

Structure-Aware Procedural Text Generation from an Image Sequence

TAICHI NISHIMURA¹, ATSUSHI HASHIMOTO² (Member, IEEE), YOSHITAKA USHIKU² (Member, IEEE), HIROTAKA KAMEKO³, YOKO YAMAKATA⁴ (Member, IEEE), and SHINSUKE MORI³

¹Graduate School of Informatics, Kyoto University, Kyoto 606-8501, Japan

²OMRON SINIC X Corporation, Tokyo 113-0033, Japan

³Academic Center for Computing and Media Studies, Kyoto University, Kyoto 606-8501, Japan

⁴Graduate School of Information Science and Technology, The University of Tokyo, Tokyo 113-8656, Japan

Corresponding author: Taichi Nishimura (e-mail: nishimura.taichi.43x@st.kyoto-u.ac.jp).

ABSTRACT It is an important activity for our society to create new value by combining materials. From daily cooking to industrial manufacturing, procedural texts describe the way to do it allowing readers to reproduce procedures for these activities. As pointed by some previous studies for natural language understanding, one important property of the procedural text is its context dependency, which is the merging operations of materials and can be represented by a graph or tree structure. This paper aims to investigate the impact of explicitly introducing such a structure on the vision and language task of procedural text generation from an image sequence. To this end, we propose (1) a new dataset, which extends a definition of a tree structure *merging tree* to a vision and language version and (2) a novel structure-aware procedural text generation model, which learns the context dependency efficiently. Experimental results show that the proposed method can boost the performance of traditional versatile methods.

INDEX TERMS Natural language processing, text generation, procedural text, vision and language

I. INTRODUCTION

An archive of procedural texts, which have been written in the same abstraction level to human understanding, is an accumulation of wisdom in our practice. It is an ambitious challenge to obtain such an abstract representation of practical knowledge from low-level visual observation. As one of the tasks to this goal, procedural text generation from an image sequence [1], [2] has been proposed to investigate machine's capabilities in human-like understanding of instructional visual stories. This task is rising in popularity due to its goal-oriented property; the models must generate coherent sentences.

To generate coherent sentences, we assume that the model must preserve *context dependency*, which is the global dependency of actions and involved materials to a single final product. Previous work to generate a procedural text from an image sequence [1], [2] thoroughly focused on representing the global context in an image sequence, but it is not yet reached in capturing the context dependency explicitly. In terms of natural language understanding, some studies have represented the context dependency of a procedural text as a graph or tree structure [3]–[5], which leads the models to

reason about the path to the goal.

Inspired by these ideas, we aim to represent the context dependency in an image sequence as a structure and introduce it into the text generation models explicitly, allowing the models to generate coherent sentences. To this end, we propose (1) a new dataset, which extends the above proposed structure definition to a vision and language version and (2) a novel structure-aware procedural text generation model, which learns the context dependency efficiently.

Among the structure definitions proposed in previous studies, we refer to the Simplified Ingredient Merging Map in Recipes (SIMMR) dataset [3], which consists of procedural texts (recipes) and its structure, a “merging tree.” A merging tree is a tree structure describing the path through which materials are merged into a single final product. We adopt this structure because it is minimal but can fully express the context dependency for a procedural text generation task. As mentioned above, SIMMR is annotated to a dataset without visual data. Thus, to provide a vision and language version, we extend SIMMR to a new dataset called visual SIMMR (vSIMMR), where each merging tree is annotated to an image sequence in a subset of the traditional world-largest

dataset¹.

Using this dataset, we propose a structure-aware procedural text generation model that gains the global coherency by explicitly predicting a merging tree. Additionally, based on the intuition that the content of the instructions should be identical regardless of the representation modality, we add a tree re-prediction module to facilitate the generated procedural texts to be more coherent by preserving the merging tree structure.

We tested the proposed method on the task of generating cooking recipes using a variety of traditional procedural text generation models. The experimental results show that the proposed method can lead the base models to generate more coherent recipes in a versatile manner. Human evaluation and qualitative analysis support the impact of an increase in the metrics from the perspective of the human senses.

II. RELATED WORK

We discuss the proposed vSIMMR dataset by comparing with other datasets in II-A and II-B. We also show the novelty of the proposed method, introducing the other procedural text generation models in II-C.

A. HOW-TO DATASETS FOR VISION AND LANGUAGE

How-to datasets consisting of procedural texts with visual observations are rising in popularity for learning processes of complex tasks [12]–[14]. Among them, the cooking domain has been targeted by several researchers, and our proposed dataset, vSIMMR is also constructed in this domain. In Table 1, we compare the vSIMMR dataset with other cross-modal recipe datasets from multiple perspectives. To analyze the processes of food preparation, there are roughly two possible visual information resources: videos and image sequences.

Cooking videos are a natural input that involves observation of food states. Zhou *et al.* [8] developed the YouCook2 dataset, which consists of 2,000 YouTube videos for 89 recipes. The EPIC-Kitchens dataset [7] and Breakfast dataset [6] are large-scale cooking video datasets, but recipe texts are not attached with videos. Although cooking videos have an advantage in its observation of a continuous food-state transition, the number of cooking observations is limited. Hence, it suffers from a shortage of recipes covered in the training data.

Image sequences are an alternative way to observe food states, and the number of covered recipes tends to be much larger than that of video datasets. Recipe QA [9] is a recent dataset for cross-modal QA tasks. In [1], data from the same web site are used for a recipe generation task. Among this kind of dataset, the Cookpad Image Dataset [10] is the world's largest dataset, which contains 1.7M recipes and (optional) image sequences. In addition, it is the only cross-modal dataset with an ingredient list, and our vSIMMR dataset is based on the Cookpad Image Dataset.

B. STRUCTURE ESTIMATION FOR CONTEXT DEPENDENCY

In the 1980's, Momouchi [15] proposed PT (Procedural Text)-chart, which represents an entire workflow of actions and materials as the Backus-Naur form. This work is pioneer research to convert procedural texts into graphical representations. Recently, recipes have also been targeted in this field, and Kiddon *et al.* [4] proposed an unsupervised method to estimate a graph structure, called an *action graph* from a recipe text. An action graph is also used in the challenging visual reference resolution task described in [16], [17]. Mori *et al.* [5] provided a *recipe flow graph* dataset, which is a fine-structured machine-readable recipe representation. Jermura-wong and Nizar [3] have proposed the SIMMR dataset, which provides a merging tree, a tree structure tracking the ingredient merging operations.

Although the above work has focused on parsing text-only recipes, little work has not addressed cross-modal analysis due to limited available datasets. Pan *et al.* [11] recently created a novel cross-modal dataset, MM-ReS dataset. This dataset consists of recipes, image sequences, and annotated tree structures, allowing us to analyze the cause-and-effect relations between step texts and images in the recipe and image sequence.

Our proposed dataset vSIMMR can be seen as a simple extension of the MM-ReS dataset by connecting images not only with step texts but also with ingredients, while their work connects images only with a recipe. This difference allows us to further investigate cause-and-effect relations between the ingredients and images. Therefore, our dataset vSIMMR can be said as the only cross-modal dataset with recipes, ingredient lists, tree structures, and image sequences as shown in Table 1.

C. PROCEDURAL TEXT GENERATION FROM VARIOUS INPUTS

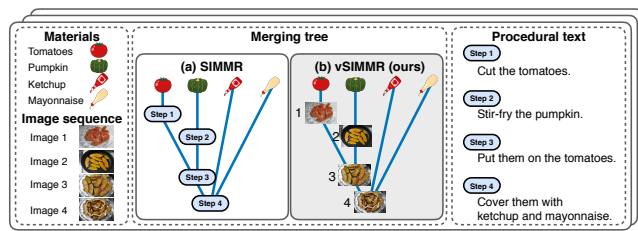
In the field of natural language generation, a procedural text is a popular target because it requires a model to generate coherent sentences by understanding its long context. Among the many types of procedural text, recipes with rich descriptions of a large variety of actions and involved ingredients are featured. Kiddon *et al.* [18] and Bosselut *et al.* [19] proposed a model to generate a recipe from its title and ingredients. To deal with the long context of a recipe, the former uses an attention mechanism, and the latter uses reinforcement learning.

Recently, many vision and language datasets have appeared in the cooking domain, and a challenging task, recipe generation from an image, has also been proposed. Salvador *et al.* [20] firstly tackled this problem and proposed a model based on Transformer [21] to generate a recipe from a single image of a prepared dish. Later, Wang *et al.* [22] proposed a structure-aware model to generate a recipe from an image by imitating a structure of the ground truth recipe in an unsupervised manner.

¹We are going to release our dataset for research purpose publicly.

TABLE 1. Comparative overview of relevant cooking datasets.

Datasets	Recipes?	Ingredients?	Structure?	Visual data	#Recipes
Breakfast [6]				Video	N/A
EPIC-Kitchens [7]				Video	N/A
YouCook2 [8]	✓			Video	89
RecipeQA [9]	✓			Image sequence	19,779
Story boarding [1]	✓			Image sequence	16,405
Cookpad Image Dataset [10]	✓	✓		Image sequence	1,715,595
Recipe Flow Graph [5]	✓	✓	✓	N/A	266
Action Graph [4]	✓	✓	✓	N/A	133
SIMMR [3]	✓	✓	✓	N/A	260
MM-ReS [11]	✓		✓	Image sequence	9,850
vSIMMR (ours)	✓	✓	✓	Image sequence	2,103

**FIGURE 1.** An overview of the vSIMMR dataset. Images in the image sequence are aligned with the intermediate nodes in the merging tree and the step texts in the procedural text.

There have also been two recent attempts to generate a recipe from an image sequence depicting the intermediate food states [1], [2]. To represent the global context in an image sequence, Chandu *et al.* [1] incorporated a finite state machine (FSM), and Nishimura *et al.* [2] incorporated a joint embedding model into a recipe generator. In line with the above studies, this paper proposes a method for explicitly modeling the context dependency as a structure in a network architecture that is compatible with previous methods.

III. PROPOSED DATASET: VSIMMR

We explain the proposed cross-modal dataset, vSIMMR, which consists of 2,106 procedural texts, image sequences, material lists, and annotated tree structures. In the following subsections, we firstly provide an overview of the SIMMR dataset [3] (III-A) and then describe important extension to it (III-B) and annotation process of the vSIMMR dataset (III-C).

A. MERGING TREE: CONTEXT DEPENDENCY DESCRIPTION IN SIMMR

SIMMR was originally proposed to represent the context dependency of a procedural text as a tree structure “merging tree,” which describes the merging path of materials to a single final product. Figure 1(a) exemplifies the merging tree of the procedural text. The leaves (terminal nodes) and intermediate nodes correspond to the materials and steps, respectively. The root node is the final step and indicates the final product of the procedural text.

**FIGURE 2.** Our browser-based annotation platform. A user can annotate a merging tree by clicking unused ingredients and intermediate nodes representing a mixture of ingredients at each step.

B. EXTENSION TO SIMMR: LARGER DATASET WITH VISUAL REFERENCE

As shown in Figure 1(b), we extend SIMMR to vSIMMR by annotating merging trees with image sequences. This extension allows us to represent the common tree structure between the visual and textual world because images in the image sequence are aligned with the step texts in the procedural text. Moreover, in terms of the number of labeled data items, the vSIMMR dataset is about eight times larger than the SIMMR dataset as shown in Table 1, and this point is also an important extension to SIMMR.

