J Inf Process Syst, Vol.18, No.3, pp.344~358, June 2022
https://doi.org/10.3745/JIPS.02.0172

ISSN 1976-913X (Print)
ISSN 2092-805X (Electronic)

JOURNAL OF INFORMATION PROCESSING SYSTEMS JIPS

Improving Abstractive Summarization by Training Masked Out-of-Vocabulary Words

Tae-Seok Lee[1], Hyun-Young Lee[2], and Seung-Shik Kang[2,*]

Abstract

Text summarization is the task of producing a shorter version of a long document while accurately preserving the main contents of the original text. Abstractive summarization generates novel words and phrases using a language generation method through text transformation and prior-embedded word information. However, newly coined words or out-of-vocabulary words decrease the performance of automatic summarization because they are not pre-trained in the machine learning process. In this study, we demonstrated an improvement in summarization quality through the contextualized embedding of BERT with out-of-vocabulary masking. In addition, explicitly providing precise pointing and an optional copy instruction along with BERT embedding, we achieved an increased accuracy than the baseline model. The recall-based word-generation metric ROUGE-1 score was 55.11 and the word-order-based ROUGE-L score was 39.65.

Keywords

BERT, Deep Learning, Generative Summarization, Selective OOV Copy Model, Unknown Words

1. Introduction

Text summarization in natural language processing is a challenging task. It aims to produce a shorter version of a long document that encapsulates the main information [1,2]. There are two summarization approaches: extractive and abstractive summarization. Extractive summarization simply extracts representative sentences that are relevant to a document [3,4]. The importance of a sentence within the document is calculated based on the location of words and their relationship to the neighboring words. Keywords or sentences are identified by considering the thematic role of a word and the grammatical structure of a sentence. The task of machine reading comprehension (MRC) is similar to extractive summarization. It answers questions by reading a document, extracting a specific passage, and highlighting the relevant portion of the document. Abstractive summarization is preferred when key sentences in the document do not represent the overall meaning of the entire document. In such a scenario, summarization is performed by analyzing the content of the document, thereby generating new sentences that are not included in the original text [5,6].

Bidirectional encoder representations from transformers (BERT) embedding is widely used for machine reading comprehension and natural language understanding [7]. Neural networks are good models for

Manuscript received April 23, 2021; first revision July 7, 2021; accepted December 3, 2021.

* **Corresponding Author:** Seung-Shik Kang (sskang@kookmin.ac.kr)

[1] Korea Institute of Science and Technology Information, Seoul, Korea (tsyi@kisti.re.kr)

[2] Dept. of Computer Science, Kookmin University, Seoul, Korea (hyunyoung2@kookmin.ac.kr, sskang@kookmin.ac.kr)

implementing abstractive summarization [8,9] and have been applied to various tasks in natural language processing, such as text classification and machine reading comprehension [10]. Recurrent neural networks (RNNs) are popular models for processing sequential information. An RNN uses the t^{th} input and the $t-1^{\text{th}}$ hidden state to create an output for the t^{th} input. This method maintains the sequential characteristics of the sentence naturally [11]. It reads a sentence from the first word instead of from the last word. However, an RNN is weak at capturing long-term dependency. Transformers address this issue by using the self-attention method instead [12-14]. In this method, words from each sentence are vectorized through the self-attention mechanism. The transformer uses positional encoding for the relative positions of a word along with the embedding of the word. This allows the model to learn relative-position information. Therefore, the embedding vector varies according to the position of a word and the context, even for the same words. It can deal with cases where the same word has different meanings.

The BERT embedding model represents a word by only using the encoder portion of the transformer. BERT input embedding uses position embedding instead of positional encoding. One-hot index embedding is used based on the position of a word, after which sentence and word embedding are added sequentially. BERT learns through a masked-language model and the next-sentence prediction that consider all words in an entire sentence before and after the input word through self-attention at each step of the sequential information. Thus, BERT embedding considers all the words in a sentence and also learns the semantic relationship of the sentences in a document.

The neural network-based abstractive summarization model uses a dictionary focusing on words frequently found in the training data. Rare words such as jargon face issues in being represented as out-of-vocabulary words. When humans encounter rare words in a document, they may summarize the document based on the meaning of surrounding sentences and through personal experience by looking up related information. Previous studies use a copy mechanism to address this challenge [15,16]. In this mechanism, out-of-vocabulary (OOV) words are collected from the pointed sub-sequences of the input document. It has been adopted in a way to increase the performance by adding a gate that determines a generation or pointing a word. Keywords in the input document are selected through this gate and the pointing positions are given selectively.

Our research will demonstrate the possibility of achieving better accuracy for abstractive summarization in the Korean language. A pre-trained Korean language model is improved by providing OOV words based on training data using a selective OOV copy method. We focus on two aspects. The first one is to build a training data set for text summarization, which works well for technical documents such as academic papers. The other is to check if our proposed model properly generates the OOV words in terms of summarization performance. We apply the following steps to improve the accuracy:

- Mask OOV words in the training stage,
- Add contextual and morphological information in the embedding model,
- Apply precise pointing and an optional copy instruction.

In the preprocessing stage, rare words are substituted with <unk> tags in order to train the OOV model. A masked-OOV (MOOV) training method is applied to the masked-language model, which intentionally distorts the input in BERT. Based on the input document, we perform a context-based word embedding to select and generate a summary through syntactic and semantic features. The sentence piece tokenizer that is used for the public BERT model does not utilize the morphological information of the Korean language. Therefore, we combine a pre-trained BERT model with a Korean morphological analyzer for performance enhancement. Korean academic papers are collected as a training data set. They are reliable documents and the author's keywords are used to construct an OOV vocabulary for training the OOV model. We improve the summarization accuracy by using the additional information that is selectively pointing or generating.

