Retrieval, Masking, and Generation: Feedback Comment Generation using Masked Comment Examples

Mana Ihori and **Hiroshi Sato** and **Tomohiro Tanaka** and **Ryo Masumura**
NTT Computer and Data Science Laboratories, NTT Corporation
1-1 Hikarinooka, Yokosuka-Shi, Kanagawa 239-0847, Japan
mana.ihori@ntt.com

Abstract

In this paper, we propose a novel method, retrieval, masking, and generation, for feedback comment generation. Feedback comment generation is a task in which a system generates feedback comments such as hints or explanatory notes for language learners, given input text and position showing where to comment. In the conventional study, the retrieve-and-edit method for retrieving feedback comments in the data pool and editing the comments has been thought effective for this task. However, the performance of this method does not perform as well as other conventional methods because its model learns to edit tokens that do not need to be rewritten in the retrieved comments. To mitigate this problem, we propose a method for combining retrieval, masking, and generation based on the retrieve-and-edit method. Specifically, tokens of feedback comments retrieved from the data pool are masked, and this masked feedback comment is used as a template to generate feedback comments. The proposed method should prevent unnecessary conversion by using not retrieved feedback comments directly but masking them. Our experiments on feedback comment generation demonstrate that the proposed method outperforms conventional methods.

1 Introduction

Feedback comment generation is a task in which, given an input text and position that shows where to comment, a system generates feedback comments such as hints or explanatory notes for non-native language learners (Nagata, 2019). In this task, it is not enough to simply point out or correct errors; the system should also explain why they are wrong. Such a system would be extremely beneficial for language learners, but there is currently no effective method for generating comments.

Conventionally, retrieval-based (Nagata, 2019), simple generation (See et al., 2017), and retrieve-and-edit (Hashimoto et al., 2018) methods have been used for the feedback comment generation task. With the retrieval-based method, the method retrieves a feedback comment from the data pool with maximum cosine similarity. Although the modeling for this method is simple, the output is not flexible; for example, this method often retrieves feedback comments that were correct as hints or explanatory notes but focused on tokens that are not in the input text. The simple generation method generates feedback comments directly given the input text and position using an encoder-decoder model. Since this method can generate flexible feedback comments, it mitigates the problem with the retrieval-based method. However, generating comments with simple generation is more difficult than with the retrieval-based method because this method should generate comments from scratch. Thus, Hanawa et al. (2021) used the retrieve-and-edit method that combines these two methods for better performance. With this method, the feedback comments retrieved with the retrieval-based method are edited using the simple generation method. However, the experimental results indicated that the retrieve-and-edit method did not perform well because its model learned excessive conversions and converted unnecessary tokens (Hanawa et al., 2021).

To mitigate this problem, we consider extending the retrieve-and-edit method with which unnecessary tokens in the retrieved feedback comments are actively edited. Our idea to specify where to edit the retrieved feedback comment is masking tokens that should be edited. By masking tokens that should be edited, the method should be able to predict only masked tokens and not change the other tokens in the retrieved feedback comments. Specifically, we mask tokens in retrieved feedback comments obtained with the retrieval-based method, and the method outputs a feedback comment given this masked feedback comment and the input text. We can create a template for comment generation, as the tokens of the retrieved feedback comment that are irrelevant to the input text are eliminated.

In this paper, we propose a novel method, *retrieval, masking, and generation*, for feedback comment generation. It consists of three modules: retrieval, masking, and generation. First, the retrieval module retrieves a feedback comment from the data pool, as with the conventional retrieval-based method. Next, the masking module executes binary classification, i.e., masking or not, for each token in the retrieved feedback comment. The masking module learns to mask tokens of the retrieved comment that are not in the reference feedback comment. Finally, the generation module generates a feedback comment given the input text and masked feedback comment. To generate feedback comments, the proposed method cascades the results of each module, which means that the performance of each module depends on the subsequent performance of the module output. To mitigate this potential problem, we introduce a multi-decoding operation that uses not only the top result but also the top k results. Our experiments on feedback comment generation demonstrate that the proposed method performs better than the above three conventional methods.