C. ANNOTATION PROCESS OF THE TREE STRUCTURE

To build the vSIMMR dataset, we annotated merging trees to a subset of the Cookpad Image Dataset, which consists of procedural texts (= recipes), material lists (= ingredient lists), and image sequences. The annotation process is roughly divided into two parts: (i) material name normalization and (ii) merging tree annotation.

(i) Material name normalization. Firstly, to annotate merging trees correctly, we manually merged any notational variations of material names. This normalization is necessary because all procedural texts and material lists in the Cookpad Image Dataset are user-generated and thus material names contain inconsistency [23]; for example, some write “2tbsp

of oil” or “2tablespoon of oil” and others simply write “oil”. We manually normalized them to the simplest form (in the above example, we converted “2tbsp of oil” and “2tablespoon of oil” into “oil”).

(ii) Merging tree annotation. Then, we annotated a merging tree referring to an image sequence and procedural text. Figure 2 shows our annotation platform, presenting the image sequence on the left pane and the corresponding procedural text on the right pane. We asked one annotator to construct the merging tree by connecting unused ingredients and intermediate nodes representing a mixture of ingredients at each step. In this process, if steps are irrelevant to the cooking process, we instructed the annotator to remove such step texts and images from an image sequence and procedural text; for example, in Figure 2, text and image of step 1 “Prepare ingredients. Left is the yogurt and right is the avocado” was removed. When the annotator submits the annotated structure to the server, the system validates whether all ingredients are used and whether the created structure is a tree structure.

IV. PROPOSED METHOD

After presenting the task definition and an overview, two main modules of the proposed method are described in IV-A and IV-B, then IV-C presents an implementation idea for semi-supervised learning.

Task definition. Let \mathcal{D}_{tr} be a training dataset, where $\{\mathbf{x}, \mathbf{y}, \mathbf{g}\} \in \mathcal{D}_{tr}$ is an input \mathbf{x} , procedural text \mathbf{y} , and its structure (e.g., merging tree) \mathbf{g} . $\mathbf{x} = \{\mathbf{x}_v, \mathbf{x}_{mat}\}$ has two components; an image sequence $\mathbf{x}_v = \{x_v^n | n = 1, \dots, N\}$ and a material list $\mathbf{x}_{mat} = \{x_{mat}^m | m = 1, \dots, M\}$. N and M are the number of images and materials, respectively. $\mathbf{y} = \{y^n | n = 1, \dots, N\}$ consists of N steps of textual instructions y^n s, where $\{x_v^n, y^n\}$ has a one-to-one correspondence. Note that \mathbf{g} is unavailable for most of the data points in \mathcal{D}_{tr} (semi-supervised setting). Hence, the procedural text generation task described in this paper is to train a model that estimates a procedural text $\hat{\mathbf{y}}$ from an input \mathbf{x} .

Overview of the proposed method. Figure 3 shows an overview of the proposed method. To incorporate context dependency into the decoder for gaining global coherency, our model tries to estimate the structure $\hat{\mathbf{g}}$ explicitly and use it in the decoder. This is roughly described by the combination of an encoder $E : \mathbf{x} \rightarrow \mathbf{z}, \hat{\mathbf{g}}$ and a decoder $D : \mathbf{z}, \hat{\mathbf{g}} \rightarrow \hat{\mathbf{y}}$, where \mathbf{z} is the latent representation of input \mathbf{x} . Note that both E and D are trained in an end-to-end manner instead of training E and D separately.

Additionally, based on our intuition that a merging tree is identical regardless of the representation modality, we add a module $E_y : \hat{\mathbf{y}} \rightarrow \hat{\hat{\mathbf{g}}}$ to re-predict the merging tree $\hat{\hat{\mathbf{g}}}$ from $\hat{\mathbf{y}}$. Here, this tree re-prediction module encourages the generated procedural text to be more coherent by preserving the merging tree.

A. STRUCTURE-AWARE PROCEDURAL TEXT GENERATION MODEL

Our model generates a procedural text coherently through the following three processes: First, (i) encodes \mathbf{x} into a latent representation \mathbf{z} . Then, (ii) determines $\hat{\mathbf{g}}$ through a Gumbel softmax resampling [24], which is employed for end-to-end training. Finally, (iii) decodes $\hat{\mathbf{y}}$ from $\hat{\mathbf{g}}$ with a structured decoder of a Tree-LSTM.

(i) Link probability matrix calculation. A graph structure, such as a merging tree, can be predicted from a probability matrix whose ij -element represents the probability that a link exists from the i -th node to the j -th node. Because a processed material cannot be merged by returning to a previous step, a step-to-step link has an order constraint, whereas a material-to-step does not. Hence, we prepare two sub-modules separately to calculate the two link probability matrices (LPMs): step-to-step LPM $P_{v \rightarrow v} \in \mathbb{R}^{N \times N}$ and material-to-step LPM $P_{mat \rightarrow v} \in \mathbb{R}^{M \times N}$.

(i-1) Step-to-step LPM calculation. Each image x_v^n is fed to a pre-trained image encoder F_v . This projects x_v^n onto z_v^n , where we define $\mathbf{z}_v = \{z_v^n | n = 1, \dots, N\}$. To calculate a link probability between two steps, \mathbf{z}_v is further converted into row and column feature vectors \mathbf{z}_v^1 and \mathbf{z}_v^2 using different biLSTMs $E_{v \rightarrow v}^1$ and $E_{v \rightarrow v}^2$, respectively. Then, each element $p_{v \rightarrow v}^{ij}$ in the step-to-step LPM $P_{v \rightarrow v}$ is calculated as follows:

$$p_{v \rightarrow v}^{ij} = \begin{cases} \varepsilon & (i \leq j) \\ \frac{\exp(z_{v,i}^1 \cdot z_{v,j}^2)}{\sum_k \exp(z_{v,i}^1 \cdot z_{v,k}^2)} & (\text{otherwise}) \end{cases} \quad (1)$$

where $\mathbf{z}_v^1 = E_{v \rightarrow v}^1(F_v(\mathbf{x}_v))$, $\mathbf{z}_v^2 = E_{v \rightarrow v}^2(F_v(\mathbf{x}_v))$.

In Eq (1), $z_{v,i}^1$ is the i -th element of \mathbf{z}_v^1 and $z_{v,j}^2$ is the j -th element of \mathbf{z}_v^2 . $p_{v \rightarrow v}^{ij}$ is set to $\varepsilon (= 1.0 \times 10^{-7})$ for numerical stability when $i \leq j$ to preserve the order constraint.

(i-2) Material-to-step LPM calculation. Next, we calculate the material-to-step LPM. For the step side, we calculate \mathbf{z}_v^{max} from \mathbf{z}_v^1 and \mathbf{z}_v^2 with an element-wise max-pooling layer. The material-side feature is calculated in two stages. First, each x_{mat}^m is converted into a distributed representation by F_{mat}^2 . Second, we further transform them into \mathbf{z}_{mat} , which is aware of ordering information of material list \mathbf{x}_{mat} . This encoding is necessary because we assume that a user-generated material list implies some order, although a material list is an unordered set generally. Specifically, as described in [26], we utilize another biLSTM $E_{mat \rightarrow v}$, which further encodes each distributed representation into a material feature $\mathbf{z}_{mat}^m \in \mathbf{z}_{mat}$. Finally, each element $p_{mat \rightarrow v}^{ij}$ in the material-to-step LPM $P_{mat \rightarrow v}$ is calculated as follows:

$$p_{mat \rightarrow v}^{ij} = \frac{\exp(z_{mat \rightarrow v}^i \cdot z_{v,j}^{max})}{\sum_k \exp(z_{mat \rightarrow v}^i \cdot z_{v,k}^{max})}, \quad (2)$$

$$\text{where } \mathbf{z}_{mat} = E_{mat \rightarrow v}(F_{mat}(\mathbf{x}_{mat})). \quad (3)$$

²For F_{mat} , we use word2vec [25] pre-trained on all recipes in \mathcal{D}_{tr} .

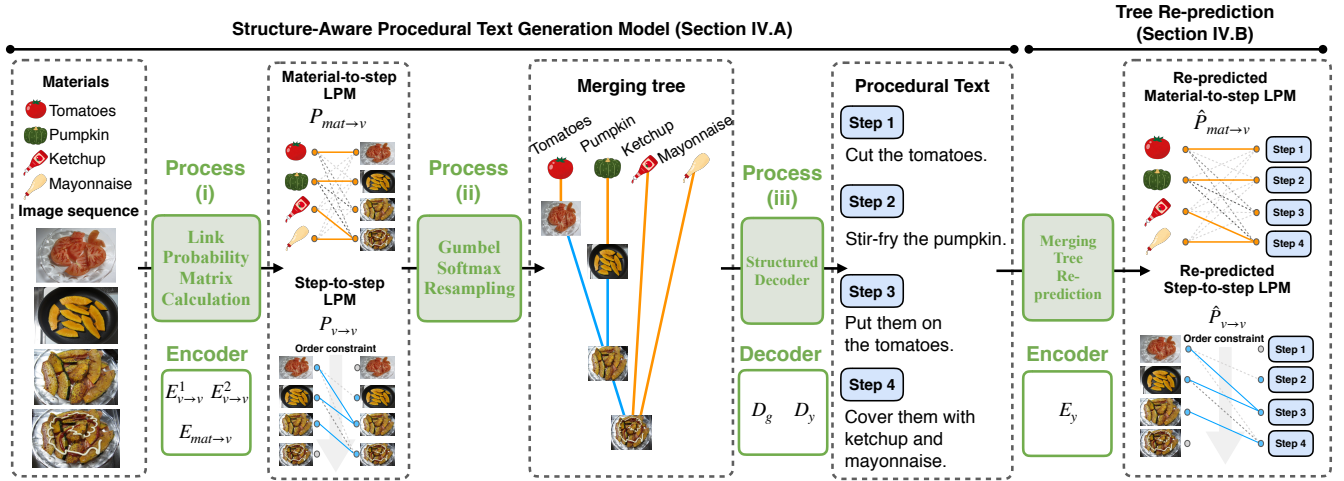


FIGURE 3. An overview of the proposed method. The structure-aware procedural text generation model (Section IV-A) has three processes: (i) Link Probability Matrix Calculation, (ii) Gumbel Softmax Resampling, and (iii) Structured Decoder. Additionally, the model re-predict a merging tree from a generated procedural text (Section IV-B).

When the sample has an annotation of the structure \mathbf{g} , we calculate a tree prediction loss $\mathcal{L}_{tp}(\mathbf{g}, P_{v \rightarrow v}, P_{mat \rightarrow v})$ as a column-wise cross-entropy loss.

(ii) Gumbel softmax resampling. To make the model fully derivative, we use a Gumbel softmax resampling. To determine the tree structure $\hat{\mathbf{g}}$ directly from the LPMs, we apply the straight-through (ST) version of Gumbel softmax resampling. The temperature parameter τ is set to 0.5 according to the discussion in [27], [28]³.

