Procedural Text Generation from an Execution Video

Atsushi Ushiku Hayato Hashimoto Atsushi Hashimoto Shinsuke Mori
Yoshida-honmachi, Sakyo-ku,
Kyoto University, Kyoto, Japan
d460655046504@gmail.com hashimoto.hayato.73e@st.kyoto-u.ac.jp
ahasimoto@mm.media.kyoto-u.ac.jp forest@i.kyoto-u.ac.jp

Abstract

In recent years, there has been a surge of interest in automatically describing images or videos in a natural language. These descriptions are useful for image/video search, etc. In this paper, we focus on procedure execution videos, in which a human makes or repairs something and propose a method for generating procedural texts from them. Since available video/text pairs are limited in size, the direct application of end-to-end deep learning is not feasible. Thus we propose to train Faster R-CNN network for object recognition and LSTM for text generation and combine them at run time. We took pairs of recipe and cooking video as an example, generated a recipe from a video, and compared it with the original recipe. The experimental results showed that our method can produce a recipe as accurate as the state-of-the-art scene descriptions.

1 Introduction

Massive effort has been done to develop a method for generating text from vision in the field of natural language processing and computer vision. More specifically, there are number of studies on generating captions for given images or videos (Yang et al., 2011; Rohrbach et al., 2013; Li et al., 2015; Donahue et al., 2015; Karpathy and Fei-Fei, 2015; Shetty and Laaksonen, 2016; Johnson et al., 2016). Most of the existing researches for video captioning, however, deal with simple and short videos (Li et al., 2015; Donahue et al., 2015; Shetty and Laaksonen, 2016) such as a ten second video in which a man playing guitar in a park.

In this paper, we propose a new problem in this field: generating a procedural text from an execution video such as cooking or machine assembly. The goal is to develop a method that takes video of a chef cooking a dish from ingredients or a mechanic assembling a machine from parts as the input, and outputs a procedural text that helps another person reproduce the same product.

We also give an initial solution to the problem, taking cooking recipe generation as an example. Because no large scale corpus consisting of related execution video and procedural text is available for now, we divide the problem into two subproblems, object recognition and text generation, and train two modules independently using different resources as their training set. Then we combine them and search for the best text. The object recognition module is designed to spot the changes in state of progress of the procedure from video and texts are generated at each time. Finally, some of the generated sentences are selected to cover the entire procedure with discarding redundant sentences.

In the experiments, we use KUSK Dataset (Hashimoto et al., 2014), which consists of pairs of recipes submitted by users to a recipe hosting service Cookpad and video of cooking according to that recipe in a laboratory. The experimental results showed that our method is capable of producing a recipe of reasonable quality.

2 Related Work

Recent studies on automatic caption generation have reported great results both in images (Xu et al., 2015; Karpathy and Fei-Fei, 2015; Johnson et al., 2016) and short video clips (Li et al., 2015; Donahue et al., 2015; Shetty and Laaksonen, 2016) by using convolutional neural network (CNN), recurrent neural network, and LSTM. (Venugopalan et al., 2015) improved the accuracy with a sequence to sequence model (Sutskever et al., 2014). In addition, (Laokulrat et al., 2016; Guo et al., 2016) also improved the accuracy of automatic caption generation by introducing an LSTM equipped with an attention mechanism. One of the features of these end-to-end models is that they directly generate sentences from videos without determining content words such as subjects and predicates. Common datasets (Lin et al., 2014; Chen and Dolan, 2011; Torabi et al., 2015; Rohrbach et al., 2015) made research on automatic caption generation popular.

Before the above end-to-end models succeeded, many researchers concentrated models generating sentences via content words or intermediate states (Guadarrama et al., 2013; Rohrbach et al., 2013). As an advantage of the technique of using intermediate states, object recognition or motion recognition model can be diverted as it is. Thus data of pairs of a medium and a caption have not been particularly required. These methods with intermediate states are inferior in accuracy to the end-to-end models using CNN and LSTM in case that enough size of training data are available. On the other hand, since creation of medium-caption pairs is expensive, methods using intermediate states are also considered to be sufficiently practical for a problem where we have insufficient size of data available for model training.