2 Feedback Comment Generation

In feedback comment generation, given the input text that has grammatical errors and position, a system outputs a feedback comment. We define the input text as $\boldsymbol{X} = \{x_1, \cdots, x_M\}$ and feedback comment as $\boldsymbol{Y} = \{y_1, \cdots, y_N\}$, where x_m and y_n are tokens, and M and N are the number of tokens in the input text and feedback comment, respectively. The position represents the range of the feedback target in character units and consists of integers. We make $\tilde{\boldsymbol{X}}$ emphasize the target characters by adding brackets to the input text on the basis based of the given position. For example, when $\boldsymbol{X}$ is “He agrees the opinion.” and the position is 3:13, $\tilde{\boldsymbol{X}}$ is “He «agrees the» opinion.” Note that in $\tilde{\boldsymbol{X}}$, the input text and position are not treated separately, but the position is also treated as a token.

3 Retrieval, Masking, and Generation Modules

The proposed method consists of three modules: retrieval, masking, and generation. Figure 1 shows an overview of the proposed method.

3.1 Retrieval Module

The retrieval module outputs a candidate of feedback comments, given the input text, position, and data pool of feedback comments. To obtain the candidate, the module calculates the cosine similarity between vectors of the input text $\boldsymbol{X}$ and feedback comment $\boldsymbol{Y}$ selected from the data pool. Since $\boldsymbol{X}$ does not include the position information, we add it by converting the position into position label $\boldsymbol{P} = \{p_1, \cdots, p_M\}$. The $p_m \in \{0, 1\}$ is the m-th binary label represented by 1 if each token in the input text is in the position range, and 0 otherwise. For example, when $\boldsymbol{X}$ is {He, agrees, the, opinion, .} and the position is 3:13, $\boldsymbol{P}$ is {0, 1, 1, 0, 0}. Therefore, given the $\boldsymbol{X}$, $\boldsymbol{P}$, and selected $\boldsymbol{Y}$, the module outputs the cosine similarity between $\boldsymbol{X}$ and $\boldsymbol{Y}$, as

$$s = \texttt{ret}(\boldsymbol{X}, \boldsymbol{P}, \boldsymbol{Y}; \boldsymbol{\Theta}_{\text{ret}}), \tag{1}$$

where $\texttt{ret}()$ is the function of the retrieval module and $\boldsymbol{\Theta}_{\text{ret}}$ is a trainable parameter set.

This module is constructed using a Transformer-based encoder model. First, the module encodes $\boldsymbol{X}$ and $\boldsymbol{Y}$ into hidden representations $\boldsymbol{Q} = \{\boldsymbol{q}_1, \cdots, \boldsymbol{q}_M\}$ and $\boldsymbol{R} = \{\boldsymbol{r}_1, \cdots, \boldsymbol{r}_N\}$ as

$$\boldsymbol{Q} = \texttt{TransformerEncoder}(\boldsymbol{X}; \boldsymbol{\Theta}_{\text{ret}}), \tag{2}$$

$$\boldsymbol{R} = \texttt{TransformerEncoder}(\boldsymbol{Y}; \boldsymbol{\Theta}_{\text{ret}}), \tag{3}$$

where $\texttt{TransformerEncoder}()$ is the Transformer encoder that consists of an embedding layer, scaled dot product multi-head self-attention layer, and position-wise feed-forward network (Vaswani et al., 2017). We also use a pre-trained BERT that performs well for various natural language understanding tasks (Devlin et al., 2018) as $\texttt{TransformerEncoder}()$ in the module.

Next, it multiplies $\boldsymbol{Q}$ and $\boldsymbol{P}$ to convert the hidden representations into a single vector $\boldsymbol{u}$ as

$$\boldsymbol{u} = \sum_{m=1}^{M} \boldsymbol{q}_m \cdot p_m. \tag{4}$$

We also use $\boldsymbol{r}_1$, which is the embedding of the [CLS] token at the beginning of the feedback comment as a single vector of $\boldsymbol{R}$.

Finally, we calculate the cosine similarity of $\boldsymbol{u}$ and $\boldsymbol{r}_1$ as

$$\texttt{ret}(\boldsymbol{X}, \boldsymbol{P}, \boldsymbol{Y}; \boldsymbol{\Theta}_{\text{ret}}) = \frac{\boldsymbol{u} \cdot \boldsymbol{r}_1}{\|\boldsymbol{u}\| \|\boldsymbol{r}_1\|}. \tag{5}$$

The module outputs a candidate of the feedback comment that has the highest similarity in the data pool.

