Exploring Methods for Generating Feedback Comments for Writing Learning

Kazuaki Hanawa[1,2], **Ryo Nagata**[3,4], **Kentaro Inui**[2,1]
[1]RIKEN Center for Advanced Intelligence Project, [2]Tohoku University
[3]Konan University, [4]JST, PRESTO
kazuaki.hanawa@riken.jp
nagata-emnlp2021 @ ml.hyogo-u.ac.jp.
inui@ecei.tohoku.ac.jp

Abstract

The task of generating explanatory notes for language learners is known as feedback comment generation. Although various generation techniques are available, little is known about which methods are appropriate for this task. Nagata (2019) demonstrates the effectiveness of neural-retrieval-based methods in generating feedback comments for preposition use. Retrieval-based methods have limitations in that they can only output feedback comments existing in a given training data. Furthermore, feedback comments can be made on other grammatical and writing items than preposition use, which is still unaddressed. To shed light on these points, we investigate a wider range of methods for generating many feedback comments in this study. Our close analysis of the type of task leads us to investigate three different architectures for comment generation: (i) a neural-retrieval-based method as a baseline, (ii) a pointer-generator-based generation method as a neural seq2seq method, (iii) a retrieve-and-edit method, a hybrid of (i) and (ii). Intuitively, the pointer-generator should outperform neural-retrieval, and retrieve-and-edit should perform best. However, in our experiments, this expectation is completely overturned. We closely analyze the results to reveal the major causes of these counter-intuitive results and report on our findings from the experiments. [1]

1 Introduction

Feedback comment generation is the task of generating explanatory notes for writing learning. An example of a feedback comment would be:

(1) Target sentence: **We discussed about it.*
Feedback comment: Since *discuss* is a transitive verb, *about* is not required.[2]

This type of feedback can assist the writer in determining why their writing was wrong and how to fix it.

While datasets (Nagata, 2019; Nagata et al., 2020; Pilan et al., 2020) for this task have become available, very little is known about the feedback comment generation methods. Nagata (2019) demonstrates deep neural network (DNN)-based retrieval methods are effective in generating feedback comments for preposition use, which is almost the only knowledge available. Recently, a wide variety of DNN-based generation methods are available and they can improve feedback comment generation performance significantly. Furthermore, Nagata (2019) focused on feedback comments for preposition use. DNN-based retrieval methods might not perform well on more general feedback comments (Nagata et al., 2020; Pilan et al., 2020) because they involve a wide range of writing items and rules. More sophisticated methods will likely perform better in such cases.

Given these contexts, we investigate feedback comment generation methods for preposition and general feedback comments in this study to demonstrate promising steps in this task. The first problem we must address is deciding which methods to investigate because several methods are available as mentioned above. Following the findings, a retrieval-based method (which, will be referred to as RETRIEVAL-BASED, hereafter) is naturally chosen as a baseline method (Nagata, 2019). However, RETRIEVAL-BASED is inflexible in that it can only output feedback comments existing in a given training data. DNN-based generation (i.e., sequence-to-sequence) methods demonstrate more flexible generation. We choose the Pointer Generator Network (See et al., 2017) as a simple DNN-based generation representative, which will be referred to as SIMPLE GENERATION, hereafter. It is an encoder-decoder neural network with attention and copy mechanisms. It is preferable to have a copy

[1]Our source code is available at https://github.com/k-hanawa/fcg_emnlp2021

[2]Note that feedback comments are actually written in Japanese but we show English translation for clarity in this paper.

Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 9719–9730
November 7–11, 2021.

mechanism because feedback comments often cite words from their target text. Furthermore, a hybrid of these two methods that edit retrieved examples by the pointer generator network, which will be referred to as RETRIEVE-AND-EDIT, hereafter. RETRIEVE-AND-EDIT, which can benefit from retrieval and DNN language generation, is expected to perform better than generating feedback comments from scratch. To summarize, theoretically, performance order is expected to be RETRIEVAL-BASED < SIMPLE GENERATION < RETRIEVE-AND-EDIT.

However, our experiments show completely different performance orders in preposition and general feedback comments: RETRIEVE-AND-EDIT < RETRIEVAL-BASED < SIMPLE GENERATION for the former whereas SIMPLE GENERATION < RETRIEVE-AND-EDIT < RETRIEVAL-BASED for the latter.

We investigate the generation results to reveal the following four findings:

- RETRIEVE-AND-EDIT frequently makes unnecessary edits (we will call this phenomenon *over-editing*). The over-editing problem leads to performance degradation that outweighs the advantage of flexible generation.
- SIMPLE GENERATION outperforms others in terms of preposition feedback comment generation but generates more *mixed* feedback comments than the other two in terms of general feedback comment generation, for reasons we will discuss in Section 6.2.
- For these reasons, RETRIEVAL-BASED performs the best in general feedback comments, which implies that significant progress is required to improve in general feedback comments by DNN-based generation methods.
- SIMPLE GENERATION and RETRIEVE-AND-EDIT generate more feedback comments that may mislead learners than RETRIEVAL-BASED because of their flexibility.

Based on these analyses, we reach the following conclusion.