Unlike conventional methods using intermediate states such as subjects, objects, and predicates, for procedure execution videos, there is a problem that the use of recognition results of general actions is not appropriate because of the abstraction level. It is considered preparing tailored data for motion recognition for each kind of procedure execution videos have high cost because it is often vague even for human annotators to assign every concrete motions into text-level motion categories. In contrast, objects directly appear in texts and there is much less ambiguity than motions. Therefore, it is reasonable for the procedural text generation to focus more on object recognition than motion recognition. In addition, the procedure execution videos generally show works performed by one person, thus subject recognition is not necessary. It is preferable to set the object recognition results as an intermediate state and generate sentences from it. Since predicates are not easy to be recognized, they are estimated or supplemented from recognized objects using language knowledge.

Many studies generate a caption consisting of one sentence for a video clip. Studies on the automatic caption generation of documents consisting of multiple sentences like procedural text do not attract much attention as far as we know. One similar study is done by (Kaufman et al., 2016), which gives captions for a movie that is divided into scenes beforehand.

Table 1: Definition of r-NE tags.

r-NE tag	meaning
F	Food
T	Tool
D	Duration
Q	Quantity
Ac	Action by the chef
Af	Action by foods
Sf	State of foods
St	State of tools

3 Task Definition

In this section, we describe our novel task in detail. Then we present prerequisites of our solution.

3.1 Procedural Text Generation from Video

We propose a task of generating a procedural text from an execution video. Figure 1 shows the overview of the task. In general, an execution video records a sequence of activities to make or repair something from the beginning to the end. As the first trial, we deal with cooking videos in which only one person appears (mainly the hands only). In the beginning, there are some ingredients and tools on the cooking table and some appear in the video later. Then it finishes with a completed dish. This is the input of the task.

The output of our task is a procedural text, consisting of some sentences in a natural language, which explains procedures to be conducted by workers to make or repair something. The counterpart of cooking videos of the first trial is recipes. A recipe describes how to cook a certain dish. In general, a recipe includes the dish name and an ingredient list in addition to the instruction text part. In our task, however, we focus on generating the text part only. Thus, this is the output of the task. In the subsequent sections, we refer to that text part by the term recipe.

As an evaluation metrics, it is preferable to measure how much the output text helps another chef produce the same dish. Thus, the ideal may be

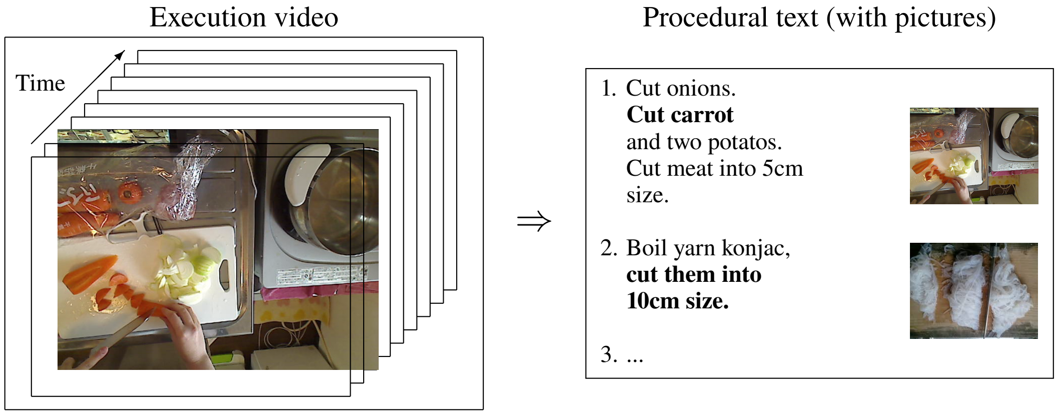


Figure 1: Task overview.

objective evaluation over the dishes produced by chefs reading the generated recipes. We propose, however, to adopt BLEU score as a metric of procedural text for the convenience of automated evaluation.

One of the advantages to choose the cooking domain as a benchmark of procedural text generation from video is that there are a huge number of recipes available on the Web. Therefore it is easy to develop a generative model of recipes for the task. In addition, there are recipe/video pairs available for various researches. For example, the KUSK Dataset (Hashimoto et al., 2014), which we use in the experiments, contains recipes and their cooking videos. Note that the lengths of these cooking videos are about 20 minutes or more, which are much longer than video clips used in automatic video captioning researches. And also note that the texts are kinds of summaries mentioning only the necessary objects and actions to complete a certain mission. Such texts are intrinsically different from scene descriptions in automatic video (or image) captioning researches.