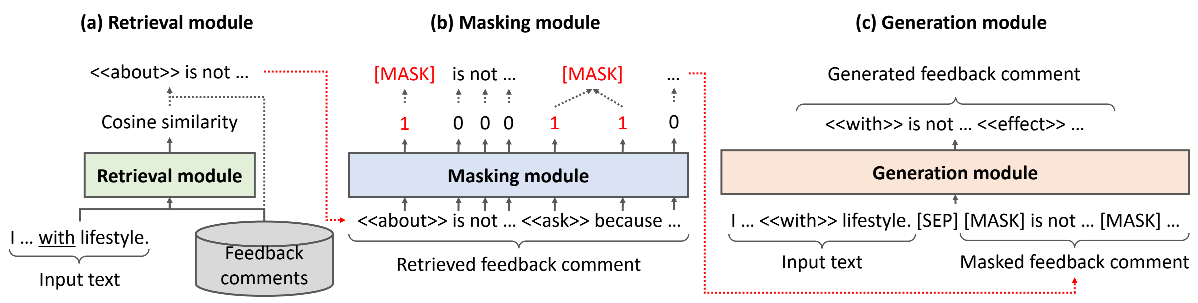

Figure 1: Overview of proposed method. Underlined words in input text represent position.

Training: The retrieval module is trained using dataset $\mathcal{D}_{\text{ret}} = \{(\boldsymbol{X}^1, \boldsymbol{P}^1, \mathcal{Y}, \mathcal{S}^1), \cdots, (\boldsymbol{X}^C, \boldsymbol{P}^C, \mathcal{Y}, \mathcal{S}^C)\}$. Here, C is the number of the input text and position, $\mathcal{Y} = \{\boldsymbol{Y}^1, \cdots, \boldsymbol{Y}^D\}$ is the data pool of feedback comments, $\mathcal{S}^c = \{s^{c,1}, \cdots, s^{c,D}\}$ is the sets of Levenshtein similarities (Levenshtein et al., 1966) for c-th reference feedback comment $\boldsymbol{Y}^c$, and $s^{c,d}$ is the Levenshtein similarity of $\boldsymbol{Y}^c$ and selected comment $\boldsymbol{Y}^d$. This module is trained to approach the cosine similarity of $\boldsymbol{X}^c$ and $\boldsymbol{Y}^d$ for $s^{c,d}$ using mean squared error. The training loss function $\mathcal{L}_{\text{ret}}$ is defined as

$$\mathcal{L}_{\text{ret}} = \frac{1}{C \cdot D} \sum_{c=1}^{C} \sum_{d=1}^{D} \left(s^{c,d} - \texttt{ret}(\boldsymbol{X}^c, \boldsymbol{P}^c, \boldsymbol{Y}^d; \boldsymbol{\Theta}_{\text{ret}})\right)^2, \quad (6)$$

Retrieval: The retrieval module outputs a candidate of feedback comment $\tilde{\boldsymbol{Y}}$ with the highest similarity per input text from the data pool $\mathcal{Y}$ as

$$\tilde{\boldsymbol{Y}} = \arg\max_{\boldsymbol{Y} \in \mathcal{Y}} \texttt{ret}(\boldsymbol{X}, \boldsymbol{P}, \boldsymbol{Y}; \boldsymbol{\Theta}_{\text{ret}}). \quad (7)$$

3.2 Masking Module

The masking module executes binary classification, i.e., masking or not, for each token in the retrieved feedback comment that is output of the retrieval module. Given the $\boldsymbol{X}$, $\boldsymbol{P}$, and $\tilde{\boldsymbol{Y}}$, the module outputs the masking label $\boldsymbol{L} = \{l_1, \cdots, l_N\}$ where $l_n \in \{0, 1\}$ is the n-the binary label as

$$P(\boldsymbol{L}|\boldsymbol{X}, \boldsymbol{P}, \tilde{\boldsymbol{Y}}; \boldsymbol{\Theta}_{\text{mask}}) = \prod_{n=1}^{N} P(l_n|\boldsymbol{X}, \boldsymbol{P}, \tilde{\boldsymbol{Y}}; \boldsymbol{\Theta}_{\text{mask}}), \quad (8)$$

where $\boldsymbol{\Theta}_{\text{mask}}$ is the trainable parameter set.