(iii) Structured decoder. To incorporate the context dependency based on $\hat{\mathbf{g}}$ into a text generator, we use a variant of Tree-LSTM, called Child-sum Tree LSTM [29]. Child-sum Tree-LSTM $D_g : z_v^{max}, z_{mat}, \hat{\mathbf{g}} \rightarrow \mathbf{h}$, where $\mathbf{h} = \{h^n | n = 1, \dots, N\}$, converts intermediate nodes of $\hat{\mathbf{g}}$ into hidden states \mathbf{h} . In forwarding phase, the Child-sum Tree-LSTM updates the hidden state h_s and memory cell c_s from each child and $z_{v,n}^{max}$ based on observation. See appendix A for the calculation details.

The output of Tree-LSTM \mathbf{h} is further decoded using a procedural text generator $D_y : z_v^n, h^n \rightarrow \hat{y}^n$. As instances of D_y , we used five state-of-the-art procedural text generators in our experiments. Finally, the model generates a procedural text, and the cross-entropy loss \mathcal{L}_{text} is calculated.

B. MERGING TREE RE-PREDICTION

Because the tree structure is identical regardless of the visual and linguistic representation, it should be consistently re-predictable from the textual information in nature. Explicitly preserving such a structure facilitates the generated procedural texts to be more coherent. Hence, we prepared $E_y : \hat{\mathbf{y}} \rightarrow \hat{\mathbf{z}}_{\hat{\mathbf{y}}}$ for predicting $\hat{P}_{v \rightarrow v}$ and $\hat{P}_{mat \rightarrow v}$. The re-predicted matrices are then compared to \mathbf{g} , where \mathbf{g} is the ground truth (for samples with an annotation) or a result

determined by column-wise argmax on $P_{v \rightarrow v}$ and $P_{mat \rightarrow v}$ (for samples without an annotation). In addition, $\hat{P}_{mat \rightarrow v}$ is simply obtained by substituting z_v^{max} in Eq (3) into $\hat{\mathbf{z}}_{\hat{\mathbf{y}}}$. Here, $\hat{P}_{v \rightarrow v}$ is obtained by substituting z_v^1 and z_v^2 in Eq (1) into z_v^{max} and $\hat{\mathbf{z}}_{\hat{\mathbf{y}}}$, respectively. Finally, a tree consistency loss $\mathcal{L}_{tc}(\mathbf{g}, \hat{P}_{v \rightarrow v}, \hat{P}_{mat \rightarrow v})$ is calculated as the column-wise cross-entropy loss on $\hat{P}_{v \rightarrow v}$ and $\hat{P}_{mat \rightarrow v}$ with supervision \mathbf{g} .

C. VAE-BASED RECONSTRUCTION MODULES FOR SEMI-SUPERVISED LEARNING

As discussed in [30] for semi-supervised learning, we use two variational autoencoder (VAE) architectures [31]: a tree2image VAE and a step2image VAE. For the VAE models, we employ β VAE [32], which is widely used owing to its ability to disentangle the latent representations. We calculate the reconstruction losses $\mathcal{L}_{t2i}(\mathbf{h}, z_v)$ and $\mathcal{L}_{s2i}(\mathbf{z}_{\hat{\mathbf{y}}}, z_v)$ in tree2image VAE and step2image VAE as follow:

$$\begin{aligned} \mathcal{L}_{t2i}(\mathbf{h}, z_v) &= \mathbb{E}_{q_\phi^1} [\log p(z_v | \mathbf{h})] - \beta D_{KL}(q_\phi^1(\mathbf{h} | z_v) || p(\mathbf{h})) \quad (4) \\ \mathcal{L}_{s2i}(\mathbf{z}_{\hat{\mathbf{y}}}, z_v) &= \mathbb{E}_{q_\phi^2} [\log p(z_v | \mathbf{z}_{\hat{\mathbf{y}}})] - \beta D_{KL}(q_\phi^2(\mathbf{z}_{\hat{\mathbf{y}}} | z_v) || p(\mathbf{z}_{\hat{\mathbf{y}}})) \quad (5) \end{aligned}$$

where q_ϕ^1 represents concatenated networks of the encoders E and Tree-LSTM D_g and q_ϕ^2 represents the networks appended the re-prediction module E_y to q_ϕ^1 . Here, β is set to be 4 according to the discussion in [32].

Finally, the model is trained with the loss function \mathcal{L} , which is defined as follows:

$$\mathcal{L} = \mathcal{L}_{text} + \mathcal{L}_{tp} + \lambda_{tc} \mathcal{L}_{tc} + \lambda_{t2i} \mathcal{L}_{t2i} + \lambda_{s2i} \mathcal{L}_{s2i}, \quad (6)$$

where the λ 's are hyper-parameters tune by grid search with a validation set (see appendix B).

³This setting is also supported by an experience appearing described in appendix C

TABLE 2. Statistics of our Cookpad Image Dataset split (data used in vSIMMER are excluded).

Cookpad Image Dataset	train	val	test
#recipes	163,525	18,051	20,193
#steps	6.24	6.15	6.26
#words	148.52	147.02	148.50
#ingredients	7.85	7.79	7.86

TABLE 3. Statistics of vSIMMR.

vSIMMR	train	val	test
#recipes	1,603	250	250
#steps	6.78	6.74	6.85
#words	118.23	113.91	114.68
#ingredients	6.58	6.37	6.64

V. EXPERIMENTS

This section presents the evaluation of the generated procedural texts in the three criteria: automatic evaluation (V-C), human evaluation (V-D), and qualitative analysis (V-E), with a thorough ablation. We also discuss the merging tree accuracy (V-D) and the satisfactory number of labeled data items for procedural text generation (V-G).

A. DATASETS

We used the Cookpad Image Dataset, which has 1.7M recipes (= procedural texts) and ingredient lists (= material lists) with (optional) image sequences. For our experiments, we discarded recipes having steps without images. Each recipe has a complete image sequence x_v , a recipe y , and an ingredient list x_{mat} . We further discarded some recipes with less than two steps or two ingredients because such a recipe has little ambiguity in its structure g . As a result, the size of the dataset is 200k.

Preprocessing. Each image was resized so that the longer edge had 256 pixels while keeping the aspect ratio. A central region with 224×224 pixels was then cropped. All recipes in the Cookpad Image Dataset are written in Japanese. Thus, to split the sentences into words, we used a Japanese morphological parser, KyTea [34]. All words appearing less than three times were then replaced with an unknown word symbol. Table 2 and 3 show the dataset's statistics.

B. IMPLEMENTATION DETAILS

As the encoder for visual inputs F_v , we used the one proposed in [2]. This model, based on ResNet50 [35], was pre-trained by an image-step retrieval task on the Cookpad Image Dataset. We preliminarily confirmed that this metric learning provides a better performance in recipe generation than pre-training on ImageNet [36], owing to the domain fitting. We set the output dimension of the image encoder F_v to 2,048. The parameters of the pre-trained model F_v were fixed during training for recipe generation, as described in previous studies [1], [2], [37], [38]. The output dimension of word2vec F_{mat} for the ingredient encoder was set to 300, and the dimension of the LSTM hidden layer of encoders, $E_{v \rightarrow v}^1$, $E_{v \rightarrow v}^2$ and E_y , was set to 512. The batch size was set

to 16, and Adam [39] was used as the optimizer.

Models. To prove the versatile impact of the proposed method, we tested it on various state-of-the-art methods for image-sequence-to-recipe generation, as described below.

- **Images2seq** [37] was originally proposed for the task of visual storytelling (ViST). This model uses biLSTM to consider the global context in an image sequence.
- **GLAC Net** [38] achieved a state-of-the-art performance on ViST. This model takes the global context by fusing global and local image feature vectors encoded by biLSTM and F_v .
- **SSiD** [1] is a method proposed for procedural text generation. It vectorizes steps using sent2vec [40], and then they are clustered by k-means, where the obtained clusters are treated as the states of an FSM. Based on the transition probabilities of the FSM, the model explores the change of the context to represent the global context richly.
- **SSiL** [1] is a variant of SSiD, which has an additional loss of Kullback-Leibler divergence to better imitate the step transition of the ground truth.
- **RetAttn** [2] achieves a comparative method to SSiL, but by a different approach; it pre-trains the model with image-text retrieval task, then in the text generation task, the retrieved text features are involved with an attention mechanism to generate a better quality text.

Ablations. For all five methods above, we conducted ablation studies on the following variations to reveal the impact of a merging tree estimation.

- **Baseline (original)** does not use any modules of the proposed method⁴.
- **Half model** computes the tree prediction loss \mathcal{L}_{tp} for samples with the ground truth with the text generation loss \mathcal{L}_{text} . In addition, \mathcal{L}_{t2i} is also calculated for samples with no annotation.
- **Full model** calculates \mathcal{L}_{tc} in addition to the half model. In the full model, two losses \mathcal{L}_{t2i} and \mathcal{L}_{s2i} ⁵ are additionally calculated for samples with no annotation.

Note that the baseline models originally have no ingredient lists in their inputs. Thus, for a fair comparison, we calculate the ingredient vectors in the same manner as with the proposed method and concatenate the average vector of the ingredient vectors with each image feature vector. When training and evaluating the baseline models, for a fair comparison, we used recipes in both the vSIMMR and Cookpad Image Dataset.

Enhanced baselines. The number of trainable parameters in the baseline model is about 1.5 times smaller than that in the half and full models. This difference occurs because the full model has additional modules for encoding the structure although the number of parameters in the text generator is

⁴The parameter setting of each model is followed to the original paper.

⁵In the half model, \mathcal{L}_{s2i} cannot be calculated because the step vectors $z_{\hat{g}}$ appear only in the re-prediction module E_y in q_{ϕ}^2 , which is the full model (see Eq (5)).

TABLE 4. Word-overlap metrics for the five base models with half and full models. The scores in bold are the best for each base model. B=BLEU, RL=ROUGE-L, D=Distinct, I=Ingredient, and Ac=Action. * indicates statistically significant difference ($p < 0.001$) from the base models (original) through bootstrap sampling [33].