3.2 Prerequisites

To solve the problem above, we enumerate the preconditions necessary for our method in the recipe generation case.

3.2.1 Domain Specific Named Entity

First we assume a set of terms (word sequences) called named entities (x-**NEs**) representing important object names in the target domain x. They are the objects to be recognized by computer vision (CV).

In the recipe case, noun phrases for ingredients and tools are important object names. In this paper, we adopt the recipe named entities (**r-NEs**) defined in (Mori et al., 2014), whose types are listed in Table 1. There are eight r-NE tag types, but our CV part recognizes only foods (F) and tools (T). We use the notation "チンゲン 菜/F" ("qing-geng-cai/F") to indicate that "チンゲン 菜" is an r-NE and its type is food (F)[1].

3.2.2 Named Entity Recognizer

In order to develop a useful generative model we must locate x-NEs in given sentences. So-called named entity recognizer (**NER**) is suitable for this task. In this paper, we adopt NERs based on sequence labeling techniques that can be trained by an annotated corpus.

3.2.3 Object Recognition

Our method requires the module that can detect the appearance and the disappearance of materials and tools involved in the procedure. In the cooking video case, we use Faster R-CNN model (Ren et al., 2015) fine-tuned with relatively small set of images of foods and cooking tools.

3.2.4 Procedural Text Examples

As we mentioned in Section 1, there is no large amount of video/sentence pairs available for our problem. But instead, in some cases, large text-only corpus is available in the domain. The corpus will allow us to train a generative model of the instruction sentences.

4 Proposed Method

In this section, we explain the proposed method for recipe generation from cooking videos. The out-

[1] The language resources used in our experiments are in Japanese. Thus our system outputs recipes in Japanese. However, our method can generate recipes in another language by preparing the prerequisites in that language.