- SIMPLE GENERATION performs best in a setting with few variations of feedback comments such as preposition feedback comments.
- RETRIEVE-AND-EDIT is considered to be promising for general feedback comment generation, but the over-editing problem prevents it from performing properly.
- It will be necessary to develop a system that makes it easier for users to select results, such as confidence estimation.

2 Related Work

There has been little study on feedback comment generation. Some researchers (Mccoy and Pennington, 1996; Kakegawa et al., 2000; Nagata et al., 2014) attempted to develop rule-based methods for diagnosing errors in line with grammatical error detection/correction. Typically, rule-based methods parse input sentences and then apply rules to the resulting parse to diagnose errors. However, they encounter enormous difficulties in dealing with numerous errors and maintaining a broad set of rules. Lai and Chang (2019) proposed a template-based method that identifies error types and generates comments based on them.

Recently, datasets for research in feedback comment generation have become available. Nagata (2019) proposed a feedback comment generation task with a dataset. Nagata et al. (2020) extended the dataset to feedback comments in preposition use and in general. Pilan et al. (2020) published a dataset that included feedback comments on linking word usage with error tags and learners' revisions.

Although datasets have become available, we know little about DNN-based methods for feedback comment generation, which have been proven to be effective in various natural language generation tasks (Shang et al., 2015; Rush et al., 2015; Bahdanau et al., 2016; Yuan and Briscoe, 2016). Nagata (2019) demonstrates that a neural-retrieval-based method performs well although they can only generate feedback comments existing in the training data. This implies that retrieve-and-edit-based methods, such as those proposed by Hashimoto et al. (2018), which are a natural extension of retrieval-based methods, are likely to be effective in this task.

Several variations on the retrieve-and-edit approach have been proposed. Weston et al. (2018); Guu et al. (2018); Cao et al. (2018); Hossain et al. (2020) retrieve a similar instance using a superficial similarity of the entire input. In the feedback comment generation task, we must retrieve an instance of similar errors, and it is inappropriate to use superficial similarity because sentences with the same error are often superficially different. Hashimoto et al. (2018) proposed a method for

training a retriever to embed the input sentence in a task-specific manner. We use this model because we should retrieve instances based on the semantic similarity of the errors rather than the superficial similarity.

3 Feedback Comment Generation: What Kind of Task Is It?

3.1 Task Definition

We define the feedback comment generation task based on the one proposed by Nagata (2019). The task is to generate text that helps the writer (learner) improve their writing skill given a sentence and an error position (where to comment), which we will refer to as an *offset* hereafter. Generally, it is a remark about a grammatical error, but it can also be about other things including discourse, organization, and content. In summary, the input is an English sentence and an offset. The output is a feedback comment corresponding to its offset.

3.2 Task Properties and Promising Approaches

The feedback comment generation task has properties similar to a classifier problem; there are multiple instances with the same error type (i.e., class). Take the following as examples.

(2) Target sentence: **We reached to the station.*
Feedback comment: Since the verb *reach* is a transitive verb, the preposition *to* is not required.

(3) Target sentence: **I reached to New York.*
Feedback comment: *reach* is a transitive verb. It does not need a preposition before the object.

Both examples belong to the same error type (using a preposition after a transitive verb). These comments are superficially different, but their contents are similar and interchangeable.

Given these properties, a promising approach is to solve it as a classification problem according to error types. However, the error types are not specified explicitly. Then, the RETRIEVAL-BASED, which retrieves instances to the input, would be useful because it does not require an explicit definition of error types.

However, retrieving similar instances is insufficient. Consider the following example.

(4) Target sentence: **I approach to the goal.*
Feedback comment: *approach* is a transitive verb. It does not need a preposition before the object.

Example (4) falls into the same error type as Examples (2) and (3), but Example (4) uses the verb "approach" instead of "reach." Therefore, Example (3) is incorrect as feedback comment for Example (4). To deal with this inflexibility of RETRIEVAL-BASED, SIMPLE GENERATION, which is a DNN-based generation method, is considered more promising. As shown in the previous examples, words in the input sentence often appear in the feedback comment (e.g., *approach* and *to* in Example (4)). Therefore, it is desirable for feedback comment generation methods to be capable of copying words from an input text to its prediction. Therefore, we use a pointer-generator network (See et al., 2017) as our encoder-decoder.

Furthermore, RETRIEVE-AND-EDIT, which edits retrieved examples, is considered most promising. In Example (3), replacing *reach* with *approach* makes it a correct feedback comment. As in this example, it is expected that editing the retrieval example will be easier than generating a feedback comment from scratch.

In summary, this study compares the three methods: RETRIEVAL-BASED, SIMPLE GENERATION, and RETRIEVE-AND-EDIT. The next section describes these three methods in detail.

4 Methods to Explore

In this section, we describe the three feedback comment generation methods in detail. RETRIEVAL-BASED uses an architecture similar to that of SIMPLE GENERATION as a retriever. Therefore, we explain SIMPLE GENERATION, RETRIEVAL-BASED, and RETRIEVE-AND-EDIT in this order. In Appendix A, we show the diagrams of these three methods.

