IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received May 26, 2021, accepted July 1, 2021, date of publication July 14, 2021, date of current version July 29, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3097087

Reinforced Abstractive Text Summarization With Semantic Added Reward

HEEWON JANG AND WOOJU KIM
Department of Industrial Engineering, Yonsei University, Seoul 03722, South Korea
Corresponding author: Wooju Kim (wkim@yonsei.ac.kr)

ABSTRACT Text summarization is an important task in natural language processing (NLP). Neural summary models summarize information by understanding and rewriting documents through the encoder-decoder structure. Recent studies have sought to overcome the bias that cross-entropy-based learning methods can have through reinforcement learning (RL)-based learning methods or the problem of failing to learn optimized for metrics. However, the ROUGE metric with only *n*-gram matching is not a perfect solution. The purpose of this study is to improve the quality of the summary statement by proposing a reward function used in text summarization based on RL. We propose ROUGE-SIM and ROUGE-WMD, modified functions of the ROUGE function. ROUGE-SIM enables meaningfully similar words, in contrast to ROUGE-L. ROUGE-WMD is a function adding semantic similarity to ROUGE-L. The semantic similarity between articles and summary text was computed using Word Mover's Distance (WMD) methodology. Our model with two proposed reward functions demonstrated superior performance on ROUGE-1, ROUGE-2, and ROUGE_L than on ROUGE-L as a reward function. Our two models, ROUGE-SIM and ROUGE-WMD, scored 0.418 and 0.406 for ROUGE-L, respectively, for the Gigaword dataset. The two reward functions outperformed ROUGE-L even in the abstractiveness and grammatical aspects.

INDEX TERMS Text summarization, abstract summarization, reinforcement learning, semantic similarity.

I. INTRODUCTION

Text Summarization is an important task in natural language processing (NLP). Reducing the amount of text containing information can be helpful in various NLP tasks. Text summarization is divided into two methods, extractive summarization and abstractive summarization. For extractive summarization, only necessary words are copied from the input text [1]–[4]. In contrast, abstract textual summarization is a task of rewriting based on the core idea of a document. This uses human-like processes to generate text, making it more likely to generate fluent text and using new words not included in the input text [10].

Recent abstractive summarization models are based on sequence-to-sequence neural networks [5], [7], [8], [10], [33], [34], [35]. They are made up of encoders to understand input sequence and decoders to generate output sequence. They demonstrated higher performance than that in previous studies. But there are four main problems with using sequence-to-sequence neural networks to generate good text: (1) Out of Vocabulary(OOV) problem (2) generating a particular word or phrase repeatedly, (3) exposure bias at test time, and (4) non-optimized learning for evaluation metrics [14] used by models in fields such as text summarization and machine translation.

Many studies have been conducted to improve the structure of models to improve the quality of summarization by addressing these problems. The pointer mechanism copies some elements of the input sequence in the decoding process to solve the OOV problem [11]–[13]. Coverage loss [13] and intra-attention [15] are proposed for the problem of generating the same syntax repeatedly. In our experiment, we tried to reduce the iterative problem using the intra-decoder attention proposed by Paulus *et al.* [15]. Various studies have also been conducted in optimization methodology. Cross-entropy methodologies can be biased and have problems learning not to optimize performance metrics. As an answer to this, a methodology using RL has been proposed to solve both problems in a limited manner. Using ROUGE-L as a reward function [15] cannot be completely free from the bias problem because ROUGE must use the same vocabulary as the reference summary and the generated summary in the same order to produce high scores. The reference summary data affects the model's robustness because there is a limit to the training with metrics such as ROUGE-L that do not reflect the semantic value. We address these issues by proposing two

The associate editor coordinating the review of this manuscript and approving it for publication was Sergio Consoli.

reward functions that add the semantic value to ROUGE-L, used in RL-based abstractive text summarization.

1) ROUGE-SIM: A metric that modifies the ROUGE metric based on an n-gram match. When finding the length of the longest common subsequence, use the similarity of word embedding to allow matching of similar words.

2) ROUGE-WMD: Reward function that mixes Word-Mover's Distance (WMD) [16] and ROUGE-L.

Our proposed reward functions helped us select a more diverse vocabulary in the generation process, such as abbreviations or similar words, compared with ROUGE-L. The model's robustness was high because repetition was reduced, and fewer grammatical errors were evaluated manually with GRAMMARLY—it was possible to receive a lower penalty as the reward even when selecting a word that differed from the reference when replacing with the content. Most of the grammatical errors were caused by repetitive problems, which improved significantly without post-processing.