This module has a common architecture with the retrieval module. Thus, we convert $\boldsymbol{X}$ and $\boldsymbol{P}$ into a single vector $\boldsymbol{u}$ and convert $\boldsymbol{Y}$ into hidden representations $\boldsymbol{R} = \{\boldsymbol{r}_1, \cdots, \boldsymbol{r}_N\}$ with Eqs. (2), (3), and (4). Next, we compute a binary classification for each token in the retrieved comment by using these vectors as

$$P(l_n|\boldsymbol{X}, \boldsymbol{P}, \tilde{\boldsymbol{Y}}; \boldsymbol{\Theta}_{\text{mask}}) = \texttt{softmax}(\boldsymbol{v}_n; \boldsymbol{\Theta}_{\text{mask}}), \quad (9)$$

$$\boldsymbol{v}_n = [\boldsymbol{u}^{\mathrm{T}}, \boldsymbol{r}_n^{\mathrm{T}}]^{\mathrm{T}}, \quad (10)$$

where, $\texttt{softmax}()$ is a linear transformational function with a softmax activation.

The final output is a masked feedback comment with masked tokens labeled 1. The module masks tokens of the retrieved comment that are not in the reference feedback comment. It also replaces tokens that should be masked with the special token [MASK]. The generation module should then predict the same number of tokens as the number of masked tokens; however, the numbers of masked tokens and tokens that should be generated are not necessarily the same. Therefore, we use span-mask denoising (Raffel et al., 2020), which replaces consecutive tokens with a single special token.

Training: The masking module is trained using dataset $\mathcal{D}_{\text{mask}} = \{(\boldsymbol{X}^1, \boldsymbol{P}^1, \mathcal{Y}, \mathcal{L}^1), \cdots, (\boldsymbol{X}^C, \boldsymbol{P}^C, \mathcal{Y}, \mathcal{L}^C)\}$, where C is the number of input text and position, and $\mathcal{L}^c = \{\boldsymbol{L}^{c,1}, \cdots, \boldsymbol{L}^{c,D}\}$ is the label sets. Label set $\boldsymbol{L}^{c,d}$ is a set with label 1 for tokens of the $\boldsymbol{Y}^d$ selected from the data pool that are not in the reference feedback comment $\boldsymbol{Y}^c$ and 0 for all others. The training loss function $\mathcal{L}_{\text{mask}}$ is defined as

$$\mathcal{L}_{\text{mask}} = -\sum_{c=1}^{C} \sum_{d=1}^{D} \log P(\boldsymbol{L}^{c,d}|\boldsymbol{X}^c, \boldsymbol{P}^c, \boldsymbol{Y}^d; \boldsymbol{\Theta}_{\text{mask}}). \quad (11)$$

Masking: The decoding problem is defined as

$$\hat{\boldsymbol{L}} = \arg\max_{\boldsymbol{L}} P(\boldsymbol{L}|\boldsymbol{X}, \boldsymbol{P}, \tilde{\boldsymbol{Y}}; \boldsymbol{\Theta}_{\text{mask}}). \quad (12)$$

Finally, the module outputs a masked feedback comment $\bar{\boldsymbol{Y}}$ using $\hat{\boldsymbol{L}}$ and $\tilde{\boldsymbol{Y}}$ as

$$\bar{\boldsymbol{Y}} = \texttt{MASK}(\tilde{\boldsymbol{Y}}, \hat{\boldsymbol{L}}), \quad (13)$$

where MASK() is the operation to mask tokens of the retrieved feedback comment using span-mask denoising. Note that when the output results are all 0 (no tokens are masked), we directly use the retrieved comment as the feedback comment.

3.3 Generation Module

The generation module outputs $\boldsymbol{Y}$ given the input text $\tilde{\boldsymbol{X}} = \{\tilde{x}_1, \cdots, \tilde{x}_m\}$ and masked feedback comment $\bar{\boldsymbol{Y}} = \{\bar{y}_1, \cdots, \bar{y}_n\}$. The input is the concatenated sequence of the input text and masked comment with a separator token, $\boldsymbol{Z} = \{\tilde{x}_1, \cdots, \tilde{x}_m, [\text{SEP}], \bar{y}_1, \cdots, \bar{y}_n\}$. The generation probability of $\boldsymbol{Y}$ is defined as

$$P(\boldsymbol{Y}|\boldsymbol{Z}; \boldsymbol{\Theta}) = \prod_{n=1}^{N} P(y_n|y_{1:n-1}, \boldsymbol{Z}; \boldsymbol{\Theta}_{\text{gen}}), \quad (14)$$

where $\boldsymbol{\Theta}_{\text{gen}}$ is the trainable parameter set.