4.1 Notation

We will use the following notation throughout this study. We will denote a target English sentence, its length (the number of words), and i-th word by S, N, and w_i, respectively. Namely, $S = w_1, \ldots, w_N$. We will denote an offset by o. We will also denote the input and output by $\boldsymbol{x}$ and $\boldsymbol{y}$, respectively. Note that $\boldsymbol{x}$ consists of a pair of S and o and that $\boldsymbol{y}$ corresponds to its feedback comment (consisting of

a sequence of words). For example, the underlined error:

(5) Target sentence: **We approached to the goal.*
Feedback comment: Since the verb *approach* is a transitive verb, the preposition *to* is not required.

would give $S =$ "We approached $\cdots$ goal ." with $N = 6$ and $o = 3$.

Furthermore, the symbols $\boldsymbol{x}'$ and $\boldsymbol{y}'$ will be used to denote a retrieved instance (a sentence and an offset) and its feedback comment, respectively. These symbols will be used in RETRIEVAL-BASED and RETRIEVE-AND-EDIT. Furthermore, we will use $\hat{\boldsymbol{y}}$ to denote a generated (predicted) feedback comment.

The hidden vector of the word w_i will be denoted by $\boldsymbol{h}_i$, which is typically obtained by an encoder from S and position i. The context vector of $\boldsymbol{x}$ will be denoted by $\boldsymbol{c}$.

4.2 SIMPLE GENERATION

We use a standard encoder-decoder model In a SIMPLE GENERATION, which has been proven to be effective in various natural language generation tasks including translation (Sutskever et al., 2014; Bahdanau et al., 2016; Wu et al., 2016) and grammatical error correction (Yuan and Briscoe, 2016; Napoles and Callison-Burch, 2017; Junczys-Dowmunt et al., 2018). It generates a feedback comment by (i) encoding the input $\boldsymbol{x}$ with its context into $\boldsymbol{c}(= \boldsymbol{h}_o)$ and (ii) predicting the feedback comment $\hat{\boldsymbol{y}}$ from $\boldsymbol{c}$ with attention and copy mechanisms.

As discussed in Section 3, we use the pointer-generator network (See et al., 2017) as our encoder-decoder model. It generates words through the generator while also copying words from the source sentence; for each decoder time step, it controls generation and copying based on the values of the *generation probability* p_{gen} and the copying probability, which is simply the attention distribution.

In our task, it is necessary to include the information about the offset o in context vector $\boldsymbol{c}$. Specifically, a Bi-LSTM encodes each word w_i in the target sentence S into $\boldsymbol{h}_i$ by a Bi-LSTM. Then, the o-th state $\boldsymbol{h}_o$ is chosen as the context vector $\boldsymbol{c}$. It is set as the initial state of the decoder, which is another Bi-LSTM. Therefore, SIMPLE GENERATION learns to obtain $\boldsymbol{c}$ from $\boldsymbol{x}$, as well as predict $\boldsymbol{y}$ from $\boldsymbol{c}$.

Training of the network is done in a standard manner. Namely, the pointer generator-network is trained to minimize the cross-entropy loss.

4.3 RETRIEVAL-BASED

RETRIEVAL-BASED first retrieves the most similar instance $\boldsymbol{x}'$ to $\boldsymbol{x}$ from the training data with its corresponding feedback comment $\boldsymbol{y}'$. It then outputs $\boldsymbol{y}'$ as its prediction (i.e., $\hat{\boldsymbol{y}}$). The cosine similarity between the corresponding context vectors $\boldsymbol{c}'$ and $\boldsymbol{c}$ is used to measure the similarity between the two instances $\boldsymbol{x}'$ and $\boldsymbol{x}$. To encode $\boldsymbol{x}$ to $\boldsymbol{c}$, which acts as a retriever in this method, nearly the same network architecture as in SIMPLE GENERATION is used. The only difference is that this network lacks the attention mechanism so all information about x is encoded into $\boldsymbol{c}$.

Using this network, all instances in the training data are converted into context vectors in the same manner as described in Subsection 4.2. Note that in the training phase, $\boldsymbol{x}$ and $\boldsymbol{y}$ are provided, and thus they can be used to obtain their context vectors. However, in the test phase, only $\boldsymbol{x}$ is available, and the context vector $\boldsymbol{c}$ is obtained through generation (prediction) of $\hat{\boldsymbol{y}}$.

4.4 RETRIEVE-AND-EDIT

RETRIEVE-AND-EDIT follows the method proposed by Hashimoto et al. (2018). It includes SIMPLE GENERATION and RETRIEVE-AND-EDITThe two components are implemented by two independent networks called *retriever* and *editor*. The same network as RETRIEVAL-BASED is used as a retriever. Additionally, nearly the same architecture as in SIMPLE GENERATION is adopted as the editor, which is the pointer-generator network.[3] One significant difference is that the network considers retrieved instances. Because $\boldsymbol{x}'$ and $\boldsymbol{y}'$, which are obtained by the retriever, are available together with the target sentence $\boldsymbol{x}$, the network accepts the triple as its input. This is simply done by (i) encoding $\boldsymbol{x}'$, $\boldsymbol{y}'$, and $\boldsymbol{x}$ into three context vectors and (ii) concatenating the three vectors into one, and (iii) setting it to the initial of the decoder (of the pointer-generator). To be precise, $\boldsymbol{x}$, $\boldsymbol{x}'$ and $\boldsymbol{y}'$ are encoded into three context vectors by three different Bi-LSTMs; $\boldsymbol{x}'$ and $\boldsymbol{x}$ are encoded into $\boldsymbol{c}'$ and $\boldsymbol{c}$, respectively, in the same manner as SIMPLE

[3] There are several possible architectures for the editor, but we found that the one using the pointer-generator network had the best performance in a pilot study. Therefore, we use it in this paper.