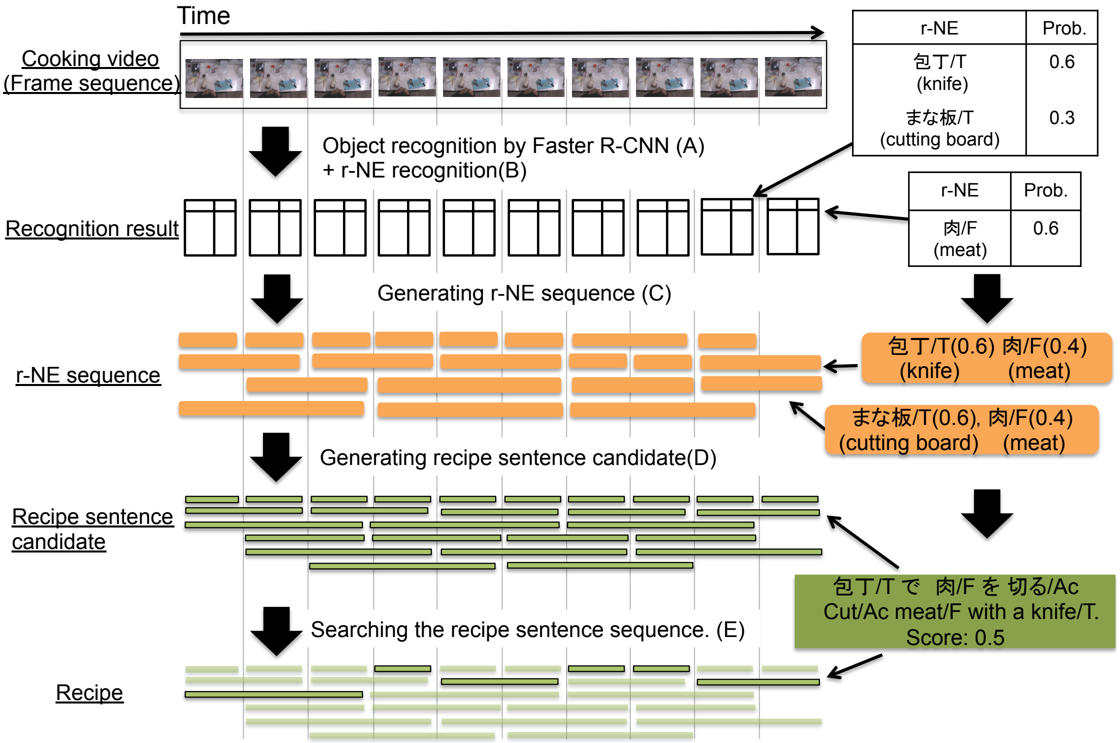

Time
Cooking video (Frame sequence)
Object recognition by Faster R-CNN (A) + r-NE recognition(B)
r-NE
Prob.
包丁/T (knife)
0.6
まな板/T (cutting board)
0.3
Recognition result
r-NE
Prob.
肉/F (meat)
0.6
Generating r-NE sequence (C)
r-NE sequence
包丁/T(0.6) 肉/F(0.4) (knife) (meat)
まな板/T(0.6), 肉/F(0.4) (cutting board) (meat)
Generating recipe sentence candidate(D)
Recipe sentence candidate
包丁/T で 肉/F を 切る/Ac
Cut/Ac meat/F with a knife/T.
Score: 0.5
Searching the recipe sentence sequence. (E)
Recipe

Figure 2: Overview of the proposed method.

line of this method is shown in Figure 2. First, we recognize objects in the video as a sequence of frames with a CNN and give an r-NE tag to each object (Figure 2 A, B). Next, we create an r-NE sequence from each partial frame sequence (Figure 2 C) and generate a candidate recipe sentence for each corresponding r-NE sequence (Figure 2 D). Each candidate recipe sentence is the one which maximizes the score indicating the likelihood of a sentence as a procedural text within the partial frame sequence. Finally, we select the sequence of recipe candidate sequences that maximizes the total score through the entire video based on Viterbi search. We output that sentence sequence as the procedural text for the input procedure execution video (Figure 2 E).

4.1 Object Recognition

Object recognition is performed only on the frames at which the chef picks up an object or places it, that is provided in KUSK Object Dataset (Hashimoto et al., 2016) with the object regions. Note that the provided frames and regions can contain plural objects because the method used in (Hashimoto et al., 2016) is based on background subtraction. To divide the detected region into object-wise regions, we adopted Faster R-CNN (Ren et al., 2015). This neural network outputs identified object region as a rectangular area while recognizing its category (Figure 2 A). It also provides confidence as a probability. An example of visualization of object recognition is shown in Figure 3, where a cutting board and a knife are in the region detected by (Hashimoto et al., 2016) .

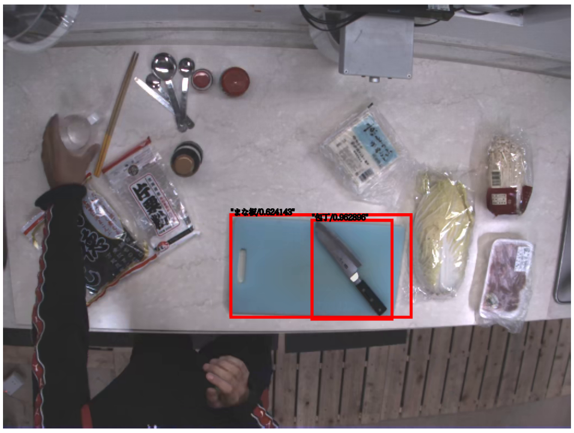


Figure 3: An example of object recognition by Faster R-CNN.

We utilized Faster R-CNN's ability of object region identification to suppress another type of false detection. The regions provided in (Hashimoto et al., 2016) contains objects that are moved only slightly by coming in contact with the hands. Such objects should not be related to the procedure. To suppress such detection but spot only objects obviously related to the procedure, we compare the location of object-wise regions before and after the contact, and ignore object regions if they have the same object name and have a certain score in Jaccard index, which is general method to measure the size of intersection of two regions. After the test of region intersection, only the objects with an obvious location change are regarded as procedure-related. This module passes only procedure-related objects to the second module. Note also that we discarded objects whose name is not listed in x-NEs before passing them to the second module.