	B1	B4	RL	D1	D2	I-B1	I-B2	I-B3	I-B4	I-RL	Ac-B1	Ac-B2	Ac-B3	Ac-B4	Ac-RL
Images2seq (original)	27.5	5.1	18.4	38.3	54.7	7.0	1.8	0.5	0.1	9.7	18.2	8.1	3.8	2.0	18.4
Images2seq (wide)	25.2	4.6	17.5	38.1	56.8	4.0	0.8	0.1	0.0	0.8	13.0	5.3	2.6	1.4	16.6
Images2seq (deep)	22.2	3.4	17.3	38.3	54.6	4.2	0.8	0.1	0.0	6.6	11.1	5.0	2.6	1.5	16.2
+ Half	27.8*	5.8*	20.6*	51.1*	75.0*	7.9*	2.2*	0.7*	0.2	12.7*	18.8*	8.4*	3.9	2.0	21.2*
+ Full	29.6*	6.3*	21.7*	47.6*	71.0*	8.8*	2.7*	0.9*	0.3*	13.8*	21.4*	9.4*	4.3*	2.1	22.9*
GLAC Net (original)	28.5	5.9	21.4	46.6	69.0	9.5	2.9	0.9	0.3	13.2	21.2	9.3	4.3	2.1	22.8
GLAC Net (wide)	26.3	5.1	19.3	40.6	58.9	7.7	2.1	0.6	0.2	10.8	18.1	8.0	3.9	2.1	20.4
GLAC Net (deep)	27.7	5.3	19.8	41.5	60.4	8.2	2.3	0.7	0.2	11.2	20.5	9.2	4.3	2.4	21.4
+ Half	28.5	5.9	21.8*	46.7	68.8	10.5*	3.4*	1.1*	0.4	15.6*	21.0	9.4	4.4	2.3	23.1
+ Full	28.9	6.1	21.3	47.2*	69.9*	11.8*	3.8*	1.2*	0.4	16.3*	21.0	9.1	4.4	2.3	22.5
SSiD (original)	28.7	6.0	20.9	45.5	66.6	8.6	2.7	0.8	0.2	13.1	20.1	8.7	4.0	2.1	21.6
SSiD (wide)	28.6	5.7	19.6	41.0	60.0	7.9	2.1	0.6	0.1	11.3	17.9	7.6	3.5	1.7	19.4
SSiD (deep)	27.6	5.2	19.7	41.0	59.1	6.9	1.9	0.5	0.2	10.6	20.1	9.0	4.3	2.2	21.8
+ Half	30.3*	6.2*	20.8	43.9	65.1	11.4*	3.7*	1.3*	0.4*	16.4*	19.8	8.7	4.4*	2.2	22.1*
+ Full	31.1*	6.4*	21.6*	48.3*	71.0*	12.9*	4.1*	1.2*	0.4*	16.7*	21.5*	9.5*	4.5*	2.2	23.5*
SSiL (original)	31.0	6.3	21.4	45.5	66.8	8.3	2.4	0.7	0.2	12.5	19.8	8.7	4.0	2.0	22.1
SSiL (wide)	26.2	5.5	20.6	46.3	67.4	9.2	2.7	0.8	0.2	13.0	21.1	9.4	4.5	2.3	22.3
SSiL (deep)	27.8	5.6	20.3	42.9	62.8	8.3	2.4	0.7	0.2	12.1	19.9	8.6	4.0	2.1	22.3
+ Half	28.1	5.4	21.4	46.7*	68.2*	11.3*	3.7*	1.2*	0.4*	16.9*	18.7	8.1	3.8	2.0	22.2
+ Full	30.4	6.4	21.9*	47.3*	70.9*	12.4*	4.2*	1.4*	0.4*	17.0*	21.4*	9.5*	4.5*	2.3*	23.0*
RetAttn (original)	32.2	6.5	21.6	40.2	60.3	11.2	3.3	1.0	0.3	14.5	22.1	9.0	3.2	1.5	21.2
RetAttn (wide)	28.3	5.9	21.1	44.5	65.1	12.0	3.1	1.2	0.3	14.1	22.0	8.9	4.0	1.9	22.0
RetAttn (deep)	29.8	6.3	21.5	44.1	65.4	12.1	3.7	1.2	0.3	14.7	22.1	9.1	4.2	1.9	22.5
+ Half	32.2	6.5	21.8	52.4*	77.8*	11.9*	3.7*	1.2	0.3	14.8	21.8	9.4*	4.2*	2.0*	22.9*
+ Full	33.2	7.1	22.1	52.7*	78.6*	12.1*	3.8*	1.2	0.3	14.8	22.3	9.5*	4.2*	2.0*	23.1*

TABLE 5. Human evaluation results. RetAttn was used for the “Baseline,” “Half,” and “Full” models. Among the three types of baselines, we employed the original baseline in our experiments. “F,” “IU,” and “IF” indicate the fluency, ingredient use, and image fitting, respectively. (a) Evaluation with recipes randomly selected from the test set. (b) Evaluation with the longest 25% recipes in (a).

(a) with all 120 recipes.						(b) with the longest 25%(=30/120) recipes.					
Baseline	Full	Tie	Half	Full	Tie	Baseline	Full	Tie	Half	Full	Tie
F	26.7	62.5	10.8	F	35.8	55.8	8.3	F	23.3	66.7	10.0
IU	30.8	66.7	2.5	IU	24.2	70.8	5.0	IU	16.7	83.3	0
IF	31.7	45.0	23.3	IF	21.7	59.2	19.2	IF	10.0	86.7	3.3

exactly the same. Thus, for a fair comparison, we added two enhanced baselines by changing the number of parameters in the biLSTM encoder: Baseline (wide) and Baseline (deep).

- **Baseline (wide)** widely changes the hidden size H by fixing the number of layers $L = 1$ in the encoder biLSTM.
- **Baseline (deep)** deeply increases the number of layers $L = 4$ and changes the hidden size H in the encoder biLSTM.

Based on these strategies, we changed the total number of parameters near to that in the full models. Detailed settings are described in Appendix B.

C. IMPACT ON PROCEDURAL TEXT GENERATION

Scores. We computed the generated recipes with commonly used metrics: word-overlap metrics (BLEU1/4 [41] and ROUGE-L [42]) and diversity metrics (Distinct1/2 [43]: percentage of distinct unigrams/bigrams). However, these metrics do not evaluate the coherency of generated recipes, the global ordering of instructions grounded in the real world. Therefore, as with [19], we also report word-overlap metrics on the extracted sequence of actions and ingredients described in the recipe. Each word sequence extracted from a recipe depict a simulated world where actions are performed and ingredients are used. Thus, the orderings of actions and

ingredients in the generated recipes should agree with those in the ground truth.

To extract the action and ingredient sequences from recipes, we trained the named entity recognizer, Flair⁶ [44] with the recipe flow graph corpus [5], in which the action and ingredient words are defined⁷. We applied the trained model to both the generated recipes and ground truth, and calculated action- and ingredient-level BLEU n ($n = 1, 2, 3, 4$) and ROUGE-L, respectively.

Results. Table 4 shows that our full model achieves steady improvements from the original and enhanced baselines in both of word-overlap metrics and diversity metrics. From this result, we can say that the supervision of the merging tree has contributed to increasing both accuracy and diversity. Moreover, in the word-overlap metrics on the action and ingredient sequences, the proposed models, particularly the full models, substantially outperform ($p < 0.001$) both the original and enhanced baselines in most metrics. This indicates that incorporating a merging tree gains the global coherency in a versatile manner, and re-predicting a merging

⁶<https://github.com/flairNLP/flair>

⁷We trained the recipe NER with 2,072, 230, and 256 sentences for training, validation, and test, respectively. The obtained model scored F1 values of 91.8 and 94.7 for the action and ingredient word recognition, respectively.

Ingredients	Burdock, Carrot, • Sugar, • Sake, • Soy sauce, • Wasabi, White sesame, Sesame oil					
Images						
Models	Step 1	Step 2	Step 3	Step 4	Step 5	Step 6
RetAttn (baseline)	Mix all ingredients.	Chop the burdock into thin strips, and put them in water.	Slice the carrot.	Pour the sesame oil into the pan, and stir-fry the carrot.	Add the carrots and stir-fry them.	After the carrot becomes wilted, add the hijiki and stir-fry it.
RetAttn (half model)	Prepare ingredients.	Chop the burdock into thin strips, and put them in water.	Cut the carrot into fine strips.	After preheating the pan, stir-fry the carrot.	After the sprout, add the carrots and stir-fry them.	Season with salt and pepper.
RetAttn (full model)	Prepare ingredients, and mix seasonings marked • in the ingredient list.	Chop the burdock into thin strips, and put them in water.	Cut the carrot into fine strips.	Pour the sesame oil into the pan, and stir-fry the carrot.	After the carrot becomes wilted, add the burdock and stir-fry it.	Add seasonings and mix them. Serve on a plate.
Ground Truth	Mix seasonings marked • in the ingredient list.	Peel the burdock, chop it into 5-cm strips, and put them in water.	Cut the carrot into fine strips.	Pour the sesame oil into the pan, and stir-fry the burdock. Add carrots and stir-fry them.	Add step 1 seasonings into the pan, and stir-fry them.	Turn off the heat, and add white sesame. Serve on a plate.

Merging tree generated by half model							Merging tree generated by full model						
	Step 1	Step 2	Step 3	Step 4	Step 5	Step 6		Step 1	Step 2	Step 3	Step 4	Step 5	Step 6
• Sugar	●	●	●	●	●	●	• Sugar	●	●	●	●	●	●
• Sake	●	●	●	●	●	●	• Sake	●	●	●	●	●	●
Burdock	●	●	●	●	●	●	• Wasabi	●	●	●	●	●	●
Carrot	●	●	●	●	●	●	• Soy sauce	●	●	●	●	●	●
• Soy sauce	●	●	●	●	●	●	Burdock	●	●	●	●	●	●
• Wasabi	●	●	●	●	●	●	Carrot	●	●	●	●	●	●
White sesame	●	●	●	●	●	●	Sesame oil	●	●	●	●	●	●
Sesame oil	●	●	●	●	●	●	White sesame	●	●	●	●	●	●

FIGURE 4. Example of generated recipes and merging trees. Here, the baseline (original), half, and full models are compared with the ground truth. This sample has no ground truth of the merging tree because it belongs to the Cookpad Image Dataset, not to vSIMMR. Note that the recipes are originally in Japanese and have been translated into English (Japanese version is shown in appendix D).

TABLE 6. Merging tree accuracy.

	Material-to-step	Step-to-step	Total
Baseline			
Merging tree only	70.1	90.0	80.3
Half models			
Images2seq	73.2	90.7	81.5
GLAC Net	72.1	91.4	81.2
SSiD	73.3	91.0	81.6
SSiL	73.1	90.6	81.3
RetAttn	73.1	90.4	81.3
Full models			
Images2seq	72.9	90.0	81.0
GLAC Net	73.7	90.5	81.7
SSiD	72.8	90.7	81.3
SSiL	73.8	90.5	81.7
RetAttn	73.0	91.0	81.5

tree further facilitates generated recipes to be more coherent.

D. HUMAN EVALUATION

To confirm the impact of the increase in the word-overlap metrics for humans, we conducted a human evaluation on 120 recipes randomly selected from the test set. We evaluated our model on the three aspects of recipe quality as examined in [1], [19]: fluency (F), ingredient use (IU), and image fitting (IF).

Settings. For each example, an annotator was asked to compare a pair of recipes, each generated from an identical image sequence but by a different model. The annotator was then asked to select the better recipe according to the three criteria above. For the ingredient use, the annotator selected the recipe that used the ingredients more correctly. For image fitting, we asked the annotators to select the recipe that was more suitable for the image sequence. For the image-sequence-to-recipe model, we used RetAttn, whose base model performed best in Table 4.