GENERATION. However, the context vector of the retrieved feedback comment $\boldsymbol{y}'$ is the final state of its Bi-LSTM.

Note that the editor has one attention mechanism. This means that attention is calculated over $\boldsymbol{x}$, $\boldsymbol{x}'$, and $\boldsymbol{y}'$ simultaneously to give an attention weight to each word of $\boldsymbol{x}$, $\boldsymbol{x}'$, and $\boldsymbol{y}'$. Therefore, the editor can generate a feedback comment considering $\boldsymbol{x}$, $\boldsymbol{x}'$, and $\boldsymbol{y}'$ and can copy any word anywhere from the three.

The two networks (retriever and editor) are trained independently, in the same manner described above. During training, the retriever extracts the most similar instance from the training data, excluding itself.

5 Performance Evaluation

5.1 Settings

We used the dataset (Nagata et al., 2020)[4] with additional data. It consisted of learner essays manually annotated with feedback comments in general (GENERAL) and those on preposition use (PREP). The essays were excerpts from ICNALE (Ishikawa, 2013), with essay topics were either (a) *It is important for college students to have a part-time job.* or (b) *Smoking should be completely banned at all the restaurants in the country.*, which hereafter will be referred to as *PTJ* and *SMK*, respectively. The number of data for training, development, and testing is shown in Table 1.

We implemented the three methods using the dataset for PTJ and SMK separately. The settings are shown in Appendix. B

5.2 Evaluation Metrics

A professional annotator, who had English teaching experience for three years, manually evaluated the results. The annotator labeled each generated feedback comment as *appropriate*, *partially appropriate*, or *inappropriate*; *partially appropriate* refers to the case that a generation result would become appropriate if part of it were edited. Specifically, we set the following two conditions: (1) the type of error is correctly identified, and (2) replacing a few words makes it appropriate. Consider the following example:

(6) **Most of the people think so.*
When referring simply to general *students*, you should use *most* as an adjective rather than a noun.

In Example (6), the feedback comment is partially appropriate because (1) the type of error (*most* is mistaken for *most of*) is correctly identified, and (2) replacing *students* with *people* makes it appropriate.

We measured performance by accuracy. We defined accuracy by the number of appropriate generation results divided by the total number of generation results, which will be referred to as a STRICT setting. Similarly, we defined RELAXED accuracy by the number of appropriate AND partially appropriate generation results divided by the total number of generation results.

5.3 Results

Table 2 shows the performance of the three methods. Contrary to our expectation, Table 2 reveals that the performance ordering in all PREP settings is RETRIEVE-AND-EDIT < RETRIEVAL-BASED < SIMPLE GENERATION. However, the performance order is SIMPLE GENERATION < RETRIEVE-AND-EDIT < RETRIEVAL-BASED in all GENERAL settings.

RETRIEVE-AND-EDIT, which edits the results of RETRIEVAL-BASED, performs worse than RETRIEVAL-BASED in each setting. This means that a correct feedback comment is often changed to incorrect by editing. Section 6.1 discusses the reasons for this.

Furthermore, SIMPLE GENERATION, which performs best in PREP, performs worst in GENERAL. Section 6.2 discusses why the counter-intuitive result occurs.

6 Discussion

6.1 Why Does Not Retrieve-and-edit Work Well?

6.1.1 The Problem: Over-editing

The generation results reveal that RETRIEVE-AND-EDIT is imposed to learn unnecessary edits in training. Ideally, it would only have to edit a few relevant words in the retrieved instance. In reality, however, it edits unnecessary words and phrases even when the retriever finds a similar instance. This is because superficially different feedback comments are given to the same error type as shown in the following:

(7) $\cdots$ **I disagree to have part-time job* $\cdots$
The verb *disagree* is not usually followed

[4] https://www.gsk.or.jp/en/catalog/gsk2019-b

	PREP				GENERAL			
	PTJ		SMK		PTJ		SMK	
	# sent	# com	# sent	# com	# sent	# com	# sent	# com
Train	12163	2439	12312	2341	18251	11510	19957	11792
Dev.	1129	245	1160	230	1315	848	1413	837
Test	1042	224	1023	214	1304	833	1328	772

Table 1: Statistics on the dataset. #sent and # com denote the number of sentences and comments, respectively

	PREP				GENERAL			
	PTJ		SMK		PTJ		SMK	
	Strict	Relaxed	Strict	Relaxed	Strict	Relaxed	Strict	Relaxed
SIMPLE GENERATION	**0.408**	**0.454**	**0.460**	**0.512**	0.192	0.244	0.196	0.224
RETRIEVAL-BASED	0.321	0.367	0.370	0.441	**0.276**	**0.360**	**0.296**	**0.380**
RETRIEVE-AND-EDIT	0.289	0.339	0.341	0.408	0.236	0.300	0.212	0.264

Table 2: Generation performance in accuracy.

by a to-infinitive phrase, so use the preposition 'with' and a gerund.