This paper is structured as follows. In Section 2, we describe the related works and explain the selective-pointing OOV copy model in Section 3. Section 4 presents how to use BERT embedding and masked OOV words. The experimental results are given in Section 5. Finally, we conclude the paper in Section 6.

2. Related Work

2.1 Seq2seq Copy Mechanism

The copy mechanism adopts a sequence-to-sequence (seq2seq) model to detect and copy OOV words from an input document [15]. Fig. 1 shows that a summarization method selects OOV items by using the copy mechanism in the decoding process. Determining a word by adding copy-attention scores involves deciding whether to generate it normally or to copy it from an input document. A higher copy-attention score indicates that the influence of a specific part of the input document is high [16]. The copy mechanism may copy the word from an input document even though it is supposed to be generated in the normal summarization process.

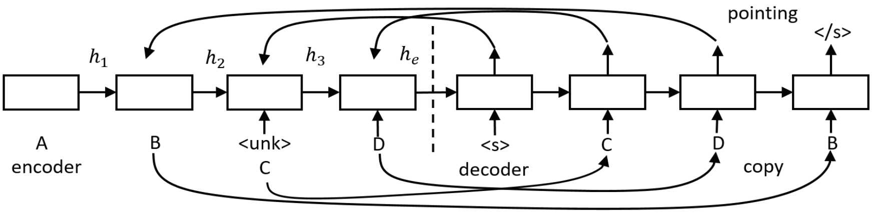


Fig. 1. Copy mechanism for abstractive summarization.

2.2 Selective-Pointing Copy Mechanism

In order to improve the copy mechanism, Nallapati et al. added position information by providing a criterion with generator g and OOV pointer p as in Fig. 2. They are used to determine whether to copy the word of the input document or to generate a word during the summarization process [17]. By using the copy mechanism with necessary words selectively, the copy-attention score is not considered for the words in the lexicon.

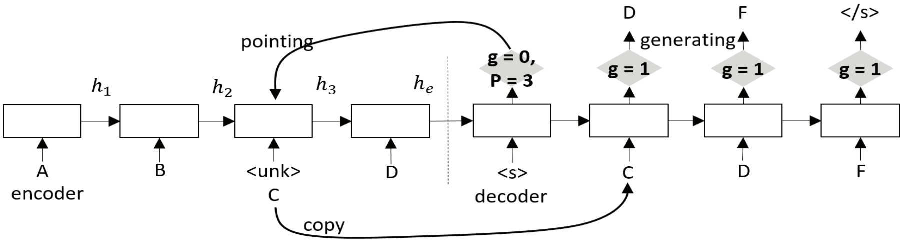


Fig. 2. Selective-pointing copy mechanism.

3. Selective-Pointing OOV Copy Model

We construct a long short-term memory (LSTM) OOV copy model with BERT embedding, which is designed to extract words by selectively generating or pointing the words. The input to the model incorporates the result of BERT context embedding. Each word, with separated morphemes, is provided to the LSTM OOV copy model through BERT embedding. We use an LSTM cell in an automatic-summary neural network, which is a seq2seq model. The encoding process of the model is shown in Fig. 3. The left side of the diagram shows an input document as a vector, sequentially receiving the words of each input sentence, while the right side shows the sequential decoding of the summary starting from the final hidden state of the encoding step, i.e., the initial value. The decoder also estimates the pointer p, the generator g, and the generation error of summary words. It optimizes the model through the loss function. The first of three LSTM encoding hidden layers uses a bidirectional RNN; bidirectional information is processed at each timestep. The three encoding layers have three corresponding decoding layers. If a value g_2 is 1, then the model generates summary words. If it is 0, the model points out $x_{2,}$ i.e., one of the input words in Fig. 3.

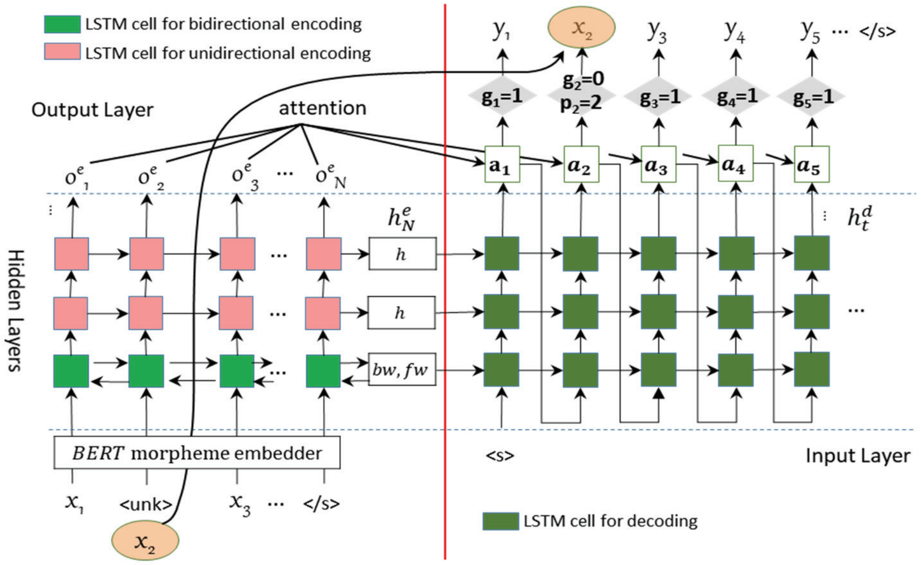


Fig. 3. Selective-pointing OOV copy model with BERT embedding. x is input word; y, output word; p_t input-word-selection number; h^e, h^d, hidden-state neural network matrices; N, the total number of timesteps.