Results. Table 5 lists the results, showing the clear advantage of the proposed method over both the baseline and half

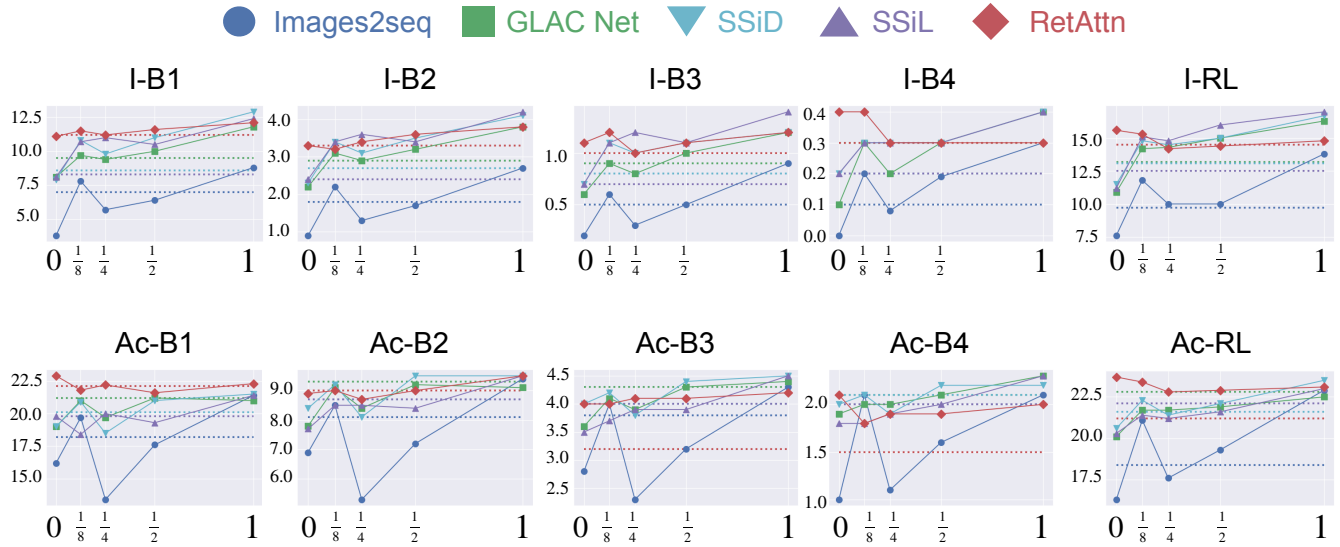


FIGURE 5. (Best viewed in color). Change in action- and ingredient-level word-overlap metrics with a controlled ratio of labeled data. Dotted horizontal lines show that the performance of the baseline (original) shown in Table 4. Note that the entire labeled data accounted for approximately 1% of the training set.

model in all three aspects. To demonstrate the effect of long context recipes, we further evaluate the proposed methods on the longest 25% ($=30/120$) of the recipes⁸. As a result, the superiority of the proposed method is highlighted more strongly. This reveals that re-predicting merging trees helped maintain the global coherency of long recipes.

E. QUALITATIVE ANALYSIS

Figure 4 shows an example of the generated recipes. Other examples are discussed in appendix D.

Insights. Maintaining global coherency is essential particularly for understanding the procedural text. The baseline model loses its maintenance because the model tends to generate unsuitable duplicate actions (e.g., “stir-fry the carrot” in steps 4 and 5). These repetitive phrases are suppressed in the recipe generated by the proposed model. In addition, owing to a merging tree estimation, it is remarkable that the full model generates a referred expression (e.g., the “•” in the recipe generated by the full model in step 1 in the figure), which refers to specific ingredients marked in the ingredient list.

Limitations. Although the estimated merging tree is completely correct, we still found some slight differences with the ground truth. For example, in step 4, the model only adds carrots, but burdock and carrots are added by the ground truth. In step 5, the model does not talk about adding seasonings, but they are added in step 6. This is likely occurred because of the nature of user-generated recipes; the image-step alignment is not regularized and fluctuations are likely. The image in step 5 is not typical in the sense that the image is not a result of the instruction step (the seasoning is added but not stir-

fried as the instruction). Although it is beyond the scope of this paper, a more strict image-step alignment in the dataset would improve the quality of the recipe generation.

F. MERGING TREE EVALUATION

Table 6 shows an evaluation of the tree prediction accuracy. Here, the baseline only estimated the tree and not the recipe, and the model was trained with the pairs of image sequences and merging trees in the training set of vSIMMR. From these results, we can see that the procedural text generation contributes to predicting the merging tree. When comparing the half model with the full model, we can also see that the best score in every metrics is achieved by the full models. This indicates that the tree re-prediction module will guide the model to estimate the merging tree correctly.

G. SATISFACTORY NUMBER OF LABELED DATA ITEMS

To estimate the satisfactory number of labeled data items, we controlled the ratio of labeled data to the values of 0, $\frac{1}{8}$, $\frac{1}{4}$, $\frac{1}{2}$, and 1. As evaluation metrics, we use the action- and ingredient-level BLEU n ($n = 1, 2, 3, 4$) and ROUGE-L to indicate the tendency of controlling training labeled data items. Note that, when the ratio is 0, the model is trained with the proposed architecture in an unsupervised manner. Figure 5 shows the results. In general, most models achieve a higher performance as the ratio of labeled data increases. This shows that training the labeled data in a semi-supervised manner leads to an increase in the quality of the recipe generation. By contrast, interestingly, RetAttn achieves a high performance without any labeled data, indicating that the proposed architecture itself enables RetAttn to generate a coherent recipe. Although the GLAC Net, SSiD, and SSiL require using from 25% to 50% of the labeled data to perform

⁸The length of a recipe is based on the two criteria: the number of steps as the primal criterion and the number of words in a recipe following it.

better than the baselines, Images2seq must use 100% of the labeled data. This result indicates that models with high complexity (GLAC Net, SSiD, SSiL) can learn from less labeled data than the simple model (Images2seq).

VI. CONCLUSION

In this paper, we investigated the impact of explicitly introducing the context dependency on the task of procedural text generation from an image sequence. To achieve this goal, we proposed the vSIMMR dataset by annotating merging trees to a subset of the Cookpad Image Dataset. Using the vSIMMR dataset, we then proposed the structure-aware procedural text generation model, which gains the global coherency by explicitly predicting a merging tree. Additionally, we added a tree re-prediction module to facilitate the generated procedural texts to be more coherent, motivated by the assumption that the structure should be common regardless of the visual and linguistic representations.

We tested the proposed method in the cooking domain, and the results showed that the proposed method can boost the performance of the traditional models in a versatile way and for various metrics. Moreover, a qualitative analysis demonstrated that the model generates a coherent recipe by understanding the context dependency efficiently.

APPENDIX A CHILD-SUM TREE-LSTM CALCULATION

When a child is the m -th material z_{mat}^m , Child-sum Tree-LSTM calculates c^m and h^m using trainable linear functions $f_c : z_{mat}^m \rightarrow c_{mat}^m$ and $f_h : z_{mat}^m \rightarrow h_{mat}^m$, respectively. On the other hand, when a child is n -th step node, it always outputs c^n and h^n .

Let $C(n)$ denote the set of children of node n . Let $W^{(*)}$ and $U^{(*)}$ be the weighted matrices, and $b^{(*)}$ be the bias. The notations $\sigma(\cdot)$ and $\tanh(\cdot)$ mean the sigmoid function and the tangent hyperbolic function, respectively. Then, the Child-sum Tree-LSTM is described as follows:

$$\tilde{h}^n = \sum_{j \in C(n)} h_j^n \quad (7)$$

$$i^n = \sigma(W^{(i)} z_{v,n}^{max} + U^{(i)} \tilde{h}^n + b^{(i)}) \quad (8)$$

$$f_j^n = \sigma(W^{(f)} z_{v,n}^{max} + U^{(f)} \tilde{h}^n + b^{(f)}) \quad (9)$$

$$o^n = \sigma(W^{(o)} z_{v,n}^{max} + U^{(o)} \tilde{h}^n + b^{(o)}) \quad (10)$$

$$u^n = \tanh(W^{(u)} z_{v,n}^{max} + U^{(u)} \tilde{h}^n + b^{(u)}) \quad (11)$$

$$c^n = i^n \odot u^n + \sum_{j \in C(n)} f_j^n \odot c_j \quad (12)$$

$$h^n = o^n \odot \tanh(c^n). \quad (13)$$

Note that the superscript n of the non-leaf nodes is the same as that of the images x_v , and thus, the indices are in a one-to-one correspondence.

TABLE 7. Hyper-parameters we used in our experiments. They are determined by a grid search with the validation set.

	λ_{tc}	λ_{t2i}	λ_{s2i}
Images2seq	0.1	0.01	0.1
GLAC Net	0.01	0.01	0.1
SSiD	0.001	0.1	0.001
SSiL	0.1	0.1	0.001
RetAttn	0.001	0.001	0.01

APPENDIX B RESULTS OF HYPER PARAMETERS TUNING

Hyper-parameter settings. We tuned the hyper-parameters λ_{tc} , λ_{t2i} , and λ_{s2i} for each of the five base models by a grid search with a validation set. Three different parameters, $\lambda_* \in \{0.001, 0.01, 0.1\}$, were used in the experiment. Hence, we searched 27 ($= 3 \times 3 \times 3$) points for each method. Table 7 lists the results of the hyper-parameter tuning.

Trainable parameter settings. As described in Section V-B, the total number of parameters θ_b in the baselines is about 1.5 times smaller than that θ_f in the full models. For a fair comparison, we added the two enhanced baselines, which have θ_b close to θ_f by changing the hidden size H and the number of layers L in the following process. First, we calculated θ_f in the full models. Next, we simply fixed L to 1 (wide) and 4 (deep), and then incremented H by 50 until θ_b is over θ_f . Table 8 shows the hidden size H , the number of layers L , and the total number of trainable parameters θ tuned in our experiments, respectively.

APPENDIX C TEMPERATURE PARAMETER OF GUMBEL SOFTMAX

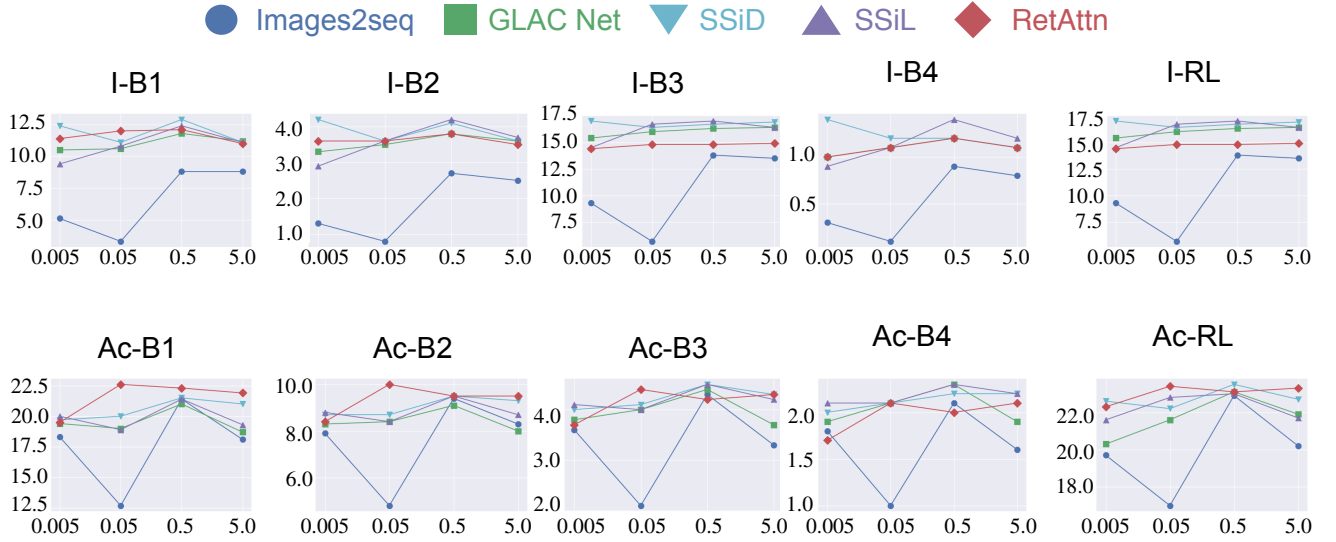
One of the extra hyper-parameters, the Gumbel softmax temperature τ , is known to significantly affect the gradient computation [24], [27], [28]. Here, we explored how the temperature rate affects the performance. Four different temperature rates $\tau \in \{0.005, 0.05, 0.5, 5\}$ were used in this experiment. Figure 6 shows the results for the overlap metrics, indicating that $\tau = 0.5$ tended to achieve the better performance among the candidates. When τ was too low, the model suffered from a gradient vanishing problem, which prevented it from learning to generate a procedural text with a sampled merging tree. By contrast, when τ was too high, it was difficult to stabilize the model. As described in [24], to tackle this problem, we also conducted an experiment on annealing the temperature from high to low at each iteration. This annealing technique did not work in our experiment, however, so we set $\tau = 0.5$ during the training.