(8) **I don't disagree to have a part-time job* $\cdots$
A to-infinitive does not normally follow the verb *disagree*. Use the structure preposition + gerund instead.

The two refer to the same rule about *disagree to*, but are superficially very different. Because feedback comments are written manually in natural language, superficial variations are inevitable; however, this situation differs in source code generation, where the original RETRIEVE-AND-EDIT (Hashimoto et al., 2018) is proposed. Therefore, RETRIEVE-AND-EDIT must learn unnecessary editing such as in the feedback comment (7) from (8) in training and vice versa even though it would be enough to output the retrieved one as it is. We visualize the generation and copying probabilities in RETRIEVE-AND-EDIT to see the over-editing problem in more detail. Some examples of the generation probabilities of the generated comments are shown in Figure 1. The larger the generation probability, the redder the word gets. Figure 1 shows that most of the words are in a deep red, meaning that they are not copied from the retrieved feedback comment, but generated from the vocabulary. However, the editor ignores nearly all words in the retrieved feedback comment. This is not editor's desired behavior at all.

6.1.2 Improvement of RETRIEVE-AND-EDIT

We analyzed what type of editing is required. We focused on partially appropriate instances of RETRIEVAL-BASED (i.e., instances that are incorrect in the STRICT setting but correct in the RELAXED setting) because they can become appropriate with little editing. There were a total of 63 partially appropriate instances.

Our manual analysis reveals that there are two major types of parts that should be edited: extraction of English words and meaning of English (translated into Japanese). Their examples and percentages are shown in Table 3. In the example of extraction of English words in Table 3, *must* in the retrieved instance must be replaced by *can* in the input text. In the example of the meaning of English in Table 3, the Japanese translation of *smokers* in the retrieved instance must be replaced by the Japanese translation of *restaurants*. Note that some instances require multiple types of rewriting, and thus the sum of them does not equal 100%.

To deal with cases of "extraction of English words," the editor must learn the relationship between English words in the feedback comment and input text. To build such an appropriate editor, we must avoid the over-editing problem. To avoid over-editing, we should give the editor with training examples in which only a few words are replaced. One way to achieve this is to use the surface similarity between the feedback comments. During training, in addition to the similarity of the hidden state (cosine similarity) used in RETRIEVE-AND-

Target sentence	Retrieved feedback comment	Output of original retrieve-and-edit
…most **of** the restaurants in …	When *most of* follows a noun accompanied by the definite article *the*, it refers to a specific group. Use *most* as an adjective rather than a noun when simply referring to smokers in general.	When *most of* follows a noun accompanied by the definite article *the*, it refers to a specific group. Use *most* as an adjective rather than a noun when simply referring to people in general.
… I only **agree** students to have …	Use the verb *agree* in the structure <*agree* + for + someone + *to*-infinitive> to express ``*to agree for someone to do something*''.	The preposition *to* is not appropriate to mean ``*to enter into*'' when the verb *agree* is used. Look up the verb *agree* in a dictionary to choose the appropriate preposition.
… should declare **about** area for …	Since the verb claim is a transitive verb, the object does not require a preposition.	Since the verb area is a transitive verb, the object does not require a preposition.

Figure 1: Visualization of generation and copying probabilities of RETRIEVE-AND-EDIT.

Parts that need to be edited	Example of input	Example of retrieval and appropriate rewriting	Percentage
Extraction of English words	Student can part time job ...	The auxiliary verb "~~must~~ can" be followed by the original form of the verb	76%
Meaning of English words	Some of restaurants separate ...	When referring to general ~~smokers~~ restaurants, use "some" as an adjective rather than a noun.	40%
Others	... health of smoker and others.	The plural form is appropriate because it refers to general smokers. ~~Note that you then refer to it using "them".~~	11%

Table 3: Examples of the results of RETRIEVAL-BASED and parts that must be rewritten, and their percentages. Red text and strikeouts indicate proper editing. Note that the feedback comments are written in Japanese, but we provide an English translation.

EDIT, the surface similarity between the feedback comments can be considered. This enables us to train the editor using examples with few superficial differences.

However, for cases of "meaning of English words", avoiding over-editing is probably insufficient. In the example in Table 3, the Japanese word "smokers" must be replaced with the Japanese word "restaurants." To achieve such a replacement, it is necessary to understand the meaning of "smokers" and "restaurants" in Japanese. External knowledge, such as a dictionary containing information on the meaning of English words/phrases is required. Making these improvements are future work.

6.2 Why Does Performance of Simple Generation Varies?

The generation results reveal that SIMPLE GENERATION behaves differently in PREP and GENERAL. It often generates feedback comments similar to an existing training instance in PREP. However, it often generates a mixture of two or more existing feedback comments in GENERAL, which we call *mixed* feedback comments). An example is:

(9) ··· **this activity is will not disturb their* ···
Use an appropriate verb form in accordance with its subject. Check if the verb *be* is necessary.