This module is constructed using a Transformer-based encoder-decoder model. First, the encoder converts the $\boldsymbol{Z}$ into hidden representations $\boldsymbol{H}$ as

$$\boldsymbol{H} = \texttt{TransformerEncoder}(\boldsymbol{Z}; \boldsymbol{\Theta}_{\text{gen}}). \quad (15)$$

Next, the decoder computes the generation probability of a token from the preceding tokens and the $\boldsymbol{H}$. The predicted probabilities of the n-th token y_n are calculated as

$$P(y_n|y_{1:n-1}, \boldsymbol{Z}; \boldsymbol{\Theta}_{\text{gen}}) = \texttt{softmax}(\boldsymbol{w}_n; \boldsymbol{\Theta}_{\text{gen}}). \quad (16)$$

The hidden representations $\boldsymbol{w}_n$ are calculated using $\boldsymbol{H}$ and $y_{1:n-1} = \{y_1, \cdots, y_{n-1}\}$ as

$$\boldsymbol{w}_n = \texttt{TransformerDecoder}(y_{1:n-1}, \boldsymbol{H}; \boldsymbol{\Theta}_{\text{gen}}), \quad (17)$$

where TransformerDecoder() is the Transformer decoder that consists of an embedding layer, scaled dot product multi-head self-attention and source target attention layers, and a position-wise feed-forward network (Vaswani et al., 2017).

Training: The generation module is trained using dataset $\mathcal{D}_{\text{gen}}=\{(\boldsymbol{Z}^1, \boldsymbol{Y}^1), \cdots, (\boldsymbol{Z}^{|\mathcal{D}_{\text{gen}}|}, \boldsymbol{Y}^{|\mathcal{D}_{\text{gen}}|})\}$. The training loss function $\mathcal{L}_{\text{gen}}$ is defined as

$$\mathcal{L}_{\text{gen}} = - \sum_{(\boldsymbol{Z}, \boldsymbol{Y}) \in \mathcal{D}_{\text{gen}}} \log P(\boldsymbol{Y}|\boldsymbol{Z}; \boldsymbol{\Theta}_{\text{gen}}). \quad (18)$$

Decoding: The decoding problem is defined as

$$\hat{\boldsymbol{Y}} = \arg\max_{\boldsymbol{Y}} P(\boldsymbol{Y}|\boldsymbol{Z}; \boldsymbol{\Theta}_{\text{gen}}). \quad (19)$$

3.4 Multi-Decoding

Since the proposed method cascades the output of three modules, the performance of each module is directly related to that of the next module. However, it is difficult to fully guarantee the output of each module. When only the top output is used, we might not be able to take full advantage of each module. To mitigate this problem, we use multi-decoding to generate a feedback comment with high confidence.

In the multi-decoding operation, the top k feedback comments per input text are first retrieved using the retrieval module. Next, masking is executed on each token of k retrieved comments using the masking module. Then, the duplicated ones for the same input text are excluded. When only the unmasked tokens remain, the comment is stored as a candidate feedback comment. Next, given each masked feedback comment and the input text, the generation module outputs the feedback comment per masked comment. As a result of this process, multiple feedback comments are generated for a single input text, including the candidates that are outputted in the masking module. Thus, we extract only one feedback comment from these comments using Algorithm 1. In this algorithm, lev() is the function to calculate Levenshtein similarity.

In the feedback comment generation task, the special output token <NO_COMMENT> indicates that a system cannot generate any reliable feedback comment. In this study, unreliable feedback comments are converted into <NO_COMMENT> in accordance with the following rules.

- Feedback comments in which bracketed tokens in the comment are not in the input text.
- Feedback comments include " the preposition is not necessary," but the input text does not have the preposition.