An attention layer in Fig. 4 is required for precisely pointing to OOV words. At the decoding timestep t, an appropriate weight a is calculated through a neural network as input with decoder hidden state h_t^d and encoder hiddern outputs o_i^e. g_t is quantified from the <unk> tag information. The <unk> tag contains the correct-answer summary. It informs the decoder whether to generate or copy the words. When g_t=1, it generates a word. When g_t=0, it copies a word that receives the highest pointing attention. For each decoding timestep, the model calculates c (attention-weighted context vector) and a (attention weight) which are determined based on the concentration of a specific word in the entire encoding. c_t is designed to generate a word at the current timestep (att_wcv) to be used as input to the next timestep.

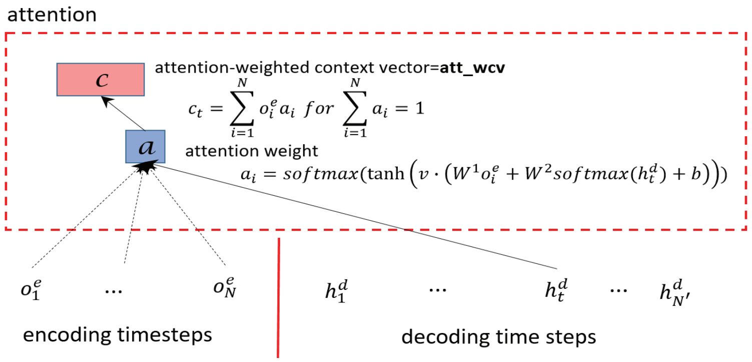

attention
c
attention-weighted context vector=**att_wcv**
$c_t = \sum_{i=1}^{N} o_i^e a_i \ for \ \sum_{i=1}^{N} a_i = 1$
a
attention weight
$a_i = softmax(\tanh(v \cdot (W^1 o_i^e + W^2 softmax(h_t^d) + b)))$
o_1^e … o_N^e
h_1^d … h_t^d … $h_{N'}^d$
encoding timesteps
decoding time steps

Fig. 4. Selective-pointing attention mechanism. *W* and *b* are weight and bias, respectively.

The overall flow of the decoding step is illustrated in Fig. 5. '*logits*' is a probability value providing the possibility of correct summary-word generation. Summary-word generation results are printed out as a lexicon index (vocab_id). Used for computing a and c in the attention layer, $\{o^e\}$ is a set of all output values from the encoding step. p_t is an index value of the largest a value. In the decoding step, the input words for training vary from the input words for prediction. While training the model, training input words are the correct summary words included in the training data. Prediction uses the words generated in the previous step. Each input word and att_wcv are added in the next timestep to correspond to the changes in the attention information at each decoding step.

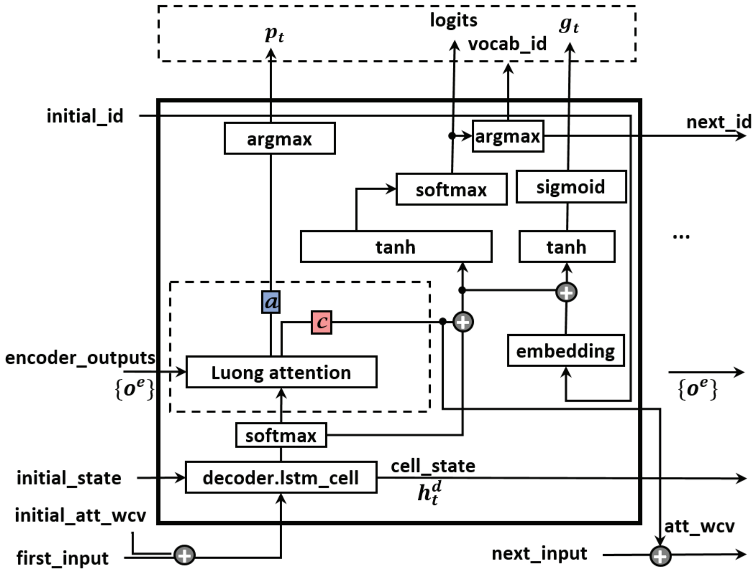


Fig. 5. Decoding step of summary word generation.

Eq. (1) is a probability function that corresponds to the position value p_t of the keyword along with g_t (the value that decides to generate a word). It provides additional information to train the encoder, decoder, and OOV pointing-attention neural network. If g_t=0, then pointing-attention is executed; otherwise, a summary-word is generated for g_t=1.

$$\begin{gathered}
P(Y|X) = \sum_{t} \begin{pmatrix} g_t P(y_t|Y_{t-1},X) \\ +(1-g_t)P(p_t|Y_{t-1},X) \end{pmatrix} \\
P(y_t|Y_{t-1},X) = softmax\left(tanh\left(W\mathrm{h}_t^d + Wc_{t-1} + Wy_{t-1} + b\right)\right) \\
P(p_t|Y_{t-1},X) = softmax\left(tanh\left(W\{o^e\} + W\mathrm{h}_t^d + b\right)\right) \\
g_t = sigmoid(tanh(Wh_t^d + Wc_t + Wy_{t-1} + b))
\end{gathered} \tag{1}$$

Eq. (1) describes a conditional probability that generates the summary words Y for the encoding of input words X. Y_{t-1} indicates a state where decoding is processed at the t–1th timestep. For summary word generation, the next-word prediction probability function, $P(y_t|Y_{t-1},X)$, is activated. In the encoding side, the pointing probability function, $P(p_t|Y_{t-1},X)$, is activated for the pointing to a word. To optimize the weight of the neural network, a loss function is defined by applying the natural log to Eq. (1) with a negative value of the cross-entropy error function, i.e., Eq. (2).