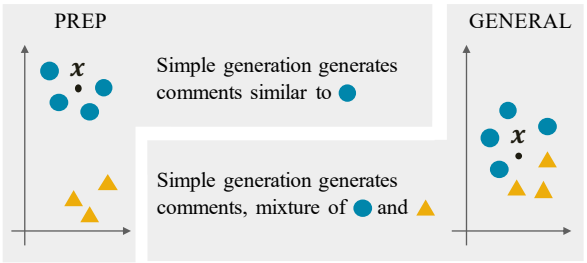
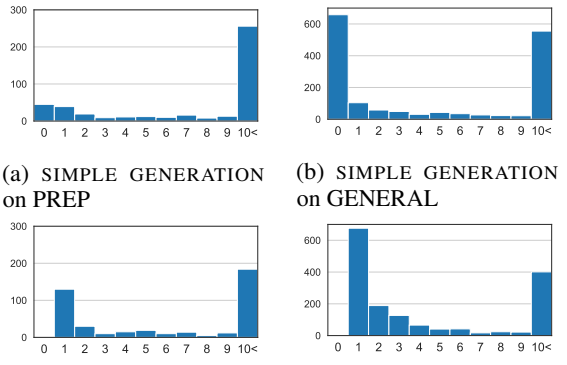


Figure 2: Hypothetical Distributions of Encoded Target Sentences in PREP and GENERAL; Data points are more separated in the former than in the latter.

The first sentence concerns the subject-verb agreement while the second sentence points out the extraneous *be*-verb. In GENERAL, mixed feedback comments tend to appear more often.

This is hypothetically explained as follows. Note first that sentences to be commented on are mapped onto the hidden space by the encoder as shown in Figure 2. In PREP, encoded target sentences are sparse and well separated in the space whereas in GENERAL they are denser, reflecting the fact that there exist far more types of comments in GENERAL. Here, the encoded target sentence $\boldsymbol{x}$ is more



(a) SIMPLE GENERATION on PREP

(b) SIMPLE GENERATION on GENERAL

(c) RETRIEVAL-BASED on PREP

(d) RETRIEVAL-BASED on GENERAL

Figure 3: The number of the same feedback comments in the training data. Horizontal axis: Number of the same comments in the training data. Vertical axis: Frequency in the test data.

likely to appear in the middle of several types of comments as shown on the right-hand side of Figure 2. In such cases, the decoder inevitably tends to generate a feedback comment corresponding to several types.

We manually counted the number of mixed feedback comments generated by SIMPLE GENERATION in PREP and GENERAL. We found that approximately 5% and 49% of the generation results in PREP and GENERAL, respectively, were mixed feedback comments, which confirm the above hypothesis.

Then, why does SIMPLE GENERATION perform best in PREP? One reason for this is the flexibility of SIMPLE GENERATION. Of the instances that RETRIEVAL-BASED fails to output correctly, 6% of the total instances become correct after rewriting a few words. These failures occur because the training data contain no instances of the same error type. However, SIMPLE GENERATION can deal with such instances.

Another reason is that the SIMPLE GENERATION prefers frequent feedback comments because it learns the conditional generation probability of feedback comments. When the model is unable to distinguish between different types of comments, it is a rational strategy to output more frequent comments to increase the accuracy.

To confirm this, we examined how many feedback comments were nearly the same in the training data. Because it is difficult to manually determine whether the comments are the same or not, we considered feedback comments to be the same if the normalized Levenshtein distance is less than 0.1. The results are shown in Figure 3. Figures 3a and 3c show that RETRIEVAL-BASED often outputs feedback comments with a frequency of one in the training data, whereas SIMPLE GENERATION often outputs comments with a frequency of two or higher in PREP. This implies that a SIMPLE GENERATION is less likely to generate low-frequency feedback comments, which may contribute to improving the accuracy. However, Figures 3b and 3d show that the same trend is not observed in GENERAL. For GENERAL, we can see that SIMPLE GENERATION often generates comments that have never appeared in the training data.

	PREP		GENERAL	
	PTJ	SMK	PTJ	SMK
SIMPLE GENERATION	0.083	0.066	0.064	0.036
RETRIEVAL-BASED	0.009	0.009	0.024	0.020
RETRIEVE-AND-EDIT	0.055	0.043	0.048	0.028

Table 4: Ratio of confusing false outputs.

6.3 Confusing False Generations

Some false generations results are more critical than others in terms of language learning; here, we define such cases as confusing false generations with the conditions: A feedback comment (i) contradicts an existing (true) rule and (ii) describes a plausible but wrong rule excluding cases that do not apply to the target sentence. Consider the following two examples:

(10) **I disagree to you.*
Since the verb *disagree* is a transitive verb, the object does not require the preposition *to*.

(11) **I disagrees to you.*
Since a verb following an auxiliary should be in base form, *to* is not necessary.

In Example (10), the feedback comment (i) contradicts the true rule that the verb *disagree* is an intransitive verb, and (ii) looks plausible. False generations like this are likely to confuse learners or make them acquire incorrect rules. Contrary to this, in Example (11), although the feedback comment states a true rule as it is, it does not apply to the target sentence at all and thus the writer will likely tell that there is something wrong with the feedback comment. For this reason, these cases are excluded from confusing false generations.

