Title-based News Summarization via MRC Framework

Abstract

Existing studies on extractive summarization have adopted an approach that scores and selects summary sentences independently. However, these models are limited to sentence-level extraction and tend to select highly generalized sentences while overlooking the overall content of a document. In this study, we propose a machine reading comprehension (MRC) framework for extractive news summarization (MRCSUM) by setting a query as the title. This enables the model to consider the semantics of a compact summary when selecting summary sentences. In particular, when a title is not available, a TITLE-LIKE QUERY is generated, which is expected to achieve the same effect as a title. The experimental results demonstrate that MRCSUM is effective for the extractive summarization of a dataset with titles as well as without titles.

1 Introduction

Automatic text summarization compresses a document into fewer sentences while maintaining the essential information. This task is categorized into extractive and abstractive summarization depending on whether the summary sentences are newly generated. This study addresses extractive summarization because it is usually free from semantic, grammatical, and factual inconsistency problems (Zhong et al., 2020). Existing studies on extractive summarization have adopted two main approaches. The first method is that whereby extractive summarization is considered as a sequence-labeling problem, and the model determines whether each sentence is selected (Cheng and Lapata, 2016); (Nallapati et al., 2017). The second is the autoregressive method that was proposed to integrate sequence labeling into the autoregressive method (Zhou et al., 2018). It selects summary sentences based on their relative importance. Pretrained language models such as BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019) have exhibited surprisingly advanced performance in various NLP tasks. Liu and Lapata (2019) (BERTEXT) used BERT for extractive summarization. However, the above models are limited to sentence-level extraction, which leads to the selection of highly generalized sentences while the overall content of a document is overlooked (Zhong et al., 2020). Zhong et al. (2020) (MATCHSUM) proposed a summarization framework for extractive summarization that implements a semantic text matching scheme for a candidate summary and a document, in which it is assumed that a good summary is semantically similar to the document. We adopted this assumption and revised it as follows: a good summary is semantically similar to the title. Unlike previous studies, we used the title as opposed to the entire document to match the candidate summary semantically.

The first role of the title of a document is to provide a compact summary to the reader, and the second is to attract the reader to read the document (Senda and Shinohara, 2002). In the case of the first, a title can be regarded as compressing
2 MRCSUM

2.1 Task Definition

Given a single document \( D = \{s_1, s_2, ..., s_n\} \) consisting of \( n \) sentences, we must extract summary sentences in \( D \) by assigning a label \( L \in \{0, 1\} \) to \( s_i \). The label \( L \) indicates whether a sentence should be included in the summary.

2.2 Dataset Construction

First, we transform the tagging style of the summarization dataset into a set of (QUESTION, ANSWER, CONTEXT) triples. Let \( s_i = \{x_1, x_2, ..., x_{n_i}\} \), where \( n_i \) denotes the number of tokens in the \( i \)-th sentence; then, we can redefine a document \( D = \{x_1, x_2, ..., x_N\} \), where \( N \) denotes the number of tokens in a document \( (N = \sum_i n_i) \). A summary sentence \( x_{start,end} = \{x_{start}, ..., x_{end}\} \) is a substring of \( D \) satisfying \( start \leq \text{end} \) and \( \text{"start, end"} \) denotes the continuous tokens from the index ‘start’ to ‘end’. Subsequently, we can obtain the triple \((q_y, x_{start,end}, D)\), where \( q_y \) is the TITLE QUERY or TITLE-LIKE QUERY.

2.3 Query Generation

The question generation method is vital because queries encode a compact summary of a document and significantly influence the sentence selection.

TITLE QUERY

Given the title, we set it as a query, as follows:

\[
q_y = \text{Title}(D)
\]

\[
\text{Title}(D) = [w^t_1, w^t_2, ..., w^t_m],
\]

where \( m \) denotes the number of tokens in the title and \( w^t_i \) denotes the \( i \)-th token in the title.

TITLE-LIKE QUERY

When the title is not available, we construct the TITLE-LIKE QUERY using the topic and keywords of the document. We use LDA to assign the topic to the document, and KeyBERT and TextRank to extract the keywords from the document (Further details are in A.2). First, we allocate topics to the entire dataset using LDA. Thereafter, a topic embedding table is initialized for all topics and the details are in A.2).