II. RELATED WORK

Most of text summarization works in the past have an extractive approach [1]–[4], [9]. They performed tasks by copying necessary words or sentences from input text or by compressing necessary elements.

Encoder-decoder-based neural networks are structures that understand inputs and generate outputs, such as the fields of machine translation [5] and text summarization [8], [10]. They are used in various NLP fields. An encoder is a structure that obtains the context representation. The input sequence is converted into a fixed vector using word embedding. Word2vec [19] and Glove [20] are used primarily as word embedding to convert natural language into fixed vectors. And the fixed vector is encoded in context representation through structures such as long short-term memory (LSTM) [17], convolution neural network and Transformer [18]. The attention mechanism enables the encoder-decoder model to see different parts of the input sequence by decoding the time step [6]. Various studies have also been conducted on the problem of repetitive generation of words such as the OOV problem that occurs overall in text generation tasks including summarization. See *et al.* [13] tried to prevent the generation of unknown words by adding the probability of copying the words appearing in the input sequence through the pointer generator, like extractive summarization, to the generation probability of the decoder. In order to solve the problem that the same word or phrase is repeatedly generated, coverage loss [13] and intra-attention [15] have been proposed.

Since Transformer [18] was proposed, various pre-trained language models such as Elmo [21], Bert [22], and Big Bird [23], have been published. Large Transformer models pre-trained on large text corpora perform well in natural language processing with only limited supervision samples. Song *et al.* [35] finetuned the BERT by setting seen and unseen words. This enhances the ability to generate unknown words similar to masking, providing good results

TABLE 1. Parameters list.

Notation	Definition
$\boldsymbol{x}$	Input sequence, $\boldsymbol{x} = \{\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_n}\}$
$\boldsymbol{y}$	Output sequence, $\boldsymbol{y} = \{\boldsymbol{y_1}, \boldsymbol{y_2}, \ldots, \boldsymbol{y_m}\}$
$\boldsymbol{e_{ti}}$	Attention score
a_{ti}	Normalized attention score
c	Context vector
h	Hidden state vector

on the Gigaword dataset. The pre-trained encoder-decoder model also demonstrates good performance in text summarization [32]–[34]. Unlike other masked language models, PEGASUS [33] proposes a pre-training scheme for text summarization, which masks sentences from input documents and predicts masked sentences.

RL trains agents to recognize a given environment and function in a way that maximizes rewards. In sequence generation tasks, traditional cross-entropy methodologies cannot be trained by reflecting evaluation metrics because metrics such as BLEU and ROUGE are undifferentiable. Traditional learning methods that are not optimized for metric can be biased and affect the performance of the model. Ranzato *et al.* [14] used the REINFORCE algorithm [24], [25] in the sequence generation model to optimize for the metric. Rennie *et al.* [26] proposed a self-critical sequence training method that does not require critic model and showed great improvement on image captioning tasks. In abstractive summarization task, Paulus *et al.* [15] train a model optimized for ROUGE, significantly improving the ROUGE recall of the model. Wang *et al.* [34] has shown good results in Gigaword dataset as an optimized model for metrics with added topic information.

III. MODEL

Our model is based on simple LSTM sequence-to-sequence model with Attention [6], a pointer mechanism for handling Out of Vocabulary (OOV) words [13], and intra-decoder attention for handling repeated words [15]. The parameters used in the following descriptions are as shown in Table 1.

A. ARCHITECTURE

1) SEQUENCE-TO-SEQUENCE ATTENTION

We used a single-layer bi-directional LSTM as the encoder and a single-layer LSTM as the decoder. We calculated the hidden state of the input sentence $\mathrm{h}_\mathrm{i}^\mathrm{e}$ and the hidden state of the decoder $\mathrm{h}_\mathrm{t}^\mathrm{d}$ by using bi-directional LSTM encoder, LSTM decoder and the attention mechanism.

$$e_{ti}^{e} = h_{t-1}^{d} W_{attn}^{e} h_{i}^{e} \tag{1}$$

$$a_{ti}^{e} = \frac{exp(e_{ti}^{e})}{\sum_{i=1}^{n} exp(e_{i}^{e})} \tag{2}$$

$$c_t^e = \sum_{i=1}^{n} a_{ti}^d h_i^e \tag{3}$$
$$h_t^d = f(h_{t-1}, y_{t-1}, c_t^e) \tag{4}$$

We define e_{ti}^e as the attention score of the hidden input state h_t^d at decoding time step t, a_{ti}^e as a normalized distribution and c_t^e as context vector at decoding time step t. We can calculate the hidden state of the decoding timestep t through the processes of (1) – (4) where f is LSTM function.