We manually counted the number of confusing false generations. Table 4 shows the statis-

tics on confusing false generations committed by each method. Table 4 shows the confusing generation rate in SIMPLE GENERATION and RETRIEVE-AND-EDIT are significantly larger than that in RETRIEVAL-BASED. Because RETRIEVAL-BASED is bound to output the retrieved instance, it will most likely avoid outputting feedback comments stating rules that do not exist. However, SIMPLE GENERATION and RETRIEVE-AND-EDIT are much flexible (RETRIEVE-AND-EDIT ignores nearly all words in the retrieved feedback comment as discussed in Section 6.1), which leads to more confusing false generations.

7 Future Direction

Based on the above findings, what direction should we take in the future in the task of feedback comment generation?

First, SIMPLE GENERATION is more effective when the type of error is limited, such as PREP. The limited scope of feedback comment is a common situation in actual education. For example, students are taught a specific grammatical item in a certain lesson.

However, when there is insufficient data for the type of error, as in the case of GENERAL, RETRIEVE-AND-EDIT can be effective if it is be improved. A better editor can be constructed by modifying the learning strategy and using external knowledge. If we can construct the proper editor, it is expected to output correctly for the 63 partially appropriate instances of RETRIEVAL-BASED (Section 6.1).

In addition, the latest architectures, such as BERT, can be used for the encoder and decoder. Our pilot study indicates that BERT is effective as an encoder, but it needs further investigation. We leave this for future work.

Considering practical situations, it is not sufficient to simply consider the accuracy alone. The number of confusing outputs of SIMPLE GENERATION and RETRIEVE-AND-EDIT increases more than that of the RETRIEVAL-BASED model (Section 6.3). When a learner receives the model's output as they are, the use of the generative model, which generates relatively many confusing comments, should be avoided. From another practical viewpoint, example-based methods, such as RETRIEVAL-BASED and RETRIEVE-AND-EDIT, have advantages that a SIMPLE GENERATION does not have. For example, users can see the retrieved instances, which makes it easier to select the output. Furthermore, when the retrieved instances are incorrect, the retrieved instances can be modified without retraining the model. These practical points of view suggest that each method should be used depending on the situation.

8 Conclusions

In this study, we investigated three feedback comment generation methods. We showed counter-intuitive results: (i) the performance of RETRIEVE-AND-EDIT was lower than the RETRIEVAL-BASED and (ii) the performance of the generation method varied depending on the task. Furthermore, we analyzed the causes of these results. Based on these analyses, the future directions are summarized as follows: (1) SIMPLE GENERATION is effective in a setting with few variations of feedback comments such as PREP, (2) RETRIEVE-AND-EDIT is expected to be promising for general feedback comment generation by suppressing over-editing and using external knowledge, (3) It will be necessary to develop a system that makes it easier for users to select the results, such as confidence estimation.

Acknowledgements

This work was partly supported by Japan Science and Technology Agency (JST), PRESTO Grant Number JPMJPR1758, Japan.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2016. Neural Machine Translation by Jointly Learning to Align and Translate.

Ziqiang Cao, Wenjie Li, Sujian Li, and Furu Wei. 2018. Retrieve, Rerank and Rewrite: Soft Template Based Neural Summarization. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 152–161.

Kelvin Guu, Tatsunori Hashimoto, Yonatan Oren, and Percy Liang. 2018. Generating Sentences by Editing Prototypes. *Transactions of the Association for Computational Linguistics*, 6(0):437–450.

Tatsunori B Hashimoto, Kelvin Guu, Yonatan Oren, and Percy S Liang. 2018. A Retrieve-and-Edit Framework for Predicting Structured Outputs. In *Advances in Neural Information Processing Systems 31*, pages 10052–10062.

Nabil Hossain, Marjan Ghazvininejad, and Luke Zettlemoyer. 2020. Simple and Effective Retrieve-Edit-Rerank Text Generation. In *Proceedings of the*

58th Annual Meeting of the Association for Computational Linguistics, pages 2532–2538.

Shinichiro Ishikawa. 2013. The ICNALE and sophisticated contrastive interlanguage analysis of Asian learners of English. *Learner corpus studies in Asia and the world*, 1:91–118.

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Shubha Guha, and Kenneth Heafield. 2018. Approaching Neural Grammatical Error Correction as a Low-Resource Machine Translation Task. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 595–606.

Jun'ichi Kakegawa, Hisayuki Kanda, Eitaro Fujioka, Makoto Itami, and Kohji Itoh. 2000. Diagnostic Processing of Japanese for Computer-Assisted Second Language Learning. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 537–546.

Yi-Huei Lai and Jason Chang. 2019. TellMeWhy: Learning to Explain Corrective Feedback for Second Language Learners. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 235–240.

Kathleen F Mccoy and Christopher A Pennington. 1996. English error correction: A syntactic user model based on principled mal-rule scoring. In *In Proceedings of the Fifth International Conference on User Modeling*, pages 59–66.

Ryo Nagata. 2019. Toward a Task of Feedback Comment Generation for Writing Learning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3206–3215.

Ryo Nagata, Kentaro Inui, and Shin'ichiro Ishikawa. 2020. Creating Corpora for Research in Feedback Comment Generation. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 340–345.

