autoregressive 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) 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, 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.

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.

† Corresponding author.
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, 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.

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-
3.1 Task Formulation

Given a training sample \((D, G)\), where \(D = (s_1, s_2, \ldots, s_n)\) denotes the original document with \(n\) sentences, \(G\) denotes reference summary. Our goal is to select \(S^* = \{s_1^*, s_2^*, \ldots, s_m^*\}\) from \(D\) to cover the salient information of it, where \(m \leq n\). A set of gold sentences \(Y = \{<y'_i, y''_i, r'_i>\}_{i=0}^{m-1}\) is derived by a greedy selection strategy, where \(y'_i, y''_i \in [0, n - 1]\), \(y'_i \in \{0, 1\}\) 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 \(h = \{h_1, h_2, \ldots, h_n\}\).

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:

\[
h^s = \text{Transformer}(h) \quad (1)
\]

Finally, we get contextual representation for sentences \(h^s = \{h_1^s, h_2^s, \ldots, h_n^s\}\) 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:

\[
H^q = \text{Decoder}(e^q; h^s) \quad (2)
\]

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 \(h^q_i\) 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 \(h^q_i\), 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:

\[
h_{i,j}^{r/l} = \text{Tanh}(h^q_i w_i + h^s_j w_j) \quad (3)
\]

where \(w_i, w_j \in \mathbb{R}^{d \times d}\) are trainable projection parameters, \(r/l\) denotes left or right. Then we get the fusion representation of the \(i\)-th query with all sentence \(h_i^{r/l} = [h_{i,0}^{r/l}, h_{i,1}^{r/l}, \ldots, h_{i,m}^{r/l}]\).

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

\[
p_i^{r/l} = \text{Softmax}(h_i^{r/l}) \quad (4)
\]

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

\[
p_i^c = \frac{\exp(h_i^c w_i + b_i^c)}{\sum_{c' \in C} \exp(h_i^{c'} w_i + b_i^{c'})} \quad (5)
\]

where \(w_i^c\) and \(b_i^c\) are learnable parameter.
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 $\phi$ 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 $\phi$. Then the cost of assigning $\hat{Y}_i$ with $Y_j$ is defined as:

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

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

$$o^* = \text{argmin}_{o \in O_M} \sum_i C_{\text{match}}(\hat{Y}_{o(i)}, Y_j)$$

where $O_M$ is the space of all $M$-length permutations and $O_M$ 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:

$$\mathcal{L}(\hat{Y}, Y) = \sum_i \left\{ -\log p_{o^*(i)}^c(c_i) + 1_{\{c_i \neq \phi\}} \right\} + \left\{ -\log p_{o^*(i)}^l(l_i) - \log p_{o^*(i)}^r(r_i) \right\}$$

### 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
---|---|---|---|---|---
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. BERTTEXT (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: BERTTEXT + RL (Bae et al., 2019) directly maximizes summary-level ROUGE scores through reinforcement learning based on BERTTEXT. BERTTEXT + Tri-Blocking (Liu and Lapata, 2019) introduces trigram blocking during inference based on BERTTEXT. 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.
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).

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

<table>
<thead>
<tr>
<th>Methods</th>
<th>WikiHow</th>
<th>PubMed</th>
<th>Multi-News</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>R-1</td>
<td>R-2</td>
<td>R-L</td>
</tr>
<tr>
<td>LEAD</td>
<td>24.97</td>
<td>5.83</td>
<td>23.24</td>
</tr>
<tr>
<td>ORACLE</td>
<td>35.59</td>
<td>12.98</td>
<td>32.68</td>
</tr>
<tr>
<td>BERTTEXT</td>
<td>30.31</td>
<td>8.71</td>
<td>28.24</td>
</tr>
<tr>
<td>BERTTEXT+TriBlk</td>
<td>30.37</td>
<td>8.45</td>
<td>28.28</td>
</tr>
<tr>
<td>MatchSum</td>
<td>31.85</td>
<td>8.98</td>
<td>29.58</td>
</tr>
<tr>
<td>SetSum</td>
<td>31.66</td>
<td>8.72</td>
<td>29.36</td>
</tr>
</tbody>
</table>

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 \( M \). 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.