\[
\text{Topic}(D) = W^{topic}[\text{topic}_D].
\]

where \( W^{topic} \in \mathbb{R}^{T \times H} \) is the embedding matrix that is fine-tuned during the training, and \( \text{topic}_D \) is...
the index of a topic. $T$ is the number of the topics
and $H$ is the hidden size. Second, we extract the
top five keywords from a single document, which
is used as a query along with the topic embedding.
The **TITLE-LIKE QUERY** consisting of the topic and
keywords is expressed as follows:

$$q_y = [\text{topic}_D, K_1, ..., K_5],$$

where $K_i$ denotes the $i$-th keyword. Finally, $q_y$ can
be redefined as follows:

$$q_y = \begin{cases} [W^t_1, ..., W^t_m], & \text{if given the title} \\ [\text{topic}_D, K_1, ..., K_5], & \text{Otherwise} \end{cases}$$

### 2.4 Model Details

**MRCSUM** extracts the span $x_{\text{start,end}}$, which is a
summary from $D$, given the question $q_y$. We use
a pretrained language model (Devlin et al., 2019;
Liu et al., 2019) as the encoder to encode a question
$q_y$ and a document $D = \{x_1, x_2, ..., x_N\}$. The
input sequence is the concatenation of question $q_y$
and document $D$, $\{[CLS], q_y, [SEP], x_1, ..., x_N\}$. The
pretrained language model outputs contextualized representations $E \in \mathbb{R}^{(N) \times d}$, where $d$ refers
to the hidden size.

#### 2.4.1 Summary Sentence Selection

**Start and End Index Prediction**  Given the contextualized representation matrix $E$, **MRCSUM** predicts the probability of each token being a start and an end index, as follows:

$$P_{\text{start}} = \text{softmax}_{\text{each row}}(E \cdot T_{\text{start}}) \in \mathbb{R}^{(N) \times 2}$$
$$P_{\text{end}} = \text{softmax}_{\text{each row}}(E \cdot T_{\text{end}}) \in \mathbb{R}^{(N) \times 2},$$

where $T_{\text{start}}, T_{\text{end}} \in \mathbb{R}^d$ are the weights to learn.
Each row of $P_{\text{start}}, P_{\text{end}}$ refers to the probability
distribution, which indicates whether each index is
the start and end position of a summary.

**Start–End Pairing**  Multiple summary sentences may exist within a single document. Therefore, **MRCSUM** must predict multiple start and end indices. In addition, it must match a predicted start index with its corresponding end index. To obtain the start and end indices (i.e., $\hat{I}_{\text{start}}$ and $\hat{I}_{\text{end}}$), argmax is applied to each row of $P_{\text{start}}$ and $P_{\text{end}}$:

$$\hat{I}_{\text{start}} = \{i | \text{argmax}(P_{\text{start}}^{(i)}) = 1, i = 1, ..., N\}$$
$$\hat{I}_{\text{end}} = \{i | \text{argmax}(P_{\text{end}}^{(i)}) = 1, i = 1, ..., N\},$$

where $(i)$ refers to the $i$-th row of a matrix. To pair the start index $i_{\text{start}} \in \hat{I}_{\text{start}}$ with its corresponding end index $i_{\text{end}} \in \hat{I}_{\text{end}}$, **MRCSUM** predicts the pairing score as follows:

$$P_{\text{start\_end}} = \text{sigmoid}(p \cdot \text{concat}(E_{\text{start}}; E_{\text{end}})),$$

where $p \in \mathbb{R}^{\frac{d}{2}}$ denotes the weights to learn.

#### 2.5 Training Method

Two label sequences, $Y_{\text{start}} = \{y_1^s, y_2^s, ..., y_N^s\}$
and $Y_{\text{end}} = \{y_1^e, y_2^e, ..., y_N^e\}$, need to be predicted by **MRCSUM** during training. Therefore, two losses are calculated for the start and end index predictions, as follows:

$$L_{\text{start}} = CE(P_{\text{start}}, Y_{\text{start}})$$
$$L_{\text{end}} = CE(P_{\text{end}}, Y_{\text{end}}),$$

where $CE$ denotes the cross-entropy. Another la-
bel sequence, $Y_{\text{start\_end}}$, indicates whether each
start index should be paired with each end index.
Therefore, we obtain the following start–end index
pairing loss:

$$L_{\text{span}} = CE(P_{\text{start\_end}}, Y_{\text{start\_end}}),$$

Finally, these losses are minimized as follows:

$$L = L_{\text{start}} + L_{\text{end}} + L_{\text{span}},$$

### 3 Experiments

#### 3.1 Experimental Setup

We evaluated our model using both English and
Korean datasets. For the English dataset, we
used CNN/DailyMail, which is a widely used
news summarization dataset modified by Nallapati
et al. (2017). We followed the same data-labeling
method as that in Liu and Lapata (2019). For the
Korean dataset, we used Modu-corpus\(^1\), which is
a collection of various task datasets collected by
the National Institute of Korean Language (NIKL)
(Kim et al., 2021). We used a news summarization
data set from Modu-corpus that we named Modu-
news. Modu-news is a dataset in which three ab-
stract summary sentences (i.e., a gold abstractive
summary) are generated by annotators who select
three extracted summary sentences (i.e., the gold
extractive summary) from the original news text
and paraphrase them. This dataset also contains the
title of a document. Statistics of Modu-news and
CNN/DailyMail are provided in A.4.