Algorithm 1 Multi-decoding operation

```
Require: $\boldsymbol{X}, \texttt{candidates} = \{\hat{\boldsymbol{Y}}^1, \cdots, \hat{\boldsymbol{Y}}^k\}$
 1: $\texttt{all\_ave} = 0, \texttt{all\_cnt} = 0$
 2: for $i \leftarrow 0$ to $k$ do
 3:     $\texttt{ave} = 0, \texttt{cnt} = 0$
 4:     for $j \leftarrow 0$ to $k$ do
 5:         if $i \neq j$ then
 6:             $\texttt{ave} = \texttt{ave} + \texttt{lev}(\hat{\boldsymbol{Y}}^i, \hat{\boldsymbol{Y}}^j)$
 7:             if $\hat{\boldsymbol{Y}}^i = \hat{\boldsymbol{Y}}^j$ then
 8:                 $\texttt{cnt} = \texttt{cnt} + 1$
 9:             end if
10:         end if
11:     end for
12:     $\texttt{ave} = \texttt{ave} + \texttt{lev}(\hat{\boldsymbol{Y}}^i, \boldsymbol{X})$
13:     $\texttt{ave} = \texttt{ave}/k$
14:     if $\texttt{all\_cnt} < \texttt{cnt}$ then
15:         $\texttt{all\_cnt} = \texttt{cnt}$
16:         $\boldsymbol{Y}_c = \hat{\boldsymbol{Y}}^i$
17:         $\texttt{flag} = 0$
18:     else if $\texttt{cnt} = \texttt{all\_cnt}$ then
19:         $\texttt{flag} = 1$
20:     end if
21:     if $\texttt{all\_ave} < \texttt{ave}$ then
22:         $\texttt{all\_ave} = \texttt{ave}$
23:         $\boldsymbol{Y}_a = \hat{\boldsymbol{Y}}^i$
24:     end if
25: end for
26: if $\texttt{flag} = 1$ then
27:     $\boldsymbol{Y} = \boldsymbol{Y}_a$
28: else
29:     $\boldsymbol{Y} = \boldsymbol{Y}_c$
30: end if
31: return $\boldsymbol{Y}$
```

4 Experiments

4.1 Datasets

We used a dataset provided by Generation Challenge 2022 that contains input text, position, and feedback comments. The dataset has 4,868 sentences in a training set, 170 sentences in a validation set, and 215 sentences in a test set. In the dataset, the errors in the input text only cover preposition uses. The three modules of the proposed method require individual datasets for training; thus, we created datasets for each module from this provided dataset.

Retrieval module: The dataset for the retrieval module consists of the input text, position, feedback comment selected from the data pool, and Levenshtein similarity. The input text and position are the same as the provided dataset. Also, the Levenshtein similarity is calculated using reference feedback comments and selected comments from the data pool (feedback comments in the training data of the provided dataset). In the provided dataset, there are many low-similarity combinations of the reference feedback comment and selected comments. Thus, the dataset would be unbalanced if we used all combinations for training. To prevent this problem, we randomly removed samples so there would be less than five with the same first decimal place value of similarity. Finally, we divided the dataset into 139,687 sentences in a training set and 5,001 sentences in a validation set.

Masking module: The dataset for the masking module consists of the input text, position, feedback comments selected from the data pool, and masking labels. We use the sets of input text, position, and selected comments that were created in the retrieval module for the masking module to prevent data imbalance. To make masking labels, we took a word-by-word alignment for all comments and reference feedback comments. We then labeled tokens that were not in the reference feedback comments as 1 and others as 0 for the selected comments. Finally, there were 139,687 sentences in a training set and 5,001 sentences in a validation set, the same as the dataset of the retrieval module.

Generation module: The dataset for the generation module consists of the sequence that was concatenated with the input text and masked feedback comments, and reference feedback comments. The input text and masked feedback comment are connected using a separator token [SEP]. To create the dataset, we used the datasets of the retrieval and masking modules. First, we extracted the top five and five random feedback comments per input text by using the Levenshtein similarity in the dataset of the retrieval module. Next, we extracted the masked feedback comments that correspond to the above ten feedback comments from the dataset of the masking module. Then, we deleted the duplicated masked comments. Finally, these input text and masked feedback comments were concatenated into a single sequence and paired with the reference comment. In addition, we divided the dataset into 48,309 sentences in a training set and 1,000 sentences in a validation set.

4.2 Setup

We implemented the proposed method (with all three modules); the retrieval-and-generation method, which is the proposed method without the masking module and regarded as the retrieve-and-edit method; the retrieval module only, regarded as the retrieval-based method; and the generation module only, regarded as the simple generation method. We also used the pointer-generator network (See et al., 2017) provided by Generation Challenge 2022 as a baseline for comparison. We converted the comments generated with all methods into <NO_COMMENT> when they met the rules discussed in Subsection 3.4.