2) INTRA-DECODER ATTENTION

Intra-decoder attention is a structure to prevent the decoder from generating a repeated phrase [15]. This structure creates a context vector incorporating information about the sequence decoded in the previous timestep during the word generation process. By using the context vector of the decoder when generating a word, information about the word representations created in the past is used in the decoding process and repeated phrases are reduced.

$$e_{tt'}^d = h_t^{d^T} W_{attn}^d h_{t'}^d \tag{5}$$
$$a_{tt'}^d = \frac{exp(e_{tt'}^d)}{\sum_{j=1}^{t-1} exp(e_{tj}^d)} \tag{6}$$
$$c_t^d = \sum_{j=1}^{t-1} a_{tj}^d h_j^d \tag{7}$$

A decoder context vector (c_t^d) is calculated for each decoding timestep t. The calculation process is the same as (5)–(7).

3) POINTER GENERATOR

Pointer Generator is a structure to solve the OOV problem by copying unseen or rare words that are difficult to generate from the input sequence.

$$p(y_t|u_t = 0) = softmax(W_{out}\left[h_t^d\right]\left[c_t^d\right] + b_{out}) \tag{8}$$
$$p(y_t = x_i|u_t = 1) = a_{ti}^e \tag{9}$$
$$p(u_t = 1) = \delta(W_u\left[h_t^d\right]\left[c_t^d\right] + b_u) \tag{10}$$
$$p(y_t) = p(u_t = 1)\, p(y_t \mid u_t = 1) + p(u_t = 0)p(y_t|u_t = 0) \tag{11}$$

(8) and (9) are probability distributions of word generation that can be obtained from our token-generation decoder and attention, respectively. (10) is the ratio of the copy mechanism used in the decoding timestep t where δ is a sigmoid function. With (8), (9), and (10), we can obtain the final word generation probability distribution (11) where $p(u_t = 0) = 1 - p(u_t = 1)$.

B. POLICY LEARNING

Policy gradients using RL can solve metric-optimized learning and bias problems [24], [25]. However, the ROUGE-L metric, which is most often used in the abstractive text summarization task, is an index proportional to the longest common subsequence. and there is another problem that a high ROUGE-L score does not guarantee improved readable

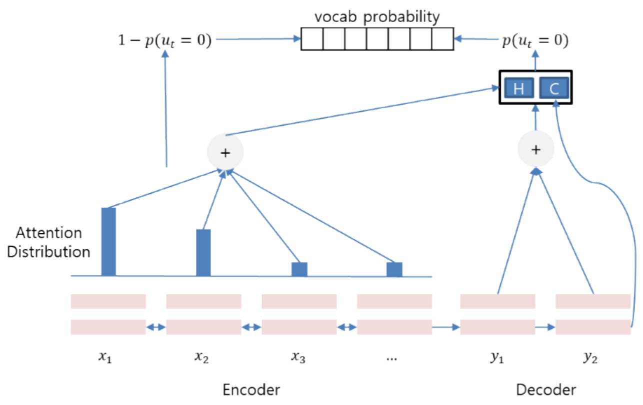


FIGURE 1. Illustration of model structure. The decoder context vector (C) and the decoder hidden state (H) are calculated.

text [30].

$$L_{ml} = -\sum_{t=1}^{n'} logp\left(y_t^* \mid y_1^* \ldots, y_{t-1}^*, x\right)) \tag{12}$$
$$L_{rl} = \left(r\left(\hat{y}\right) - r\left(y^s\right)\right) \sum_{t=1}^{n'} logp\left(y_t^s \mid y_1^s \ldots, y_{t-1}^s, x\right)) \tag{13}$$
$$L_{mixed} = \gamma L_{rl} + (1-\gamma)L_{ml} \tag{14}$$

$\hat{y}$ is the maximum probability sentence, y^s is the sampled sentence, and y^* is the reference summary. So, we use L_{mixed} to maintain the readability of the generated sentence [15], [27], [28]. γ is the scaling factor, and 0.998 is used in the experiment.