---

\(^1\)https://corpus.korean.go.kr/
3.2 Experimental Results

<table>
<thead>
<tr>
<th>Model</th>
<th>R-1</th>
<th>R-2</th>
<th>R-L</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEAD-3</td>
<td>55.84</td>
<td>38.83</td>
<td>40.59</td>
</tr>
<tr>
<td>ORACLE</td>
<td>75.57</td>
<td>63.50</td>
<td>66.76</td>
</tr>
<tr>
<td>MATCHSUM</td>
<td>56.65</td>
<td>38.47</td>
<td>41.67</td>
</tr>
<tr>
<td>MRCSUM TEXT RANK</td>
<td>56.91</td>
<td>40.58</td>
<td>43.07</td>
</tr>
<tr>
<td>MRCSUM KEYBERT</td>
<td>56.78</td>
<td>40.44</td>
<td>42.92</td>
</tr>
<tr>
<td>MRCSUM TITLE</td>
<td>57.82</td>
<td>41.83</td>
<td>44.03</td>
</tr>
</tbody>
</table>

Table 1: ROUGE F1 results on Modu-news test set. The average results of five runs with random initialization are displayed. \( p \)-value < 0.05

<table>
<thead>
<tr>
<th>Model</th>
<th>R-1</th>
<th>R-2</th>
<th>R-L</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEAD-3</td>
<td>40.43</td>
<td>17.62</td>
<td>36.67</td>
</tr>
<tr>
<td>ORACLE</td>
<td>52.59</td>
<td>31.23</td>
<td>48.87</td>
</tr>
<tr>
<td>BERT + Tri-Blocking</td>
<td>42.57</td>
<td>19.96</td>
<td>39.04</td>
</tr>
<tr>
<td>MATCHSUM (BERT-base)</td>
<td>44.22</td>
<td>20.62</td>
<td>40.38</td>
</tr>
<tr>
<td>MATCHSUM (RoBERTa-base)</td>
<td>44.41</td>
<td>20.86</td>
<td>40.55</td>
</tr>
<tr>
<td>MRCSUM (BERT-base)</td>
<td>44.77</td>
<td>21.01</td>
<td>40.63</td>
</tr>
<tr>
<td>MRCSUM (RoBERTa-base)</td>
<td>44.81</td>
<td>21.07</td>
<td>40.66</td>
</tr>
</tbody>
</table>

Table 2: ROUGE F1 results on CNN/DM test set. All results except for those of MRCSUM are cited from (MATCHSUM). The average results of five runs with random initialization are reported. \( p \)-value < 0.05

Table 3: Human evaluation results of summaries for 50 randomly sampled articles from Modu-news test set. The kappa ratio between evaluator scores was 0.42.

<table>
<thead>
<tr>
<th>Model</th>
<th>Informativeness</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEAD-3</td>
<td>0.81</td>
</tr>
<tr>
<td>ORACLE</td>
<td>1.43</td>
</tr>
<tr>
<td>MATCHSUM</td>
<td>1.15</td>
</tr>
<tr>
<td>MRCSUM TITLE</td>
<td>1.33</td>
</tr>
<tr>
<td>MRCSUM TEXT RANK</td>
<td>0.98</td>
</tr>
</tbody>
</table>

4 Conclusions

In this study, we have proposed MRCSUM, which considers the semantics of a compact summary because it trains the selection of summary sentences for the title. Moreover, when the title is not available, the TITLE-LIKE QUERY can be used. Our experimental results and human evaluations have demonstrated the effectiveness of our model for news summarization. In future work, we will explore distinguishing whether a news title is a summary or a teaser that attracts the reader.
Limitations

Note that our MRCSum primarily works when the title roles a compact summary of a news since we use the title of a news as a query to match with the summary semantically. Although we use the title-like query when the title is unavailable, it is less likely to be effective when the title is used to entice a potential reader. However, from a different point of view, it may be practical to use the title-like query when the title of a document is used for enticement. Therefore, we suggest distinguishing the title of a document whether it is a summary in future work.