These methods were fine-tuned using a pre-trained model. The retrieval and masking modules used a pre-trained BERT (Devlin et al., 2018) (bert-based-cased from the HuggingFace Transformers library (Wolf et al., 2020)). The generation module used a pre-trained T5 (Raffel et al., 2020) (t5-base from the HuggingFace Transformers library (Wolf et al., 2020)). We fine-tuned these pre-trained models using the dataset constructed in Subsection 4.1 and used the RAdam optimizer (Liu et al., 2019) with the mini-batch size set to 64. In the multi-decoding operation, we set k to seven. We also fine-tuned only the generation module using the provided dataset and the retrieval and generation method using the dataset that had unmasked feedback comments in the dataset for the generation module. Note that only the pointer-generator network was not pre-trained.

4.3 Results

Table 1 lists the experimental results of the feedback comment generation. The values represent BLEU scores (Papineni et al., 2002), where precision is calculated by dividing the sum of BLEU for each generation by the number of expected feedback comments, and recall is calculated by dividing the sum of BLEU for each generation by the number of generations excluding <NO_COMMENT>. The precision and recall results are then used to calculate F1.

The table shows that the proposed method with multi-decoding outperformed the other methods. Specifically, the performance of the proposed method improved using multi-decoding. With the proposed method, we believe that the multi-decoding improved the probability of generating a feedback comment that was close to the correct comment because the method could generate different feedback comments by using different masked comments. It generated different feedback comments for the same input text using different masked comments, as shown in Table 2. The table shows that these comments were generated by predicting mask tokens of masked feedback comments. These results indicate that masking tokens in the retrieved feedback comments are important for rewriting these comments for the input text.

Method	Precision	Recall	F1
pointe-generator	0.334	0.334	0.334
retrieval	0.424	0.422	0.423
generation	0.464	0.464	0.464
retrieval-and-generation	0.482	0.482	0.482
+ multi-decoding	0.480	0.480	0.480
proposed	0.483	0.481	0.482
+ multi-decoding*	**0.495**	**0.493**	**0.494**

* This is our best result, although it differs from officially published results.

Table 1: Results of feedback comment generation.

With the retrieval-and-generation method, when we used different retrieved feedback comments for the same input text, it generated the same feedback comment, as shown in Table 3. Therefore, even if we used multi-decoding for this method, the performance would almost be the same without multi-decoding. We assume it would be difficult to rewrite the retrieved feedback comment for the input text with this method. It also performed better than the generation module. This indicates that a large amount of training data was effective, not the use of retrieved feedback comments without masking.

The retrieval module underperformed the other methods. This is because it often retrieved comments that were correct as feedback comments but focused on tokens that were not in the input text, as was the problem with the conventional study.

4.4 Ablation Study

Table 1 shows that the performances of the proposed method and retrieval-and-generation method were equivalent. We believe that the performance of the masking module adversely affected the performance of the proposed method because the proposed method cascades the results of three modules. Thus, at the inference, we compared the final results using predicted masking with correct masking. Table 4 shows the results, and the proposed method

Input text	Reference	
After all , as a student , he or she needs to put the study at the first place .	«At» is not the correct <preposition> to be used with the set phrase formed using «the first place» meaning "to prioritize something". 'In' is the <preposition> to be used with the <noun> «place».	
Retrieved	**Masked**	**Generated**
The <preposition> «at» is normally used to indicate a relatively short period of time such as the time of day. Look up the <noun> «time» in a dictionary to learn the appropriate <preposition> to be used to indicate a period.	The <preposition> «at» is normally used to indicate [MASK] relatively short period [MASK] time [MASK] the time [MASK] day. Look up the <noun> **«time»** in a dictionary to learn the [MASK] <preposition> to [MASK] used to indicate [MASK] **period**.	The <preposition> «at» is normally used to indicate a relatively short period of time such as the time of day. Look up the <noun> «place» in a dictionary to learn the appropriate <preposition> to be used to indicate a place.
Using the <preposition> «in» makes the expression literally mean "to have in one hand". Look up the <noun> «hand» in a dictionary to learn the appropriate <preposition> to be used to form an <idiom> introducing one of two ideas.	[MASK] the <preposition> [MASK] **the expression** [MASK] Look up the <noun> [MASK] in a dictionary to learn the [MASK] <preposition> to [MASK] used **to** [MASK]	«At» is not the appropriate <preposition> to be used with the <noun> «place» to express "to put something in the first place". Look up the <noun> «place» in a dictionary to learn the appropriate <preposition> to be used.