C. REWARD FUNCTION

1) ROUGE-L

ROUGE is an indicator for evaluating natural language generative models such as text summarization and machine translation, measuring performance through comparisons with answer sets. ROUGE-L is one of the ROUGE metrics that can be computed as depicted in (15)∼(18) using the Longest Common Subsequence (LCS) of the ground truth summary and the generated summary. β is defined in [29]. In the text summarization task, the precision of ROUGE-L is the brevity of the generated summary and its ability to recall how much information the generated summary contains from the reference summary.

$$LCS\left(X_i, Y_j\right) = \begin{cases} 0 \\ \quad if\ i = 0 or j = 0 \\ LCS\left(X_{i-1}, Y_{j-1}\right) + 1 \\ \quad if\ x_i = y_j \\ max\left(LCS\left(X_i, Y_{j-1}\right), LCS\left(X_{i-1}, Y_j\right)\right) \\ \quad if\ x_i \neq y_j \end{cases} \tag{15}$$

$$precision = \frac{LCS\left(X_i, Y_j\right)}{total\ words\ in\ system\ summary} \tag{16}$$

$$recall = \frac{LCS\left(X_i, Y_j\right)}{total\ words\ in\ reference\ summary} \tag{17}$$

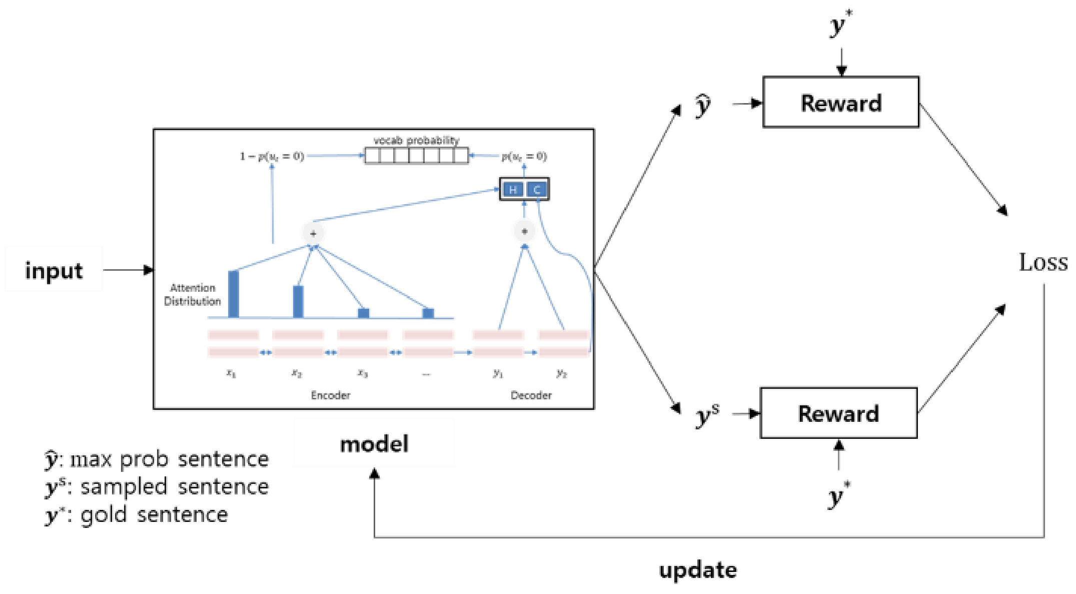


FIGURE 2. Illustration of policy learning.

$$F1 = \frac{(1+\beta^2)precision * recall}{\beta^2 * precision + recall} \tag{18}$$

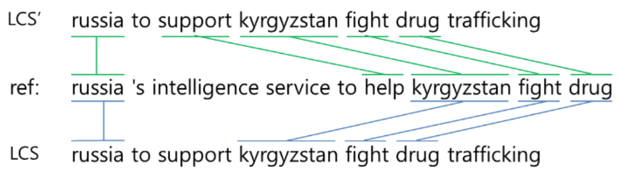
Paulus *et al.* found that ROUGE-L outperformed ROUGE-1 and ROUGE-2 in their experiment [15]. Therefore, we use ROUGE-L for our base model. When using ROUGE-L as a reward function, the length of the LCS must be lengthened to receive a high reward score, and it is advantageous to generate the words included in the reference summary in order. Therefore, as the learning progresses, only the results biased to the reference summary earn high reward scores, while the same meaning summary does not adequately reflect the semantic score. We introduce two new reward functions to address these issues.

2) ROUGE-SIM

ROUGE-SIM is a metric modified from ROUGE-L. In contrast to counting exactly-matching words as when calculating the LCS of ROUGE-L, ROUGE-SIM does not penalize the length of LCS for words with a similarity of 0.8 or more to words appearing in the text.