$$-\log P(y|x) = \sum_{t} \begin{pmatrix} g_t\{-\log P(y_t|Y_{t-1},X)\} + \\ (1-g_t)\{-\log P(p_t|Y_{t-1},X)\} \end{pmatrix} \tag{2}$$

Negative log-likelihood is a neural-network loss function that is used to minimize the difference between a correct answer and a prediction. The loss function in Eq. (3) is described by combining the generating function for summary word $generation\_NLL$ and the pointing function $pointing\_NLL$ that directs the words in the input document. $generation\_NLL$ is a loss function that learns to match the output of the decoder with the word in actual summary, while $pointing\_NLL$ is a loss function ensuring the target summary words. The position of a word in the input document is accurately pointed assuming that the word generated in the summary is not in the lexicon. These functions normalize the output of a neural network and are expressed as cross-entropy functions for the error between prediction and the correct answer.

$$\text{loss} = \sum_{t} \begin{pmatrix} g_t \cdot generation\_NLL + \\ (1-g_t) \cdot pointing\_NLL \end{pmatrix} \tag{3}$$

4. BERT Embedding with Masked OOV Words

The OOV model is trained for academic papers. The BERT model is trained for MRC corpus in the National Information Society Agency (http://aihub.or.kr), which contains 107,717 news articles. After preprocessing, the BERT model is generated by pre-training the documents excluding academic papers. Then, it is fine-tuned through OOV model training. Finally, the performance is evaluated by using the Recall-Oriented Understudy for Gisting Evaluation (ROUGE) metric [18]. We also collected 4,707 academic papers on computer science published by the Korean Institute of Information Scientists and Engineers (http://www.kiise.or.kr) and crawled through the NDSL website (http://www.ndsl.kr) operated by the Korea Institute of Science and Technology Information (http://www.kisti.or.kr) (Table 1). The collected dataset is randomly divided into training, evaluation, and test data in the ratio 6:1:3 for the validation experiment.

Table 1. Data collection of academic papers

Journal title	Duration	Count
Journal of KIISE	2014–2017	211
Journal of KIISE: Transactions on Computing Practices	2014–2017	138
Journal of KIISE: Database	2002–2014	601
Journal of KIISE: Software and Applications	2002–2014	1,291
Journal of KIISE: Computing Practices and Letters	2007–2014	981
Journal of KIISE: Information Networking	2002–2014	729
Journal of KIISE: Computer Systems and Theory	2002–2014	756
Total	-	4,707

4.1 Bidirectional Encoder Representations from Transformers

BERT embedding is created through pre-training using data generated for a masked language model and the next-sentence prediction training. The BERT pre-training data, extracted from news articles in sentence units, is composed of 1,497,079 lines. BERT model training was conducted with options presented in Table 2. Considering the mean length of sentences in the training documents, max_seq_length is set as 128 words.

Table 2. Hyperparameters for BERT

Variable	Value
attention_probs_dropout_prob	0.1
directionality	bidi
hidden_act	gelu
hidden_dropout_prob	0.1
hidden_size	256
initializer_range	0.02
intermediate_size	128
max_position_embeddings	653
num_attention_heads	4
num_hidden_layers	6
type_vocab_size	2
vocab_size	202,777

The masked language parameter is set to mask approximately 15% of all words. The hyperparameters of the BERT model, i.e., hidden_size, num_attention_heads, and num_hidden_layers, are set to 256, 4, and 2–12, respectively, considering the device performance. With these settings, the size of one hidden attention becomes 64. The size of the feed-forward hidden neural network, which collects the multi-head attention output in the transformer encoding block, is set to 128. A Korean morphological analyzer, MeCab-ko, was used at the input layer of BERT. The best pre-training results were achieved with six hidden layers. The accuracy of the masked language model was 0.5267 and that of next-sentence prediction was 0.9875 (Fig. 6). Although the BERT-based model has 12 hidden layers, only six hidden layers are used in our BERT model because of the small set of documents collected.

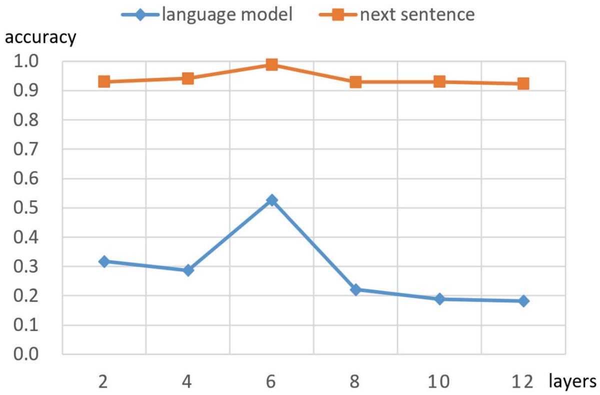


Fig. 6. Performance comparison of language model according to number of BERT hidden layers.

4.2 Training Masked OOV Words

Scientific article summarization differs from news article or general document summarization because scientific papers are typically long and contain complex concepts and technical terms [19]. Erera et al. [20] described a typical scenario for scientific paper usage through a qualitative user study. Users first read the title, and if relevant, continue to read the abstract. Thus, the title is considered a summary of the abstract. Abstractive summarization with OOV words requires a list of keywords that are added to the input document. Keywords may or may not appear in the abstract but are essential for learning the summarization pattern. In general, peer-reviewed articles follow a specific document type or expression and have keywords selected by the authors. Thus, the title and abstract of the articles are the summary statement and input document. We assume that the title of the article summarizes the contents of the paper and the keywords selected by the authors are the most important keywords of the paper. Inverse document frequency scores are calculated for each keyword, and the three keywords with the highest scores are selected as OOV words.