APPENDIX D GENERATED RECIPE EXAMPLES

In addition to the example described in the main paper, we show other examples of generated recipes in Figures 7-11, in which different numbers of steps are sampled. Note that the first example (Figure 7) is the Japanese version of the example (Figure 4) described in the main paper. All recipes are originally in Japanese and have been translated into English. As the sequence becomes longer, the baseline model







TABLE 8. The model's parameter settings used in the experiments. H , L , and θ represent the hidden size, the number of layers, and the total number of parameters, respectively.

	Baseline (original)			Baseline (wide)			Baseline (deep)			Full model		
	H	L	θ	H	L	θ	H	L	θ	H	L	θ
Images2seq	512	1	46,460,593	1,600	1	83,800,753	700	4	86,872,753	512	1	82,385,073
GLAC Net	1,024	2	110,312,113	2,250	1	162,487,741	1,000	4	161,438,241	1,024	2	161,284,017
SSiD	1,024	2	110,710,293	2,400	1	162,644,629	1,050	4	165,994,429	1,024	2	161,460,145
SSiL	1,024	2	110,710,293	2,400	1	162,644,629	1,050	4	165,994,429	1,024	2	161,460,145
RetAttn	512	1	109,738,161	1,300	1	147,801,713	650	4	146,137,713	512	1	145,662,641

**FIGURE 6.** Change in the overlap metrics under controlled temperature rates τ .

tends to generate actions in the improper order. For example, in steps 6 and 7 in Figure 11, the baseline model generates “add,” “mix,” and “serve,” whereas the full model generates “crush,” and “add,” which is the same order as in the ground truth. This confirms that the proposed method can consider a long context dependency by incorporating a merging tree estimate into the recipe generator.

Although the proposed method facilitates the generated recipes to be coherent, the quality of the generated recipes can be still improved. For example, in step 2 in Figure 8, the full model says “Add butter to the bowl and heat them with the microwave.” but the butter is not included in the ingredient list. This is because our method does not have any mechanism to explicitly access the vocabulary used in the ingredient list on its decoding process. To address this problem, incorporating a copying mechanism [45] into the decoder may be one of the scenarios of generating words in the ingredient list. However, as stated in Section III-C, recipes and ingredient lists are inconsistent due to the nature of user-generated contents. Thus, it is not clear whether such an additional mechanism works well for solving the problem directly.

Ingredients	ごぼう, にんじん, ● 砂糖, ● 酒, ● しょう油, ● わさび, 白ごま, ごま油					
						
Models	Step 1	Step 2	Step 3	Step 4	Step 5	Step 6
RetAttn (baseline)	材料を全て混ぜます。	ごぼうはさがきに し、水に浸します。	人参は千切りにします。	フライパンにごま油を熱し、人参を炒めます。	人参を入れ炒めます。	人参を炒め終わったらひじきを入れ炒めます。
RetAttn (half model)	材料を用意します。	ごぼうはさがきに し、水に浸します。	人参は千切りにします。	フライパンが温まったら、人参を入れ炒めます。	もやしを入れた後、人参を入れて炒めます。	塩と胡椒で味をつけます。
RetAttn (full model)	材料を用意します。●の調味料を合わせておきます。	ごぼうはさがきに し、水にさらしておきます。	にんじんは千切りにします。	フライパンにごま油を熱し、人参を炒めます。	人参がしんなりしたら、ごぼうを入れて、全体がしんなりするまで炒めます。	全体に火が通ったら、調味料を入れて、よく混ぜます。できあがり！
Ground Truth	●の調味料を混ぜておきます。	ごぼうは皮を剥いて5 cmぐらいの長さで細切りにし、水にさらしておきます。	にんじんを細切りにします。	熱したフライパンにごま油をひいてゴボウを炒めます。ごぼうに火がとおったらにんじんを入れて炒めます。	にんじんに火がとおったら①の調味料を入れて汁気がなくなるまで炒めます。	火を止めて、白ごまをふりかけて混ぜればできあがりです。

Merging tree generated by half model							Merging tree generated by full model						
	Step 1	Step 2	Step 3	Step 4	Step 5	Step 6		Step 1	Step 2	Step 3	Step 4	Step 5	Step 6
● 砂糖	●					●	● 砂糖	●					●
● 酒	●					●	● 酒	●					●
ごぼう	●	●				●	● ごぼう	●	●				●
にんじん	●		●			●	● にんじん	●		●			●
● しょう油	●					●	● しょう油	●					●
● わさび	●					●	ごぼう	●	●				●
白ごま	●					●	にんじん	●		●			●
ごま油	●					●	ごま油	●					●
							白ごま	●					●

FIGURE 7. (Best viewed in color). This is the Japanese version of the example (Figure 4) in the main paper.




Ingredients	Spinach, White sesame, Soy sauce, Sugar		
			
Models	Step 1	Step 2	Step 3
RetAttn (baseline)	Boil the spinach with hot water and drain it. (ほうれん草は塩茹でにし、水気を切っておく。)	Add sugar, sugar, sugar, and salt to the bowl. Mix it. (ボウルに砂糖、砂糖、砂糖を入れてよく混ぜる。)	Add step 1's seasonings to step 1's bowl. (1の調味料を1に入れる。)
RetAttn (half model)	Boil the spinach with hot salted water and Cut it into a bite size. (ほうれん草は、塩茹でして食べやすい大きさに切る。)	Add flour, baking powder, sugar, and cocoa into the bowl. Mix it. (ボウルに薄力粉とベーキングパウダーとココアを入れて混ぜる。)	Add step 1's result to step 2's result, and mix them. (1を2に入れて混ぜる。)
RetAttn (full model)	Boil the spinach with hot salted water and cut it into a bite size. (ほうれん草は、塩茹でして食べやすい長さに切っておく。)	Add butter to the bowl and heat them with the microwave. Add sugar and mix them well. (ボウルにバターを入れ、レンジで加熱し、砂糖を入れよく混ぜる。)	Add step 1's result to step 2's result, and mix them well. (1を2に入れ、よく混ぜる。)
Ground Truth	Wash the spinach and boil it with hot salted water to skim the foam. Drain and cut it into 3-cm pieces. (ほうれん草は水洗いし塩少々を加えたお湯で茹でる。水にさらしてアクを抜き、軽く絞って3cmくらいの大きさに切っておく。)	Add white sesame, soy sauce, and sugar to the bowl. Mix them well. (ボウルにごま・しょうゆ・砂糖を入れてよく混ぜる。)	Add the spinach (step 1), and mix them while loosening them. (水気を切って1のほうれん草を加え、ほぐしながら炒める。)
Merging tree generated by half model			
	Step 1	Step 2	Step 3
Spinach	●	●	●
White sesame	●	●	●
Soy sauce	●	●	●
Sugar	●	●	●
Merging tree generated by full model			
	Step 1	Step 2	Step 3
Spinach	●	●	●
White sesame	●	●	●
Soy sauce	●	●	●
Sugar	●	●	●

FIGURE 8. (Best viewed in color). Example of generated recipes and merging trees. The baseline (original), half, and full models are compared with the ground truth. The half model predicts the merging tree and uses it as the structure of the Child-sum Tree-LSTM. The full model re-predicts the merging tree from the generated recipe in addition to the modules of the half model. Compared with the baseline model, both proposed models (half and full) generate coherent recipes by using the referred expressions (e.g., in step 3, they output "add step 1's result to step 2's result").





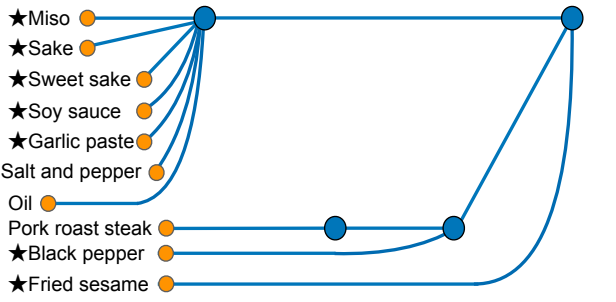
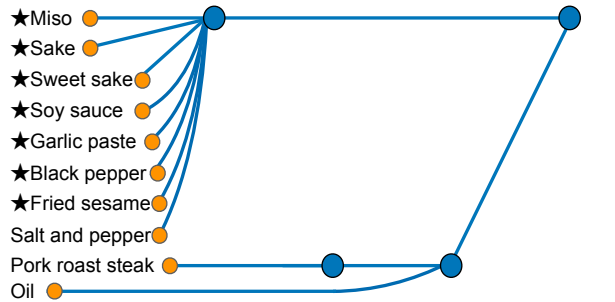







Ingredients	Pork roast steak, Salt and pepper, Oil, ★Miso, ★Sake, ★Sweet sake, ★Soy sauce, ★Garlic paste, ★Black pepper, ★Fried sesame			
				
Models	Step 1	Step 2	Step 3	Step 4
RetAttn (baseline)	Mix all ingredients. (材料を全て混ぜます。)	Cut the pork into bite-size pieces. (豚バラ肉を食べやすい大きさに切ります。)	Pour the oil and grill both sides. (フライパンに油をひき両面焼きます。)	Serve on a plate. (お皿に盛ってできあがりです。)
RetAttn (half model)	Prepare ingredients. (材料を用意します。)	Cut the pork into bite-size pieces. (豚バラ肉を食べやすい大きさに切ります。)	Pour the oil and grill both sides. (フライパンに油をひき、両面を焼きます。)	Serve on a plate. (お皿に盛ってできあがりです。)
RetAttn (full model)	Prepare ingredients. Mix ★ ingredients. (材料を用意します。★の材料を合わせておきます。)	Cut the pork into bite-size pieces. (豚肉は食べやすい大きさに切ります。)	Pour the oil into the pan and grill the pork. (フライパンに油を熱し豚肉を焼きます。)	Put lettuce on a plate. Pour the sauce onto it. It's ready to eat. (レタスを皿に盛り付け、上からタレをかけて完成です。)
Ground Truth	Mix ★ ingredients. (★は合わせておきます。)	Cut the muscle of the pork with the top of the knife. Add the salt and pepper. (豚肉は筋切りをし包丁の先で全体を刺します。塩コショウを軽く振ります。)	Pour the oil, grill the pork (step 2), and add step 1's result to it. Mix them well. (フライパンに油を熱し②を焼き肉に火が通たら①を加え肉に味を絡めます。)	Serve on a plate with your favorite vegetables. (好みの野菜と共に器に盛ります。)
Merging tree generated by half model				
	Step 1	Step 2	Step 3	Step 4
				
Merging tree generated by full model				
	Step 1	Step 2	Step 3	Step 4
				

FIGURE 9. (Best viewed in color). Example of generated recipes and merging trees. The baseline (original), half, and full models are compared with the ground truth. In step 1, the full model correctly generates a referred expression, whereas the other models do not. All models miss the proper timing for adding the seasonings in step 3. We assume that the models would predict the merging tree incorrectly. Although they predict that the seasonings from step 1 would connect with step 4, the seasonings are actually used in step 3.