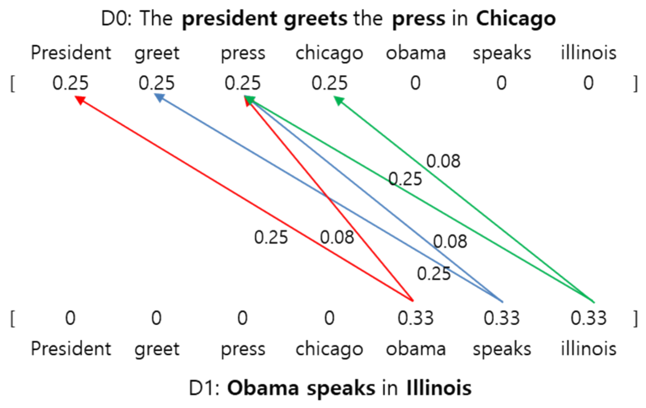
$$LCS'(X_i, Y_j) = \begin{cases} 0 \\ \quad if\ i = 0 or j = 0 \\ LCS'(X_{i-1}, Y_{j-1}) + 1 \\ \quad if\ sim\ (e(x_i), e(y_j)) > 0.8 \\ max\left(LCS'(X_i, Y_{j-1}), LCS'(X_{i-1}, Y_j)\right) \\ \quad if\ x_i \neq y_j \end{cases} \tag{19}$$

Figure 3 illustrates the LCS' and LCS of the reference summary and the generated summary. The LCS, which counts only words that match exactly, is calculated as 4. On the other hand, when the similarity between "support" and "help" in the context is 0.8 or higher, LCS' is calculated as 5.



LCS'
russia to support kyrgyzstan fight drug trafficking
ref:
russia 's intelligence service to help kyrgyzstan fight drug
LCS
russia to support kyrgyzstan fight drug trafficking

FIGURE 3. Difference between LCS and LCS. LCS' counts not only words that match exactly, but also words with high similarity (>0.8).



D0: The president greets the press in Chicago
President greet press chicago obama speaks illinois
[0.25 0.25 0.25 0.25 0 0 0]
0.08
0.25
0.25
0.08
0.08
0.25
[0 0 0 0 0.33 0.33 0.33]
President greet press chicago obama speaks illinois
D1: Obama speaks in Illinois

FIGURE 4. Illustration of word mover's distance.

By allowing semantic matching like (19), we try to reduce the penalty for using similar words and overcome the lower abstractiveness problem and the bias problem that may occur in the learning process. e(x) denotes an embedding vector of x. We used the encoder hidden state as an embedding. The f1 score of ROUGE-SIM using LCS′ can be calculated as the same process shown in (16) $\sim$ (18).

3) ROUGE-WMD

Word Mover Distance (WMD) measures the semantic distance between two documents using word embedding [15]. As depicted in Figure 4, a document can be expressed as the distribution of words included in the document except for stopwords. The semantic distance is calculated as the cost of moving all of one distribution to another. Consequently, the semantic distance between two documents is calculated as the minimum of the cumulative sum of the distances between words included in each document.

$$distance = min_{\mathbf{T}\geq 0} \sum\nolimits_{i,j=1}^{n} T_{i,j}c(i,j) \tag{20}$$

$$\sum\nolimits_{j=1}^{n} T_{ij} = d_i \tag{21}$$

$$\sum\nolimits_{i=1}^{n} T_{ij} = d'_j \tag{22}$$

c(i, j) means the Euclidean distance of the embedding vectors of the word i and word j. $T_{i,j}$ is a flow matrix, and each element means the ratio of moving from word i to word j. The semantic distance of the two documents is obtained by finding $T_{i,j}$ where the optimal distance has the minimum value. For example, in Figure 4, "obama" in D1 has a ratio of 0.33, of which a minimum cost occurs when 0.25 is transferred to "president" and 0.08 to "press". Similarly, other words have a minimum cost when transferred in the same proportion as shown in Figure 4. We used $\frac{1}{distance+1}$ + ROUGE-L as the reward function. Glove [20] was used for word embedding to obtain WMD.

IV. EXPERIMENT SETUP

A. DATASET & EXPERIMENT SETUP

We use Gigaword summarization dataset used by Rush *et al.* [7]. It consists of pairs of short articles (31.4 tokens) and titles (8.3 tokens). It contains 3.8M training, 189k development, and 1,951 test instances. The encoder and decoder lengths are set to 55 and 15, respectively, so that more than 99% of the total data can be used to consider the data length.

B. EVALUATION METRIC

We conducted quantitative evaluations, abstractiveness evaluations, and grammatically robust evaluations. For quantitative analysis, the F1 scores of ROUGE-1, ROUGE-2, and ROUGE-L were reported using ROUGE-PACKAGE [29]. We evaluate the abstractiveness of the results using novel n-grams. Novel n-grams are indicators of the percentage of n-grams in the generated summary that are not included in the input sequence (original article). We reported the novelty for 1, 2, and 4-gram in See *et al.* [13]. Novel n-gram refers to the proportion of elements in the n-gram set of generated summaries that do not appear in input text.