The training process comprises seven steps. In step 1, tokenization is performed by the morphological analyzer. In step 2, OOV words are substituted with <unk> tags and a (OOV word, <unk>) table is constructed. Step 3 generates a lexicon for indexing OOV words. In step 4, the OOV dictionary is generated to replace the <unk> tag with the actual word. In step 5, indexing is conducted by using the lexicon. Step 6 generates a selective gate's correct answer to determine whether to generate or to copy the word. Finally, step 7 creates an indicator for the correct answer to accurately point to the OOV word.

We replace words with <unk> tags considering the author's keywords in order to train the OOV model. The model is not trained in a context where all the OOV keywords are substituted with <unk> tags because it does not consider the occurrence pattern of the <unk> token. It should be trained in a context in accordance with sentences and words that occur before or after the <unk> tag. Therefore, we devised a MOOV training method based on masked language modeling, which randomly shows masked words in BERT. We use a MOOV model for the regularization of OOV words [21]. OOV words are substituted with the <unk> tag, retained, or replaced with a random word depending on the probabilities P1, P2, and P3, respectively (Fig. 7). The experimental results showed the highest performance when P1:P2:P3 is 9:1:0 (Table 3).

- P1: substituting with <unk>

ex) NAND 형 메모리 압축 (NAND type memory compression)

→ <unk>형 메모리 압축 (<unk> type memory compression)

- P2: retaining the original word

ex) NAND 형 메모리 압축 (NAND type memory compression)

→ NAND 형 메모리 압축 (NAND type memory compression)

- P3: replacing the original word with a random word

ex) NAND 형 메모리 압축 (NAND type memory compression)

→ 계산 형 메모리 압축 (calculation type memory compression)

Fig. 7. Examples of converting masked OOV words.

Unlike the BERT masked language model, substituting with random words was not meaningful for training the MOOV model. The noise effect in training the OOV model was to keep the original word or to replace the OOV word with an <unk> tag. The noise sensitivity was different because BERT produces a high-dimensional vector of words, whereas OOV model training calculated an exact position value and an indicating value of generating or selecting a word, depending on the context.

Table 3. Summarization performance

Model	ROUGE				
	L	1	2	SU4	h.mean
LSTM search $g_t + p_t$ feeding	29.55	47.01	5.95	17.80	14.32
LSTM+MOOV (8:2:0)	26.58	48.89	6.78	19.74	15.61
LSTM+MOOV (8:1:1)	26.28	47.51	8.84	20.24	18.05
LSTM+MOOV (9:1:0)	32.44	51.27	10.39	22.00	20.83
LSTM+MOOV (9:0.5:0.5)	29.47	49.38	10.08	20.83	19.87

MOOV=selective masked-OOV copy model, h.mean=harmonic mean.

4.3 Morphological Decomposition and Composition

Parts-of-speech (POS) information is added in the training stage by the Korean morphological analyzer, which separates postpositions or endings from a word. We combined sentences and words by adding POS tags for roots, postpositions, and endings, which were generated in the prediction stage as listed in Table 4. Formal morphemes correspond to parts of speech such as case markers, verb endings, and suffixes. They are a subset of Korean parts of speech to perform word embeddings. Furthermore, symbols such as "·-o·" (terminal type) or "·--·" (linking type) are used as information to reconstruct a complete sentence in the prediction stage. We did not consider prefixes and compound nouns because they complicate the model too much.

Table 4. Formal morphemes and POS tags

Formal morpheme	POS tag
Case markers	JKS, JKC, JKG, JKO, JKB, JKV, JKQ, JX, JC
Verb endings	EP, EF, EC, ETN, ETM
Prefixes and suffixes	XPN, XSN, XSV, XSA

Restoring sentences generated as morpheme units, we construct words by connecting the roots and the postpositions or endings by adding a space when two roots are adjacent to each other. In other words, we add a space before the roots and connect others by identifying the roots and the postpositions or endings. The stepwise process of morphological decomposition and generating sentences is: (1) Morphologically separate an input sentence using the morphological analyzer. (2) Add a morpheme symbol. (3) Add a space before the elements that correspond to the roots in the result of (2). (4) Concatenate all the elements of a list. (5) Remove symbols such as "·-o·" or "·--·". As a result, a sentence identical to the input sentence is obtained. In this way, we applied the morpheme-to-sentence converter module to the results generated by the selectively-pointing OOV model.

5. Experiments and Results

5.1 Experiment Settings

Information omission is important for evaluating the performance of abstractive summarization. It is determined by comparing the generated summary with the correct answer. In this study, we used the ROUGE metric. There are many varieties of ROUGE based on factors such as the appearance of correct words, the length of word-order matching, and the number of matched words. We used ROUGE-L, ROUGE-1, ROUGE-2, and ROUGE -SU4. We experimented on a PC server with specifications tabulated in Table 5, which shows the hyperparameters used to perform training. Dropout and hierarchical normalization were used to enhance model performance. LSTM cells were implemented in three layers in encoding and decoding by using a pre-trained BERT model.

Table 5. Hyperparameters for OOV copy model

Variable	Value
BERT hidden layers	6
BERT hidden size	256
Batch size	32
LSTM layers	3
Max encoding time-steps	600
Max decoding time-steps	50
Min input length	2
Hidden node size for LSTM cell	200
Embedding size	256

We regenerated MOOV data after every 2-epoch train is completed and proceeded to perform training again. The MOOV was created randomly to enhance the training effect each time. In the model training, the validation losses were measured every 30 minutes. When fine-tuning was performed in combination with the language model, we changed the learning rate three times based on the research results that the method of adjusting the learning rate can reduce the learning time [22-25]. Fig. 8 shows the loss graph of fine-tuning our entire model. We began training our model using the learning rate 2×10^{-5} after initializing with the pre-trained BERT model and then retrained the model using the learning rate 2×10^{-4} and 2×10^{-3} at the point where the training progress is slowed. Through training, we made an optimal model in 190,000 steps where the validation loss began to increase. The validation loss continued to increase because overfitting occurred after these steps. We generated summary sentences by using beam search with the trained model [26].