Ingredients	Egg, Sugar, Chocolate, Milk, Flour, Cocoa, Baking powder, Soy pulp, Drained cherry						
							
Models	Step 1	Step 2	Step 3	Step 4	Step 5	Step 6	Step 7
RetAttn (baseline)	Crack the egg into the bowl. Add sugar and mix well. (ボウルに卵を割り入れ、砂糖を加えてよく混ぜる。)	Add chocolate and mix it. (チョコレートを加えて混ぜる。)	Add milk and mix it well. (牛乳を加えて混ぜる。)	Add flour and mix it well. (ふるった薄力粉を加えて混ぜる。)	Add flour and mix it well. (薄力粉を加えて混ぜる。)	Put dough into a mold. Bake it at 180°C for 25 minutes. (型を流し入れ、180°Cのオーブンで25分焼く。)	After baking, take it out. Bake it at 180°C for 25 minutes. (焼きあがったら、型から取り出し、180度で25分焼く。)
RetAttn (half model)	Crack the egg into the bowl. Add sugar, milk and sugar. Mix well. (ボウルに卵を割りほぐし、砂糖、牛乳、砂糖を入れて混ぜる。)	Add the chocolate to step 2's result. (2にチョコレートをを入れて混ぜる。)	Add milk to step 2's result. (2に牛乳を加え、よく混ぜる。)	Add flour and mix it with the rubber spatula. (薄力粉をふるい入れ、ゴムベラでしっかり混ぜる。)	Add flour and mix it with the rubber spatula. (ふるっておいした薄力粉を加え、ゴムベラでさっくり混ぜる。)	Put dough into a mold. Bake it at 180°C for 25 minutes. (型に生地を入れ、180°Cのオーブンで25分焼く。)	Finish baking. Garnish the cocoa. (焼き上がり。ココアを添えて、完成。)
RetAttn (full model)	Crack the egg into the bowl. Add sugar and mix well. (ボウルに卵を割りほぐし、砂糖を加えてよく混ぜる。)	Melt the chocolate with hot water. Add milk into the bowl. (チョコレートを湯煎で溶かし、牛乳を入れて溶かす。)	Add milk to the step 2's result and mix it with the rubber spatula. (2に牛乳を加え、ゴムベラで混ぜる。)	Add flour and mix it with the rubber spatula. (薄力粉を加えゴムベラでよく混ぜる。)	Add baking powder and mix it well. (ベーキングパウダーをくわえ入れよく混ぜる。)	Put dough into a muffin cup. Put on a drained cherry. Bake them at 180°C for 25 minutes. (マフィン型に生地をいれ、ドランチェリーを乗せる。180°Cのオーブンで25分焼く。)	After baking, cool it in the fridge. Now it's ready for eat. (焼きあがったら、冷蔵庫で冷やして完成。)
Ground Truth	Crack the egg into the bowl. Add sugar and mix well. (ボールに卵を割りほぐし、砂糖を加えて良く混ぜ合わせる。)	Heat milk in the pot. Melt the chocolate with it. (鍋に牛乳を入れて沸騰直前まで温め、刻んだチョコレートのボールに入れて溶かす。)	Add the soy pulp into the step 1's result. Mix it well. (1におからを加え、泡立て器で良く混ぜ合わせる。)	Add step 3's result to step 2's result. Mix them well. (3に2を少しずつ加え、よく混ぜ合わせる。)	Add flour and baking powder to the step 4's result and mix it well. (粉類は合わせてふるっておき、4に加えてよく混ぜる。)	Put dough into a muffin cup. Put on a drained cherry. Bake them at 180°C for 25 minutes. (生地はカップに流し、ドレンチェリーを飾り、180度のオーブンで約25分焼く。)	If it is firm after baking, serve on a plate. (焼き終わったら、触ってみて弾力があればできあがり。)

Merging tree generated by half model							
Step	1	2	3	4	5	6	7
Egg	●	●	●	●	●	●	●
Sugar	●	●	●	●	●	●	●
Chocolate	●	●	●	●	●	●	●
Milk	●	●	●	●	●	●	●
Flour	●	●	●	●	●	●	●
Cocoa	●	●	●	●	●	●	●
Baking powder	●	●	●	●	●	●	●
Soy pulp	●	●	●	●	●	●	●
Drained cherry	●	●	●	●	●	●	●

Merging tree generated by full model							
Step	1	2	3	4	5	6	7
Egg	●	●	●	●	●	●	●
Sugar	●	●	●	●	●	●	●
Chocolate	●	●	●	●	●	●	●
Milk	●	●	●	●	●	●	●
Flour	●	●	●	●	●	●	●
Cocoa	●	●	●	●	●	●	●
Baking powder	●	●	●	●	●	●	●
Soy pulp	●	●	●	●	●	●	●
Drained cherry	●	●	●	●	●	●	●

FIGURE 10. (Best viewed in color). Example of generated recipes and merging trees. The baseline (original), half, and full models are compared with the ground truth. The baseline improperly generates duplicate actions (e.g., “Add flour” in steps 4 and 5, and “Bake” in steps 6 and 7). This is suppressed by the proposed methods, although they each output duplicate actions once: The half model outputs “Add flour” in steps 4 and 5, and the full model outputs “Add milk” in steps 2 and 3). We also emphasize that the full model can verbalize the correct instruction, “Put on a drained cherry,” by using a merging tree estimate, whereas the baseline and half models do not generate this instruction.

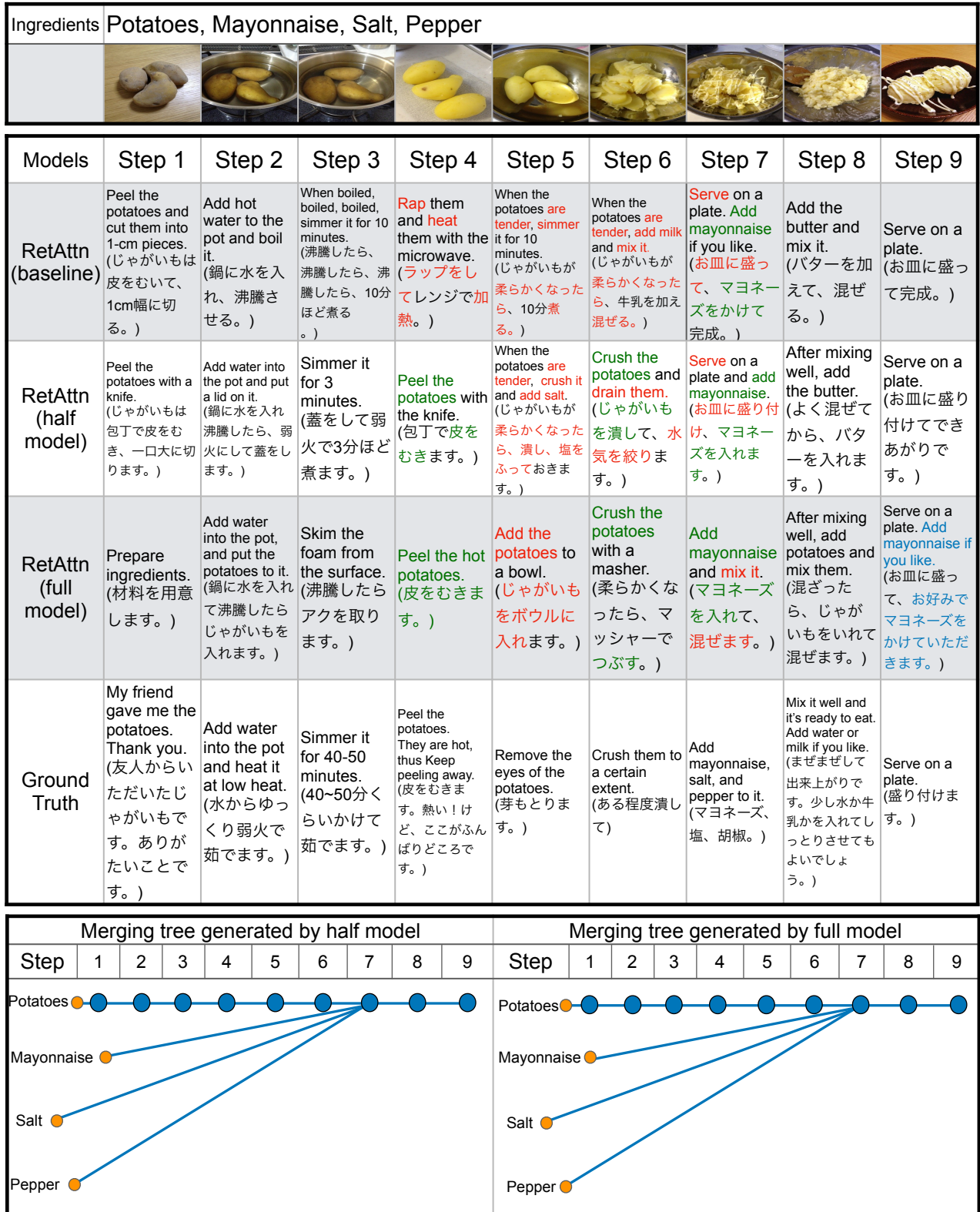


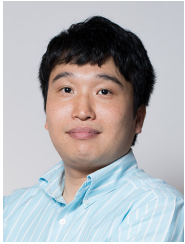
FIGURE 11. (Best viewed in color). Example of generated recipes and merging trees. The baseline (original), half, and full models are compared with the ground truth. This example has a large number of steps. Therefore, generating the recipe correctly requires the models to understand a long context dependency. The baseline model tends to generate improper actions, because it can not understand the long workflow of the recipe (e.g., "rap," "heat," "simmer," "add," "mix," and "serve" from step 4 to step 6). By contrast, the half and full models generate the recipe with the actions in the correct order, to a certain extent. Comparing these models with each other, the full model generates actions suitable for the images, whereas the half model generates duplicate actions (e.g., "crush" in steps 5 and 6). Moreover, the full model generates the instruction, "Add mayonnaise if you like," which is suited to the image but does not appear in the ground truth. This is achieved because the model fully uses the image feature.