Moreover, for grammatic analysis, we sampled and used 500 articles of the test data. First, the analysis was conducted using GRAMMARLY [31], and the evaluation was conducted manually to supplement the parts that the program could not handle, such as acceptable errors used in the reference summary or not-reported errors caused by phrase repetition. Because the decoded summary sentences have different lengths, we counted the number of grammatical errors that appeared per 100 words.

TABLE 2. Quantitative analysis.

Method	ROUGE		
	1	2	L
Pointer Generator [13]	**0.376**	**0.163**	**0.344**
PEGASUSLARGE [33]	**0.388**	**0.200**	**0.361**
ProphetNet [32]	**0.396**	**0.203**	**0.366**
RL-Topic-ConvS2s [34]	**0.369**	**0.183**	**0.346**
ControlCopying [35]	**0.392**	**0.204**	**0.367**
ROUGE-L	**0.409**	**0.124**	**0.373**
ROUGE-SIM	**0.455**	**0.236**	**0.418**
ROUGE-*WMD*	**0.442**	**0.233**	**0.406**

V. RESULT

We evaluated the performance of text summarization using the proposed reward function in three aspects: quantitative evaluation using ROUGE-PAKAGE [29], evaluation of abstractiveness by the ratio of newly emerged *n*-grams [13], and evaluation of grammatical errors using GRAMMARLY [31].

A. QUANTITATIVE ANALYSIS

The baseline models for quantitative evaluation include the following: **Pointer Generator** [13] is an RNN model using pointer mechanism and coverage loss. **PEGASUS** [33] is an encoder-decoder-based Transformer model trained with masked sentences. **ProphetNet** [32] is an encoder-decoder-based Transformer model optimized by n-gram prediction based on previous context tokens at each time step. **RL-Topic-ConvS2s** [34] is reinforcement learning models using topic information. **ControlCoping** [35] is a fine tuning model of Bert [22] by masking seen and unseen word separately. Baseline models except **Pointer Generator** are based on Transformer-based large language model. They may show lower performance in evaluations using ROUGE-PAKAGE, as there may be more cases of words that can be generated. **ROUGE-L** shows better performance on ROUGE-1 score and ROUGE-L score, while **ROUGE-SIM**, **ROUGE-WMD** show better performance on all ROUGE scores compared to all baseline models. This shows that using reinforcement learning performs better in terms of ROUGE score. Furthermore, we show that the semantic value using similarity of summary is performed better by comparing it with **RL-Topic-ConvS2s** than the reward function using topic. **ROUGE-SIM** outperformed slightly for ROUGE-1, ROUGE-2, and ROUGE-L compared with **ROUGE-WMD**. We can see that the two experiments with the sematic value added reward function significantly reduce the gap between the results of the train set and the test set.

Article	a microlight aircraft crashed near blenheim in the south island of new zealand late sunday afternoon, killing the pilot.
Ref	microlight aircraft crashes in new zealand killing pilot
ROUGE-L	microlight aircraft pilot south south island aircraft in new zealand pilot in new zealand
ROUGE-SIM	microlight aircraft crashes in new zealand killing pilot
ROUGE-*WMD*	pilot killed in plane crash in new zealand
Article	prosecutors charged an environmental activist wednesday with murder and illegal possession of firearms in the killing of dutch populist leader pim fortuyn.
Ref	assassination suspect charged with murder for shooting populist
ROUGE-L	prosecutors prosecutors dutch dutch prosecutors charged environmental activist with murder in dutch populist leader
ROUGE-SIM	environmental activist charged with murder in killing of dutch populist leader
ROUGE-*WMD*	environmental activist charged in killing of dutch populist leader

FIGURE 5. Examples of generated summary. Red letters represent repeated phrases.

TABLE 3. Abstractiveness analysis.

Method	Novelty		
	1	2	4
Reference Summary	**0.029**	**0.171**	**0.455**
Pointer Generator [13]	**0.022**	**0.060**	**0.097**
ROUGE-L	**0.0007**	**0.005**	**0.232**
ROUGE-SIM	**0.0053**	**0.033**	**0.182**
ROUGE-*WMD*	**0.0131**	**0.058**	**0.232**

B. ABSTRACTIVENESS ANALYSIS

We used the ratio of n-grams that do not appear in the article as an index to evaluate abstractiveness, such as the experiment of See *et al.* [13]. All experiments, including **Pointer Generator** model, have novelty values that are 1, 2, 4 gram lower than the **Reference Summary** which is generated by humans. Among the proposed methods, **ROUGE-WMD** has a lower value in 4-gram novelty than **ROUGE-L**, but shows better expression in 1,2 gram. The frequency of using abbreviations or synonyms not included in the input sentence was high. **ROUGE-WMD** was higher in 4-gram novelty than **ROUGE-SIM**.