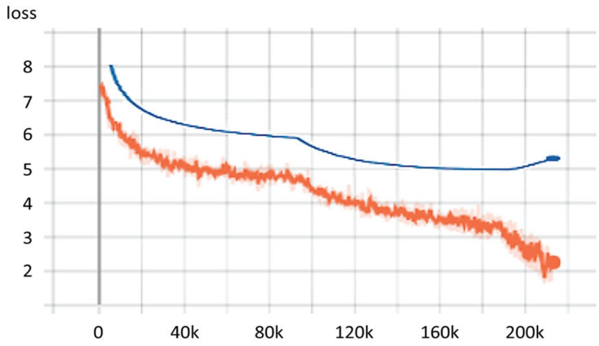


Fig. 8. Loss graph of BERT OOV model (up curve, validation loss; down curve, training loss).

5.2 Performance Evaluation

We evaluated the performance of the proposed model using n-gram-based ROUGE-N, longest-common-subsequence-based ROUGE-L, skip-bigram co-occurrence, and unigram-based ROUGE-SU. We found that the performance was improved compared to baseline models, such as the LSTM model, LSTM+p_t model, LSTM+g_t+p_t model, and LSTM+g_t+p_t+MOOV model. We evaluated six models trained using the same dataset. All the baseline models except the BERT+LSTM+g_t+p_t+MOOV model and BERT+LSTM+g_t+p_t used lookup tables that are optimized during model training without a separate embedding representation layer. First, the LSTM model was constructed with three LSTM layers and a lookup embedding table to summarize the document. The LSTM+p_t model performed summaries by using the copy mechanism for all <unk> tags. The LSTM+g_t+p_t model performed summaries using the g_t andp_t information for <unk> tags and selectively executed the copy mechanism. The LSTM+g_t +p_t+MOOV model was the LSTM+g_t+p_t model trained by MOOV.

Table 6 summarizes the performance measurements of ROUGE after obtaining a generative summary using each model with a test document as input. When BERT embedding was combined with an LSTM search (g_t+p_t) model, the highest performance was observed. According to all ROUGE measures, our model performed considerably better than the other copy-mechanism models. This was because the summary was created through the OOV copy mechanism along with the positive effects of vector expression of the morpheme-classified words according to the BERT pre-training that was reflected in the training effects of the Korean language model. ROUGE-1 is a performance index focusing on the emergence sequence of summary words, while ROUGE-L is a performance index for the occurrence of consecutive words. The ROUGE-1 performance improved by 8.10% (from 47.01 to 55.11), while ROUGE-L improved by 10.10% (from 29.55 to 39.65).

Table 7 shows the word statistics of the summarization results. We compared results with the human-generated correct answers (Fig. 9). In the case of the BERT+LSTM+MOOV copy-model, shaded words were OOV words. The summary was generated by copying the words through the pointing network. This meant that the OOV word reproduction performance of the summary was improved through BERT embedding and selective OOV copy method.

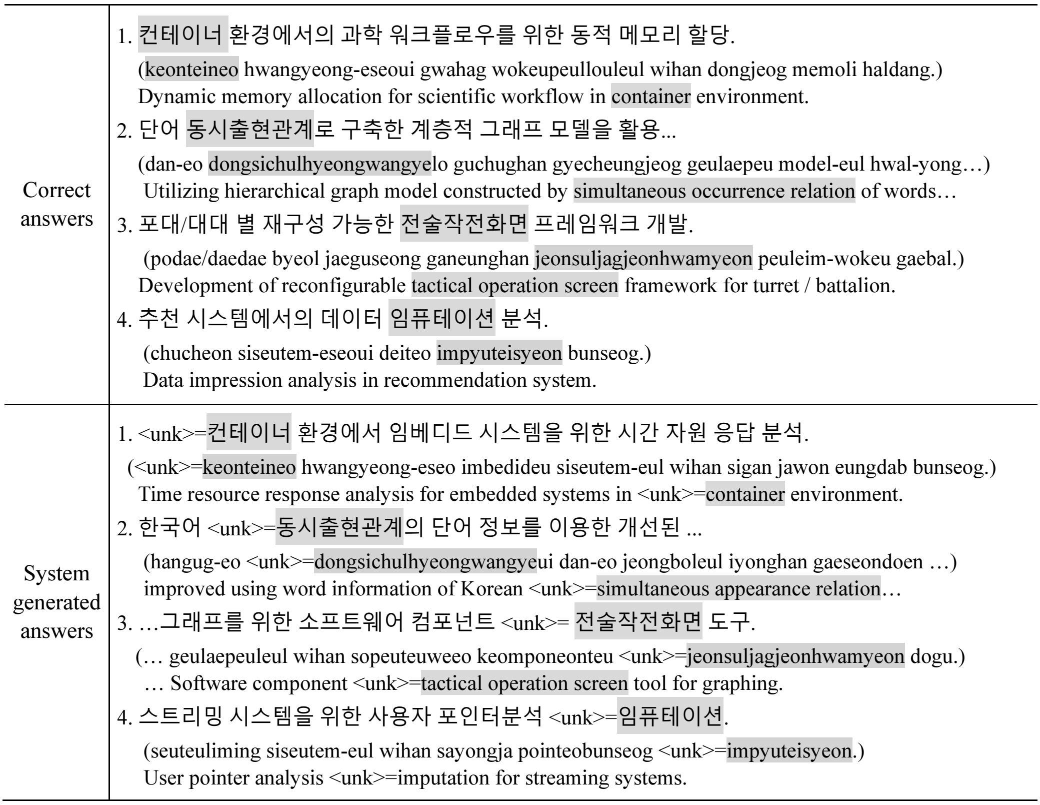
Table 6. Performance of our model and baseline