REFERENCES

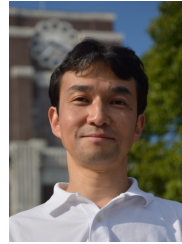
- [1] K. Chandu, E. Nyberg, and A. W. Black, "Storyboarding of recipes: Grounded contextual generation," in *Proc. ACL*, 2019, pp. 6040–6046.
- [2] T. Nishimura, A. Hashimoto, and S. Mori, "Procedural text generation from a photo sequence," in *Proc. INLG*, 2019, pp. 409–414.
- [3] J. Jermisurawong and N. Habash, "Predicting the structure of cooking recipes," in *Proc. EMNLP*, 2015, pp. 781–786.
- [4] C. Kiddon, G. T. Ponnuraj, L. Zettlemoyer, and Y. Choi, "Mise en Place: Unsupervised interpretation of instructional recipes," in *Proc. EMNLP*, 2015, pp. 982–992.
- [5] S. Mori, H. Maeta, Y. Yamakata, and T. Sasada, "Flow graph corpus from recipe texts," in *Proc. LREC*, 2014, pp. 2370–2377.
- [6] H. Kuehne, A. Arslan, and T. Serre, "The language of actions: Recovering the syntax and semantics of goal-directed human activities," in *Proc. CVPR*, 2014, pp. 780–787.
- [7] D. Damen, H. Doughy, G. M. Farinella, S. Fidler, A. Furnari, E. Kazakos, D. Moltisanti, J. Munro, T. Perrett, W. Price, and M. Wray, "Scaling egocentric vision: The EPIC-KITCHENS dataset," in *Proc. ECCV*, 2018, pp. 720–736.
- [8] L. Zhou, C. Xu, and J. J. Corso, "Towards automatic learning of procedures from web instructional videos," in *Proc. AAAI*, 2018, pp. 7590–7598.
- [9] S. Yagcioglu, A. Erdem, E. Erdem, and N. Ikizler-Cinbis, "RecipeQA: A challenge dataset for multimodal comprehension of cooking recipes," in *Proc. EMNLP*, 2018, pp. 1358–1368.
- [10] J. Harashima, Y. Someya, and Y. Kikuta, "Cookpad image dataset: An image collection as infrastructure for food research," in *Proc. ACM SIGIR*, 2017, pp. 1229–1232.
- [11] L. Pan, J. Chen, J. Wu, S. Liu, C.-W. Ngo, M.-Y. Kan, Y.-G. Jiang, and T.-S. Chua, "Multi-modal cooking workflow construction for food recipes," in *Proc. ACMMM*, 2020, pp. 1132–1141.
- [12] D. Zhukov, J.-B. Alayrac, R. G. Cinbis, D. Fouhey, I. Laptev, and J. Sivic, "Cross-task weakly supervised learning from instructional videos," in *Proc. CVPR*, 2019, pp. 3537–3545.
- [13] A. Miech, D. Zhukov, J.-B. Alayrac, M. Tapaswi, I. Laptev, and J. Sivic, "HowTo100M: Learning a text-video embedding by watching hundred million narrated video clips," in *Proc. ICCV*, 2019, pp. 2630–2640.
- [14] J.-B. Alayrac, P. Bojanowski, N. Agrawal, J. Sivic, I. Laptev, and S. Lacoste-Julien, "Unsupervised learning from narrated instruction videos," in *Proc. CVPR*, 2016, pp. 4575–4583.
- [15] Y. Momouchi, "Control structures for actions in procedural texts and pt-chart," in *Proc. COLING*, 1980, pp. 108–114.
- [16] D.-A. Huang, J. J. Lim, L. Fei-Fei, and J. C. Niebles, "Unsupervised visual-linguistic reference resolution in instructional videos," in *Proc. CVPR*, 2017, pp. 2183–2192.
- [17] D.-A. Huang, S. Nair, D. Xu, Y. Zhu, A. Garg, L. Fei-Fei, S. Savarese, and J. C. Niebles, "Neural task graphs: Generalizing to unseen tasks from a single video demonstration," in *Proc. CVPR*, 2019, pp. 8565–8574.
- [18] C. Kiddon, L. Zettlemoyer, and Y. Choi, "Globally coherent text generation with neural checklist models," in *Proc. EMNLP*, 2016, pp. 329–339.
- [19] A. Bosselut, A. Celikyilmaz, X. He, J. Gao, P.-S. Huang, and Y. Choi, "Discourse-aware neural rewards for coherent text generation," in *Proc. NAACL-HLT*, 2018, pp. 173–184.
- [20] A. Salvador, M. Drozdal, X. Giro-i-Nieto, and A. Romero, "Inverse Cooking: recipe generation from food images," in *Proc. CVPR*, 2019, pp. 10453–10462.
- [21] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. NeurIPS*, 2017, pp. 5998–6008.
- [22] H. Wang, G. Lin, S. C. H. Hoi, and C. Miao, "Structure-aware generation network for recipe generation from images," in *Proc. ECCV*, 2020.
- [23] J. Harashima and Y. Yamada, "Two-step validation in character-based ingredient normalization," in *Proc. CEA/MADiMa*, 2018, pp. 29–32.
- [24] E. Jang, S. Gu, and B. Poole, "Categorical reparametrization with gumbel-softmax," in *Proc. ICLR*, 2017.
- [25] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *NeurIPS*, 2013, pp. 3111–3119.
- [26] A. Salvador, N. Hynes, Y. Aytar, J. Marin, F. Ofli, I. Weber, and A. Torralba, "Learning cross-modal embeddings for cooking recipes and food images," in *Proc. CVPR*, 2017, pp. 3020–3028.
- [27] C. Baziotis, I. Androutsopoulos, I. Konstas, and A. Potamianos, "SEQ³: Differentiable sequence-to-sequence-to-sequence autoencoder for unsupervised abstractive sentence compression," in *Proc. NAACL-HLT*, 2019, pp. 673–681.
- [28] J. Gu, D. J. Im, and V. O. Li, "Neural machine translation with gumbel-greedy decoding," in *Proc. AAAI*, 2018, pp. 5125–5132.
- [29] K. S. Tai, R. Socher, and C. D. Manning, "Improved semantic representations from tree-structured long short-term memory networks," in *Proc. ACL-IJCNLP*, 2015, pp. 1556–1566.
- [30] D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling, "Semi-supervised learning with deep generative models," in *NeurIPS*, 2014, pp. 3581–3589.
- [31] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.
- [32] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, "βVAE: Learning basic visual concepts with a constrained variational framework," in *Proc. ICLR*, 2017.
- [33] P. Koehn, "Statistical significance tests for machine translation evaluation," in *Proc. EMNLP*, 2004, pp. 388–395.
- [34] G. Neubig, Y. Nakata, and S. Mori, "Pointwise prediction for robust, adaptable japanese morphological analysis," in *Proc. ACL*, 2011, pp. 529–533.
- [35] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. CVPR*, 2016, pp. 770–778.
- [36] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. CVPR*, 2009, pp. 248–255.
- [37] T. K. Huang, F. Ferraro, N. Mostafazadeh, I. Misra, A. Agrawal, J. Devlin, R. Girshick, X. He, P. Kohli, D. Batra, C. L. Zitnick, D. Parikh, L. Vanderwende, M. Galley, and M. Mitchell, "Visual storytelling," in *Proc. NAACL-HLT*, 2016, pp. 1233–1239.
- [38] T. Kim, M. Heo, S. Son, K. Park, and B. Zhang, "GLAC net: Global attention cascading networks for multi-image cued story generation," *arXiv preprint arXiv:1805.10973*, 2018.
- [39] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR*, 2015.
- [40] P. Matteo, P. Gupta, and M. Jaggi, "Unsupervised learning of sentence embeddings using compositional n-gram features," in *Proc. NAACL-HLT*, 2018, pp. 528–540.
- [41] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "BLEU: A method for automatic evaluation of machine translation," in *Proc. ACL*, 2002, pp. 311–318.
- [42] C.-Y. Lin and F. J. Och, "Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics," in *Proc. ACL*, 2004, pp. 605–612.
- [43] J. Li, M. Galley, C. Brockett, J. Gao, and B. Dolan, "A diversity-promoting objective function for neural conversation models," in *Proc. NAACL-HLT*, 2016, pp. 110–119.
- [44] A. Akbik, D. Blythe, and R. Vollgraf, "Contextual string embeddings for sequence labeling," in *Proc. COLING*, 2018, pp. 1638–1649.
- [45] J. Gu, Z. Lu, H. Li, and V. O. Li, "Incorporating copying mechanism in sequence-to-sequence learning," in *Proc. ACL*, 2016, pp. 1631–1640.



TAICHI NISHIMURA received the B.S. degree in engineering from Kyushu University in 2019, and the M.S. degree in informatics from Kyoto University in 2020. His research area is vision-and-language, especially text generation systems from images or videos. He received the Best Paper Award at the 11th workshop on multimedia for cooking and eating activities in 2019.

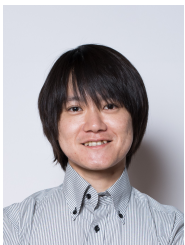


ATSUSHI HASHIMOTO received the B.S. degree in engineering, the M.S. and Ph.D. degrees in informatics from Graduate School of Informatics, Kyoto University, Japan, in 2005, 2008, and 2013, respectively. He is currently a senior researcher with Omron SINIC X corp. from 2018. He has been working on human activity analysis and recognition based on human-object interaction, and interface design and development in the field of cooking. His research interests include pattern recognition, computer vision, and human-computer interaction. He is a member of the IEEE, Institute of Electronics, Information and Communication Engineers of Japan, and Information Processing Society of Japan.



SHINSUKE MORI received B.S., M.S., and Ph.D. degrees in electrical engineering from Kyoto University, Kyoto, Japan in 1993, 1995, and 1998, respectively. Then he joined Tokyo Research Laboratory of International Business Machines Co. Ltd. (IBM). Since 2007 he was an associate professor of Academic Center for Computing and Media Studies, Kyoto University. He is currently a professor there. His research interests include computational linguistics and natural language processing. He received IPSJ Yamashita SIG Research Award in 1997, IPSJ Best Paper Award in 2010 and 2013, and 58th OHM Technology Award from Promotion Foundation for Electrical Science and Engineering in 2010. He is a member of Information Processing Society of Japan, the Association for Natural Language Processing, the Database Society of Japan, and the Association for Computational Linguistics.

...



YOSHITAKA USHIKU received his B.E., M.A., and Ph.D. degrees from the University of Tokyo in 2009, 2011, and 2014, respectively. In 2014, he joined NTT CS Labs, Japan, where he was involved in research on image recognition. From 2016 to 2018, he was a lecturer with the University of Tokyo, Japan. Currently, he is a Principal Investigator at OMRON SINIC X and Chief Research Officer at Ridge-i since 2018 and 2019, respectively. His research interests lie in cross-media understanding through machine learning, mainly for computer vision and natural language processing. He received ACM Multimedia Grand Challenge Special Prize in 2011, ACM Multimedia Open Source Software Competition Honorable Mention in 2017, and NVIDIA Pioneering Research Awards in 2017 and 2018.



HIROTAKA KAMEKO received his B.E., M.E., and Ph.D. degrees from the University of Tokyo in 2013, 2015 and 2018, respectively. He is currently an assistant professor of Academic Center for Computing and Media Studies, Kyoto University. His research interests include natural language processing and game AI. He is a member of IPSJ and ANLP.



YOKO YAMAKATA received her Ph.D. degree in Informatics from Kyoto University, Japan in 2006. She was an expert researcher of NICT, Japan from 2007, a lecturer and an associate professor of Kyoto University from 2010, a JSPS research fellow at the University of Tokyo from 2015, an international visiting researcher of the University of Sussex, UK between 2016-2017 and currently an associate professor of the University of Tokyo, Japan. Her research interests include multimedia processing especially regarding cooking and eating activities.