Table 2: Example of proposed method's output. <u>Underlined words</u> in input text represent position. **Bold words** in masked comments mean that they were edited but not masked. Red words in generated comments mean that they were not in masked comments.

Input text	Reference
... colleagues is totally different with the way ...	The <preposition> «with» is often used to indicate concordance. Consult a ...
Retrieved	**Generation**
«In» is not the <preposition> used with 'bad' to qualify the subsequent ...	The <preposition> «with» is often used to indicate concordance. Consult a ...
«to» is not the correct <preposition> to be used to refer to the target. Look ...	The <preposition> «with» is often used to indicate concordance. Consult a ...

Table 3: Example of retrieval-and-generation method's output. <u>Underlined words</u> in input text represent position.

Method	Precision	Recall	F1
proposed	0.483	0.481	0.482
with correct masking	**0.539**	**0.539**	**0.539**

Table 4: Results using predicted or correct masking.

with correct masking was significantly improved in performance. Table 5 also shows the generated feedback comments using predicted and correct masked feedback comments. The predicted masking was wrong. Although the generated feedback comment using the correct masked comment was the same as the reference feedback comment, using the predicted masked comment was different. In addition, when we used predicted masking, the proposed method also edited tokens other than those in the masked comments. We assume its model determined that predicting only the masked tokens would generate unnatural feedback comments. In other words, when the wrong masked feedback comment was used, the generation task was more difficult than using correct masking. Therefore, it is inferred that the design of the masking module is important for the proposed method, and we should improve this for future work.

5 Conclusion

In this paper, we proposed a novel method, *retrieval, masking, and generation*, for feedback comment generation. The proposed method has three modules, retrieval, masking, and generation, and generates feedback comments by cascading each module output. First, the retrieval module extracts an example of feedback comments appropriate for the input text from the data pool. Next, the masking module masks tokens of the retrieved example that are not in the reference feedback comment. Finally, given a concatenated sequence of the input text and masked feedback comment, the generation module outputs the feedback comment in the generation module. With this three-module method, the performance of each module depends on the subsequent performance of the module output. Since it is difficult to guarantee the perfect performance of each module, we use a multi-decoding operation that uses not only the top result but also the top k results for each decoding. Our experimental results indicate that the proposed method outperforms the three conventional methods.

Input text	Reference	Retrieved
... please keep consider about that.	The <verb> «consider» is a <transitive verb> when used to express "to think about" and does not need the <preposition> «about».	Since the <verb> encounter is a <transitive verb> when used to express "to meet someone", a <preposition> does not need to precede the <object>.
	Masked	Generation
Predicted Masking	[MASK] the <verb> [MASK] is <transitive verb> **when used to** [MASK] a <preposition> **does not need to precede the** [MASK]	When the <verb> «consider» is used as a <transitive verb> in a sense of "care for someone", a <preposition> is unnecessary.
Correct Masking	[MASK] <verb> [MASK] is a <transitive verb> when used to express "to [MASK] does not need [MASK]	The <verb> «consider» is a <transitive verb> when used to express "to think about" and does not need the <preposition> «about».

Table 5: Example of proposed method using predicted or correct masking. Underlined words in input text represent position. **Bold words** in masked comments mean that they were edited but not masked. Red words in generated comments mean that they were not in masked comments.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Kazuaki Hanawa, Ryo Nagata, and Kentaro Inui. 2021. Exploring methods for generating feedback comments for writing learning. In *Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9719–9730.

Tatsunori B Hashimoto, Kelvin Guu, Yonatan Oren, and Percy S Liang. 2018. A retrieve-and-edit framework for predicting structured outputs. In *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, page 10052–10062.

Vladimir I Levenshtein et al. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, pages 707–710.

Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. 2019. On the variance of the adaptive learning rate and beyond. In *Proc. International Conference on Learning Representations (ICLR)*.

Ryo Nagata. 2019. Toward a task of feedback comment generation for writing learning. In *Proc. Conference on Empirical Methods in Natural Language Processing and International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3206–3215.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proc. the Association for Computational Linguistics (ACL)*, pages 311–318.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proc. Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1073–1083.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proc. Advances in neural information processing systems (NIPS)*, pages 5998–6008.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proc. Conference on Empirical Methods in Natural Language Processing: System Demonstrations (EMNLP)*, pages 38–45.