Model	ROUGE				
	L	1	2	SU4	h.mean
LSTM search	22.92	40.46	5.36	16.49	12.67
LSTM p_t feeding	25.37	43.60	5.46	16.48	13.07
LSTM $g_t + p_t$ feeding	29.55	47.01	5.95	17.80	14.32
LSTM $g_t + p_t$ +MOOV (9:1:0)	32.44	51.27	10.39	22.00	20.83
BERT + LSTM + $g_t + p_t$	34.60	51.45	8.43	21.30	18.70
BERT + LSTM + $g_t + p_t$ + MOOV (9:1:0)	39.65	55.11	11.49	23.53	23.14

MOOV (9:1:0)=selective masked-OOV copy model with substituting probabilities 9:1:0; h.mean=harmonic mean.

Table 7. Word statistics of summarization results

Statistic item	Value
Total number of pointed words	1,660
Total number of summary words	9,438
Percentage of pointed words (%)	17.59
Average number of pointed words per summary sentence	1.66



Correct answers

1. 컨테이너 환경에서의 과학 워크플로우를 위한 동적 메모리 할당.
(keonteineo hwangyeong-eseoui gwahag wokeupeullouleul wihan dongjeog memoli haldang.)
Dynamic memory allocation for scientific workflow in container environment.

2. 단어 동시출현관계로 구축한 계층적 그래프 모델을 활용...
(dan-eo dongsichulhyeongwangyelo guchughan gyecheungjeog geulaepeu model-eul hwal-yong…)
Utilizing hierarchical graph model constructed by simultaneous occurrence relation of words…

3. 포대/대대 별 재구성 가능한 전술작전화면 프레임워크 개발.
(podae/daedae byeol jaeguseong ganeunghan jeonsuljagjeonhwamyeon peuleim-wokeu gaebal.)
Development of reconfigurable tactical operation screen framework for turret / battalion.

4. 추천 시스템에서의 데이터 임퓨테이션 분석.
(chucheon siseutem-eseoui deiteo impyuteisyeon bunseog.)
Data impression analysis in recommendation system.

System generated answers

1. <unk>=컨테이너 환경에서 임베디드 시스템을 위한 시간 자원 응답 분석.
(<unk>=keonteineo hwangyeong-eseo imbedideu siseutem-eul wihan sigan jawon eungdab bunseog.)
Time resource response analysis for embedded systems in <unk>=container environment.

2. 한국어 <unk>=동시출현관계의 단어 정보를 이용한 개선된 ...
(hangug-eo <unk>=dongsichulhyeongwangyeui dan-eo jeongboleul iyonghan gaeseondoen …)
improved using word information of Korean <unk>=simultaneous appearance relation…

3. …그래프를 위한 소프트웨어 컴포넌트 <unk>= 전술작전화면 도구.
(… geulaepeuleul wihan sopeuteuweeo keomponeonteu <unk>=jeonsuljagjeonhwamyeon dogu.)
… Software component <unk>=tactical operation screen tool for graphing.

4. 스트리밍 시스템을 위한 사용자 포인터분석 <unk>=임퓨테이션.
(seuteuliming siseutem-eul wihan sayongja pointeobunseog <unk>=impyuteisyeon.)
User pointer analysis <unk>=imputation for streaming systems.

Fig. 9. Examples of abstractive summary generation. Shaded text indicates OOV words.

6. Conclusion

We improved the performance of abstractive summarization by training MOOV words using pre-trained embedding with position OOV words and selective OOV copy method. The OOV copy-pointing method in LSTM with BERT embedding improved the performance of document summarization. Experiments on academic papers demonstrated that summary performance was improved when MOOV and morpheme-to-sentence conversion were applied to the selectively-pointing OOV copy model. Specifically, we generated a summary by pointing to the OOV words, which were keywords selected by the authors, in the input document. In order to improve the quality of summarization, we trained a neural network gate that determined the pointing operation and the generation operation. ROUGE metric evaluation was increased because of the word-creation effects against the <unk> tags. Experimental results showed that ROUGE-1 score was enhanced from 40.46 to 55.11.

In a natural language system, OOV words make the system misunderstand semantic and syntactic information of text. So, they degrade the performance of the natural language system and cause misinterpretation of the meaning of the text. Our approach can be applied to dialogue-based systems to improve their performance and correctly understand the meaning of utterances. This work produces a summary sentence containing OOV words, but the quality of summarization can be improved through further research on generating more natural sentences.

Acknowledgement

This research was supported by the Ministry of Education of the Republic of Korean and the National Research Foundation of Korea (No. NRF-2019S1A5A2A03046571) and this research was supported by the Korea Institute of Science and Technology Information (No. K-21-L01-C06-S01). Our research on BERT embedding, selective-pointing mechanism, OOV masking, and deep learning methods contributes to solve the bias and fairness problem in AI systems.

References

[1] M. Allahyari, S. Pouriyeh, M. Assefi, S. Safaei, E. D. Trippe, J. B. Gutierrez, and K. Kochut, “Text summarization techniques: a brief survey,” 2017 [Online]. Available: https://arxiv.org/abs/1707.02268.

[2] N. Nazari and M. A. Mahdavi, “A survey on automatic text summarization,” *Journal of AI and Data Mining*, vol. 7, no. 1, pp. 121-135, 2019.

[3] S. Narayan, S. B. Cohen, and M. Lapata, “Ranking sentences for extractive summarization with reinforcement learning,” 2018 [Online]. Available: https://arxiv.org/abs/1802.08636.