C. GRAMMATICAL ANALYSIS

We sampled 500 cases in the test dataset to measure the grammatical error. We primarily measured grammatical errors using GRAMMARLY, and manually verified errors. For the base model, most of the errors are caused by repetition of phrase. As a result of manual verification, the text generated by the model of **ROUGE-WMD** has significantly less repetition. This is because the proportion of repeated words increases relatively when unnecessary repetition occurs. The distribution of the system summary distorted by repetition is thought to naturally disappear during the learning process, as the transition cost increases in comparison with the distribution of the reference summary.

TABLE 4. Grammatical analysis.

Method	Errors per 1000 words
ROUGE-L	**16.48**
ROUGE-SIM	**4.02**
ROUGE-WMD	**3.10**

VI. CONCLUSION

We present two reward functions, ROUGE-SIM and ROUGE-WMD, that add semantic values to functions based on conventional n-gram matching. Experiments in the Architecture Analysis part show that the decoding architecture matches the reward function. As in the results of Quantitative Analysis, the proposed models using architecture and the return function were less biased than other experiments. Our model performed better than sequence-to-sequence based models, Transformer based pre-learning models, and other reinforcement learning based models. We also show better results in Abstractness Analysis, and Grammatical Analysis. Although there were fewer new representations than human-written summaries in Abstractness Analysis, new representations were the most frequent among models using pointer mechanisms. Furthermore, we show improvements in terms of readability as grammatical errors (especially iterations) are almost eliminated compared to models using ROUGE-L in Grammatical Analysis.

However, this study has some limitations. First, we use is news data, and paraphrasing is not a big problem. This can be a problem for documents with jargon, so verification is required in documents from various domains. Secondly, we used a single-layer LSTM which has relatively low number of trainable weights to validate decoding architectures and reward functions. We need to verify whether our proposal can improve the performance of Transformer-based large models.

REFERENCES

[1] K. Knight and D. Marcu, "Summarization beyond sentence extraction: A probabilistic approach to sentence compression," *Artif. Intell.*, vol. 139, no. 1, pp. 91–107, Jul. 2002.

[2] B. Dorr, D. Zajic, and R. Schwartz, "Hedge trimmer: A parse-and-trim approach to headline generation," in *Proc. HLT-NAACL Text Summarization Workshop*, Jan. 2003, pp. 1–4.

[3] J. Clarke and M. Lapata, "Global inference for sentence compression: An integer linear programming approach," *J. Artif. Intell. Res.*, vol. 31, pp. 399–429, Mar. 2008.

[4] R. Nallapati, F. Zhai, and B. Zhou, "SummaRuNNer: A recurrent neural network based sequence model for extractive summarization of documents," in *Proc. 31st AAAI Conf.*, 2016, pp. 1–7.

[5] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 3104–3112.

[6] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," 2014, *arXiv:1409.0473*. [Online]. Available: http://arxiv.org/abs/1409.0473

[7] A. M. Rush, S. Chopra, and J. Weston, "A neural attention model for abstractive sentence summarization," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2015, pp. 379–389.

[8] S. Chopra, M. Auli, and A. M. Rush, "Abstractive sentence summarization with attentive recurrent neural networks," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol.*, 2016, pp. 93–98.

[9] Y. Miao and P. Blunsom, "Language as a latent variable: Discrete generative models for sentence compression," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2016, pp. 1–10.

[10] R. Nallapati, B. Zhou, C. dos Santos, C. Gulcehre, and B. Xiang, "Abstractive text summarization using sequence-to-sequence RNNs and beyond," in *Proc. 20th SIGNLL Conf. Comput. Natural Lang. Learn.*, 2016, pp. 1–12.

[11] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer networks," *Comput. Sci.*, vol. 28, no. 2015, pp. 1–9, 2015.

[12] C. Gulcehre, S. Ahn, R. Nallapati, B. Zhou, and Y. Bengio, "Pointing the unknown words," in *Proc. 54th Annu. Meeting Assoc. Comput. Linguistics*, 2016, pp. 1–10.