Ryo Nagata, Mikko Vilenius, and Edward Whittaker. 2014. Correcting Preposition Errors in Learner English Using Error Case Frames and Feedback Messages. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 754–764.

Courtney Napoles and Chris Callison-Burch. 2017. Systematically Adapting Machine Translation for Grammatical Error Correction. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 345–356.

Ildiko Pilan, John Lee, Chak Yan Yeung, and Jonathan Webster. 2020. A Dataset for Investigating the Impact of Feedback on Student Revision Outcome. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 332–339.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A Neural Attention Model for Abstractive Sentence Summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get To The Point: Summarization with Pointer-Generator Networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 1073–1083.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural Responding Machine for Short-Text Conversation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 1577–1586.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems 27*, pages 3104–3112.

Jason Weston, Emily Dinan, and Alexander Miller. 2018. Retrieve and Refine: Improved Sequence Generation Models For Dialogue. In *Proceedings of the 2018 EMNLP Workshop SCAI: The 2nd International Workshop on Search-Oriented Conversational AI*, pages 87–92.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *arXiv preprint arXiv:1609.08144*.

Zheng Yuan and Ted Briscoe. 2016. Grammatical error correction using neural machine translation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 380–386.

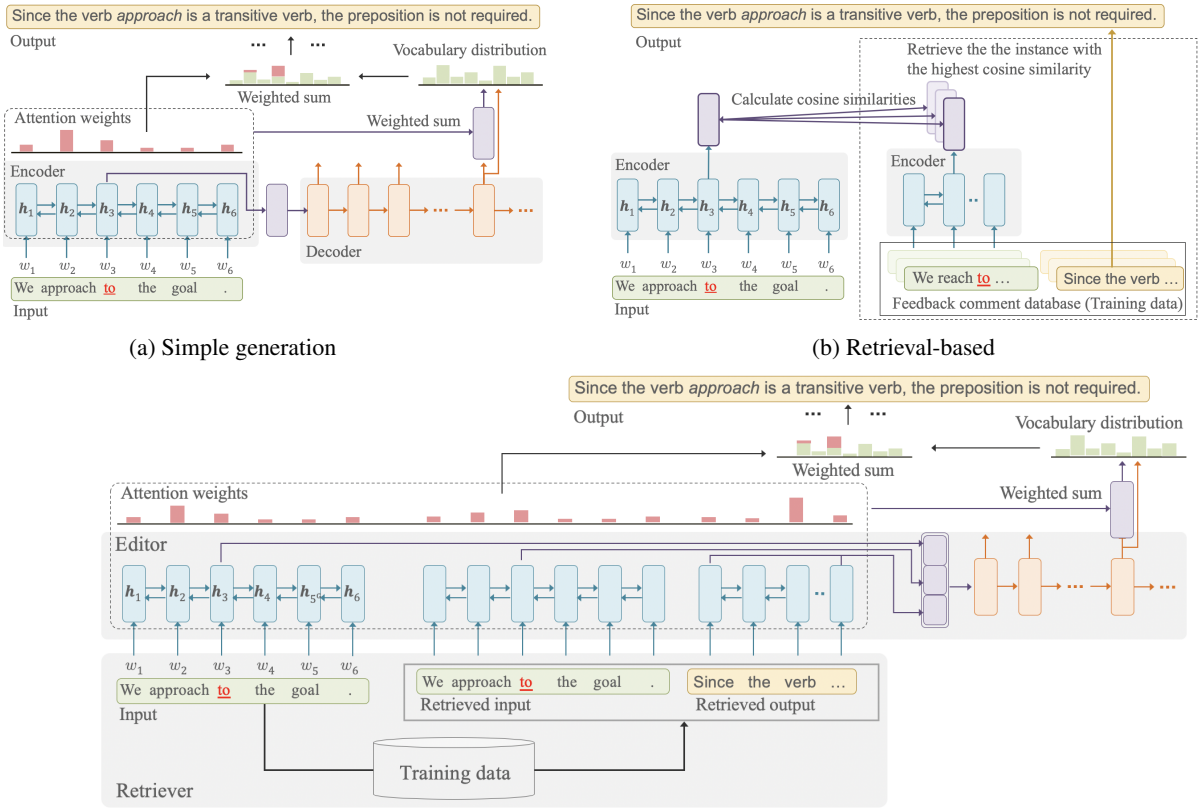
A Figures of Three Methods

Figure 4 shows the diagrams of each method.

B Hyperparameter

Setting of training We used a one-layer Bi-LSTM with 300 hidden sizes in all the comment generation methods. We used pre-trained word embeddings obtained from ICNALE (Ishikawa, 2013), which was the original corpus of the feedback comment dataset. We trained all methods for 50 epochs using the Adam optimizer with a learning rate of 0.001. We repeated the training five times with different seeds and adopted the one with the maximum BLEU in the development set.

Hyperparameter search We did a hyperparameter search regarding LSTM hidden sizes and training epochs. Specifically, 300, 400, 800, and 1600 for the hidden size and 10, 20, 30, 40, 50, 100, 150, and 200 training epochs. We adopted the ones that maximized BLEU in the validation set.



(a) Simple generation

(b) Retrieval-based

(c) Retrieve-and-edit

Figure 4: Overview of the three methods compared in this paper.