[4] J. Cheng and M. Lapata, “Neural summarization by extracting sentences and words,” in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, Berlin, Germany, 2016, pp. 484-494.

[5] W. Wang, Y. Gao, H. Y. Huang, and Y. Zhou, “Concept pointer network for abstractive summarization,” 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)*, Hong Kong, China, 2019, pp. 3076-3085.

[6] G. Rossiello, P. Basile, and G. Semeraro, “Centroid-based text summarization through compositionality of word embeddings,” in *Proceedings of the MultiLing 2017 Workshop on Summarization and Summary Evaluation Across Source Types and Genres*, Valencia, Spain, 2017, pp. 12-21.

[7] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, “BERT: pre-training of deep bidirectional transformers for language understanding,” 2018 [Online]. Available: https://arxiv.org/abs/1810.04805.

[8] Y. Dong, “A survey on neural network-based summarization methods,” 2018 [Online]. Available: https://arxiv.org/abs/1804.04589.

[9] T. Shi, Y. Keneshloo, N. Ramakrishnan, and C. K. Reddy, “Neural abstractive text summarization with sequence-to-sequence models,” *ACM Transactions on Data Science*, vol. 2, no. 1, article no. 1, 2021. https://doi.org/10.1145/3419106

[10] M. Hu, Y. Peng, F. Wei, Z. Huang, D. Li, N. Yang, and M. Zhou, “Attention-guided answer distillation for machine reading comprehension,” 2018 [Online]. Available: https://arxiv.org/abs/1808.07644.

[11] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” *Advances in Neural Information Processing Systems*, vol. 27, pp. 3104-3112, 2014.

[12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” *Advances in Neural Information Processing Systems*, vol. 30, pp. 5998-6008, 2017.

[13] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” 2014 [Online]. Available: https://arxiv.org/abs/1409.0473.

[14] S. Xu, H. Li, P. Yuan, Y. Wu, X. He, and B. Zhou, “Self-attention guided copy mechanism for abstractive summarization,” in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, Virtual Event, 2020, pp. 1355-1362.

[15] J. Gu, Z. Lu, H. Li, and V. O. Li, “Incorporating copying mechanism in sequence-to-sequence learning,” in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, Berlin, Germany, 2016, pp. 1631-1640.

[16] C. Gulcehre, S. Ahn, R. Nallapati, B. Zhou, and Y. Bengio, “Pointing the unknown words,” in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, Berlin, Germany, 2016, pp. 140-149.

[17] R. Nallapati, B. Zhou, C. Gulcehre, and B. Xiang, “Abstractive text summarization using sequence-to-sequence RNNs and beyond,” 2016 [Online]. Available: https://arxiv.org/abs/1602.06023.

[18] K. Ganesan, “Rouge 2.0: updated and improved measures for evaluation of summarization tasks,” 2018 [Online]. Available: https://arxiv.org/abs/1803.01937.

[19] M. Yasunaga, J. Kasai, R. Zhang, A. R. Fabbri, I. Li, D. Friedman, and D. R. Radev, “ScisummNet: a large annotated corpus and content-impact models for scientific paper summarization with citation networks,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, Honolulu, HI, 2019, pp. 7386-7393.

[20] S. Erera, M. Shmueli-Scheuer, G. Feigenblat, O. P. Nakash, O. Boni, H. Roitman, et al., “A summarization system for scientific documents,” 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)*, Hong Kong, China, 2019, pp. 211-216.

[21] T. Lee and S. Kang, “Automatic text summarization based on selective OOV copy mechanism with BERT embedding,” *Journal of KIISE*, vol. 47, no. 1, pp. 36-44, 2020.

[22] L. N. Smith, “Cyclical learning rates for training neural networks,” in *Proceedings of 2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, Santa Rosa, CA, 2017, pp. 464-472.

[23] I. Loshchilov and F. Hutter, “SGDR: stochastic gradient descent with warm restarts,” 2016 [Online]. Available: https://arxiv.org/abs/1608.03983.

[24] J. Howard and S. Ruder, “Universal language model fine-tuning for text classification,” in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, Melbourne, Australia, 2018, pp. 328-339.

[25] G. Huang, Y. Li, G. Pleiss, Z. Liu, J. E. Hopcroft, and K. Q. Weinberger, “Snapshot ensembles: train 1, get m for free,” 2017 [Online]. Available: https://arxiv.org/abs/1704.00109.

[26] K. Goyal, G. Neubig, C. Dyer, and T. Berg-Kirkpatrick, “A continuous relaxation of beam search for end-to-end training of neural sequence models,” in *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, New Orleans, LA, 2018, pp. 3045-3052.

Tae-Seok Lee **https://orcid.org/0000-0003-2427-4092**

He received the M.S. degree in Computer Science from Korea Univ. in 2004 and Ph.D. degree in Computer Science from Kookmin Univ. in 2020. He is working in KISTI (Korea Institute of Science and Technology Information). His research interests are neural-network based language modeling, information retrieval, and information extraction.

Hyun-Young Lee **https://orcid.org/0000-0003-2553-6576**

He received the B.S. degree and M.S. degree in Computer Science from Kookmin University in 2016 and 2019 respectively. He is a Ph.D. candidate in Computer Science at Kookmin University since 2019. His research interests include natural language processing, machine learning, deep learning, and recommendation system.

Seung-Shik Kang **https://orcid.org/0000-0003-3318-6326**

He received B.S., M.S., and Ph.D. degrees in Computer Science from Seoul National University in 1986, 1988 and 1993, respectively. He is working for Kookmin University as a full professor. His research interests include natural language processing, text mining, big data processing, and machine learning.