[13] A. See, P. J. Liu, and C. D. Manning, "Get to the point: Summarization with pointer-generator networks," in *Proc. 55th Annu. Meeting Assoc. Comput. Linguistics*, vol. 1, Jul. 2017, pp. 1073–1083.

[14] M. Ranzato, S. Chopra, M. Auli, and W. Zaremba, "Sequence level training with recurrent neural networks," 2015, *arXiv:1511.06732*. [Online]. Available: http://arxiv.org/abs/1511.06732

[15] R. Paulus, C. Xiong, and R. Socher, "A deep reinforced model for abstractive summarization," 2017, *arXiv:1705.04304*. [Online]. Available: http://arxiv.org/abs/1705.04304

[16] M. Kusner, Y. Sun, N. Kolkin, and K. Weinberger, "From word embeddings to document distances," in *Proc. 32nd Int. Conf. Mach. Learn.*, 2015, pp. 957–966. [Online]. Available: http://proceedings.mlr.press/v37/kusnerb15.html

[17] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[18] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.

[19] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," 2013, *arXiv:1310.4546*. [Online]. Available: http://arxiv.org/abs/1310.4546

[20] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2014, pp. 1–12.

[21] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," 2018, *arXiv:1802.05365*. [Online]. Available: http://arxiv.org/abs/1802.05365

[22] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," 2018, *arXiv:1810.04805*. [Online]. Available: http://arxiv.org/abs/1810.04805

[23] M. Zaheer, G. Guruganesh, A. Dubey, J. Ainslie, C. Alberti, S. Ontanon, P. Pham, A. Ravula, Q. Wang, L. Yang, and A. Ahmed, "Big bird: Transformers for longer sequences," 2020, *arXiv:2007.14062*. [Online]. Available: http://arxiv.org/abs/2007.14062

[24] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 229–256, 1992.

[25] W. Zaremba and I. Sutskever, "Reinforcement learning neural Turing machines-revised," 2015, *arXiv:1505.00521*. [Online]. Available: http://arxiv.org/abs/1505.00521

[26] S. J. Rennie, E. Marcheret, Y. Mroueh, J. Ross, and V. Goel, "Self-critical sequence training for image captioning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 7008–7024.

[27] R. Pasunuru and M. Bansal, "Reinforced video captioning with entailment rewards," 2017, *arXiv:1708.02300*. [Online]. Available: http://arxiv.org/abs/1708.02300

[28] Y. Wu *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation," 2016, *arXiv:1609.08144*. [Online]. Available: http://arxiv.org/abs/1609.08144

[29] C.-Y. Lin and E. Hovy, "Manual and automatic evaluation of summaries," in *Proc. Workshop Autom. Summarization*, 2002, pp. 1–8.

[30] C.-W. Liu, R. Lowe, I. Serban, M. Noseworthy, L. Charlin, and J. Pineau, "How NOT to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2016, pp. 1–15.

[31] *Write Your Best With Grammarly*. Accessed: Mar. 26, 2021. [Online]. Available: https://www.grammarly.com/

[32] W. Qi, Y. Yan, Y. Gong, D. Liu, N. Duan, J. Chen, R. Zhang, and M. Zhou, "ProphetNet: Predicting future N-gram for sequence-to-sequence pre-training," 2020, *arXiv:2001.04063*. [Online]. Available: http://arxiv.org/abs/2001.04063

[33] J. Zhang, Y. Zhao, M. Saleh, and P. J. Liu, "PEGASUS: Pre-training with extracted gap-sentences for abstractive summarization," *Proc. 37th Int. Conf. Mach. Learn.* Vienna, Austria: PMLR, vol. 119, 2020, pp. 11328–11339.

[34] L. Wang, J. Yao, Y. Tao, L. Zhong, W. Liu, and Q. Du, "A reinforced topic-aware convolutional sequence-to-sequence model for abstractive text summarization," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, Jul. 2018, pp. 1–8.

[35] K. Song, B. Wang, Z. Feng, R. Liu, and F. Liu, "Controlling the amount of verbatim copying in abstractive summarization," in *Proc. AAAI Conf. Artif. Intell.*, 2020, vol. 34, no. 5, pp. 8902–8909.

HEEWON JANG received the M.S. degree in industrial engineering from Yonsei University, in 2016, where he is currently pursuing the Ph.D. degree. His main research interests include natural language processing, machine learning, and artificial intelligence.

WOOJU KIM received the Ph.D. degree in operations research from KAIST, South Korea, in 1994. He is currently a Professor with the School of Industrial Engineering, Yonsei University. His main research interests include natural language processing, reliable knowledge discovery, big data intelligence, machine learning, and artificial intelligence.