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

<table>
<thead>
<tr>
<th>Settings</th>
<th>R-1</th>
<th>R-2</th>
<th>R-L</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full model</td>
<td><strong>25.49</strong></td>
<td><strong>6.39</strong></td>
<td><strong>20.33</strong></td>
</tr>
<tr>
<td>w/o Decoder</td>
<td>24.11</td>
<td>5.75</td>
<td>19.32</td>
</tr>
<tr>
<td>w/o Decoder + TriBlk</td>
<td>24.20</td>
<td>5.78</td>
<td>19.43</td>
</tr>
<tr>
<td>CELoss+Fix</td>
<td>24.52</td>
<td>5.92</td>
<td>19.77</td>
</tr>
<tr>
<td>CELoss+Randm</td>
<td>24.50</td>
<td>5.91</td>
<td>19.76</td>
</tr>
<tr>
<td>query freeze</td>
<td>24.21</td>
<td>5.51</td>
<td>19.31</td>
</tr>
</tbody>
</table>

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.
<table>
<thead>
<tr>
<th>Query Size (M)</th>
<th>R-1</th>
<th>R-2</th>
<th>R-L</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>24.75</td>
<td>6.05</td>
<td>19.75</td>
</tr>
<tr>
<td><strong>60</strong></td>
<td><strong>25.49</strong></td>
<td><strong>6.39</strong></td>
<td><strong>20.33</strong></td>
</tr>
<tr>
<td>80</td>
<td>24.82</td>
<td>5.99</td>
<td>19.93</td>
</tr>
<tr>
<td>100</td>
<td>24.90</td>
<td>5.96</td>
<td>19.90</td>
</tr>
<tr>
<td>200</td>
<td>24.70</td>
<td>6.13</td>
<td>19.76</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Layer</th>
<th>Interaction</th>
<th>Reddit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>R-1</td>
</tr>
<tr>
<td>2</td>
<td>Q-S</td>
<td>25.14</td>
</tr>
<tr>
<td>2</td>
<td>Q-T</td>
<td>25.03</td>
</tr>
<tr>
<td>4</td>
<td>Q-S</td>
<td>24.95</td>
</tr>
<tr>
<td>4</td>
<td>Q-T</td>
<td>24.73</td>
</tr>
<tr>
<td><strong>6</strong></td>
<td><strong>Q-S</strong></td>
<td><strong>25.49</strong></td>
</tr>
<tr>
<td>6</td>
<td>Q-T</td>
<td>24.71</td>
</tr>
</tbody>
</table>

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 BERTx (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.

<table>
<thead>
<tr>
<th>Models</th>
<th>Informativeness</th>
<th>1st</th>
<th>2nd</th>
<th>3rd</th>
<th>Avg. R.</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERTx</td>
<td>0.23</td>
<td>0.35</td>
<td>0.42</td>
<td></td>
<td>1.81</td>
</tr>
<tr>
<td>MatchSum</td>
<td>0.35</td>
<td>0.35</td>
<td>0.30</td>
<td></td>
<td>2.05</td>
</tr>
<tr>
<td>SetSum</td>
<td>0.42</td>
<td>0.30</td>
<td>0.28</td>
<td></td>
<td>2.14</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Models</th>
<th>Coherence</th>
<th>1st</th>
<th>2nd</th>
<th>3rd</th>
<th>Avg. R.</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERTx</td>
<td>0.26</td>
<td>0.34</td>
<td>0.40</td>
<td></td>
<td>1.86</td>
</tr>
<tr>
<td>MatchSum</td>
<td>0.34</td>
<td>0.34</td>
<td>0.32</td>
<td></td>
<td>2.02</td>
</tr>
<tr>
<td>SetSum</td>
<td>0.40</td>
<td>0.32</td>
<td>0.28</td>
<td></td>
<td>2.12</td>
</tr>
</tbody>
</table>

Table 8: Human evaluation results on CNN/DailyMail 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


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.


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.