References


A Appendix

A.1 Related Work

Extractive Summarization

In recent years, successful extractive summarization has been achieved using neural networks (Cheng and Lapata, 2016; Liu and Lapata, 2019; Zhong et al., 2020). The extractive summarization model mainly adopts an encoder–decoder structure that generates a vector representation of each sentence. The modeling of cross-sentence relations is one of the most effective methods for extracting appropriate sentences from a document, and it is generally achieved using recurrent neural networks (RNNs) (Cheng and Lapata, 2016; Nallapati et al., 2017; Zhou et al., 2018). However, models based on RNNs are limited to sentence-level long dependency, whereby a long document or multiple documents result in significant performance degradation. Another method for extracting the cross-sentence relations is to design a graph structure. Early traditional methods, such as LexRank (Erkan and Radev, 2004) and TextRank (Mihalcea and Tarau, 2004), computed the cosine similarity. Liu and Lapata (2019); Wang et al. (2020) established the encoder based on the Transformer, which trains the interaction between sentence pairs.

Query-Based Summarization

To generate a summary, query-based summarization (QBS) aims to structure sentences relating to the context of queries, and extractive techniques are common methods for conducting QBS. Otterbacher et al. (2009) suggested the form of question-answering extractive summarization based on Biased LexRank. Furthermore, Wang et al. (2013) presented the use of sparse trees and sentence compressions. Hermann et al. (2015) applied a neural network for QBS. However, extractive methods suffer from the problems of low coherence and less fluent summaries. To overcome this, in subsequent research on QBS, deep learning has been suggested as an approach to create a query-based abstractive summary (Baumel et al., 2018; Hasselqvist et al., 2017). Nema et al. (2017) addressed the issue of repeated phrases by using encoder–decoder based models while attempting to generate query-specific abstractive summaries. Xie et al. (2020) developed conditional self-attention to capture the conditional dependencies between given queries and input sequence pairs. Narayan et al. (2017, 2018); Li et al. (2018) used the side information to sum-
marize the text. Narayan et al. (2017) proposed a neural network for extractive summarization with side information such as title and the image captions. Narayan et al. (2018) used keywords from the input text to guide the process of summarization. These two works demonstrated that using the title and keywords of the document is helpful for summarization. Inspired by previous works (Narayan et al., 2017, 2018), we used the title, topic, and keywords of a document to consider the semantics of a compact summary.

A.2 Details for Assigning Topic and Extracting Keywords

**Topic Assign using LDA**

For using the LDA, we need to determine the number of topics. We set 10 and 65 topics for the Modu-news and CNN/DM datasets, respectively. Then, since the LDA produces a topic distribution, not a single topic, we choose the highest probability score topic and assign it to each document.

**Keywords Extracting**

When using TextRank, the keywords are extracted by the TF-IDF method in a single document. Then, TextRank produces a keyword distribution, and we choose the top 5 keywords. On the other hand, when using KeyBERT, we need two BERT models to encode a document and a combination of words, respectively. Each BERT model produces the representation of a document and that of a combination of words. Then, KeyBERT calculates the cosine similarity between these two representations. Same as TextRank, we choose the top 5 keywords.

A.3 Implementation Details

We implemented our model using the open-source PyTorch (Paszke et al., 2019) deep learning library. We adopted three types of pretrained language models for the shared encoder: BERT-base, RoBERTa-base, and RoBERTa-large. We set the batch sizes to 32 and 12 for the base and large models, respectively. We set the initial learning rate to 5e-5 for BERT-base and RoBERTa-base, and 5e-6 for RoBERTa-large. We conducted all experiments with an RTX 8000 GPU. Our code will be available on Github.

A.4 Human Evaluation Details

Table 4 shows a detailed description of CNN/DM and Modu-news datasets. For human evaluation, we sampled 50 articles from the dataset.

![Image](image.png)

**Figure 2:** Change in ROUGE score according to keyword recall ratio of title in TITLE-LIKE QUERY. This is the result for the Modu-news dataset.

Three people evaluated each article and summaries (e.g., MATCHSUM, MRCSUM-TITLE, MRCSUM-TEXRANK, LEAD-3, and a reference). We ask evaluators to score between 0 and 2 on how the summary is informative. A score of 0 refers to the summary not supporting an article’s important information. A score of 1 refers to the summary partially supporting an article’s important information. A score of 2 refers to the summary fully supporting an article’s important information.

A.5 Analysis of TITLE-LIKE Query

Figure 2 depicts the change in the ROUGE score according to the keyword recall ratio of the title in the TITLE-LIKE QUERY. It can be observed that when recall ratios were 0, the ROUGE scores were significantly higher than 0.2. The reason is that sometimes the author writes titles abstractly. Moreover, a higher keyword recall ratio results in a higher ROUGE score. This suggests that it is vital to extract words that imply the semantics of a document, such as the keywords in the title.