Hereafter, we only focus the frames with the procedure-related objects listed in x-NEs, and describe the sequence of such frames as follows:

$$\boldsymbol{f} = f_1,\ f_2,\ \ldots,\ f_{|\boldsymbol{f}|}, \tag{1}$$

where f_i is the i-th frame and $|\boldsymbol{f}|$ is the length of the sequence.

4.2 Recipe Named Entity Recognition

We use the named entity recognizer (Sasada et al., 2015) to the object in the i-th frame f_i (Figure 2 B). Let $\mathcal{E}_i$ be the object set whose tags are F or T in f_i. Then, we denote the number of elements in this set as $|\mathcal{E}_i|$. The j-th r-NE of $\mathcal{E}_i$ is denoted as e_i^j. Then $P(e_i^j|f_i)$ denotes the conditional probability in which the element e_i^j (a food or a tool) is estimated to exist in the frame f_i.

4.3 Recipe Named Entity Sequence

Let $\boldsymbol{f}_i^{i+(l-1)} = f_i, f_{i+1}, ..., f_{i+(l-1)}$ be a substring, of length l, of $\boldsymbol{f}$ that corresponds to a single recipe sentence. A frame f_i may contain some r-NEs $\mathcal{E}_i$. Then a sequence of r-NEs contained in $\boldsymbol{f}_i^{i+(l-1)}$ can be expressed by $\boldsymbol{e} \in \mathcal{E}_i \times \mathcal{E}_{i+1} \times ... \times \mathcal{E}_{i+(l-1)}$. Note that the number of all the possible sequences is $\prod_{k=i}^{i+(l-1)} |\mathcal{E}_k|$. For example in Figure 2, $\boldsymbol{e}$ is (cutting board/T, meat/F) or (knife/T, meat/F).

In addition, in order to treat a sequence as a set, we introduce the following notation:

$$\{\boldsymbol{e}\} = \{e_k^{j_k} | i \leq k \leq i + (l-1)\}. \tag{2}$$

Note that j_k depends on k.

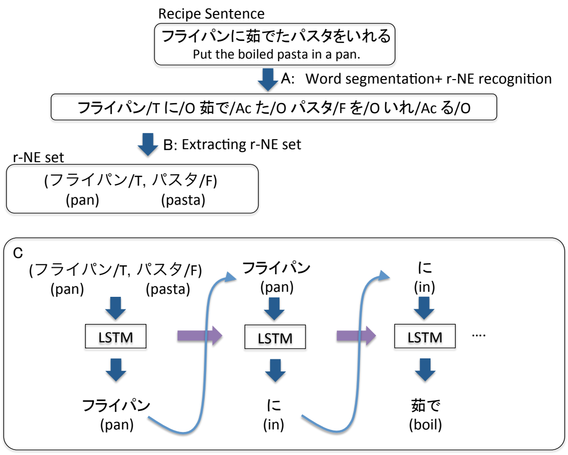


Figure 4: LSTM language model training. This model generates a sentence given an r-NE set.

Considering the likelihood of object recognition and the likelihood of a combination of r-NEs included in the sequence, we set the likelihood $P(\boldsymbol{e})$ that $\boldsymbol{e}$ appears as follows:

$$P(\boldsymbol{e}) = \overline{P}(\boldsymbol{e}) \times P(\{\boldsymbol{e}\})^{-l}, \tag{3}$$

where $\overline{P}(\boldsymbol{e})$ is the average of the probability of the result of object recognition:

$$\overline{P}(\boldsymbol{e}) = \frac{1}{l} \sum_{k=i}^{i+(l-1)} P(e_k^{j_k}|f_k). \tag{4}$$

This value indicates the likelihood of object recognition. Also $P(\{\boldsymbol{e}\})^{-l}$ is the likelihood of a combination of r-NEs determined from the frequency of a sentence in which all the elements of $\{\boldsymbol{e}\}$ appear in the corpus.

$$P(\{\boldsymbol{e}\}) = \left(\frac{count(\{\boldsymbol{e}\})}{C}\right), \tag{5}$$

where C is the number of sentences in the recipe corpus and $count(\{\boldsymbol{e}\})$ is the frequency of sentences in which all the elements of $\{\boldsymbol{e}\}$ appear at the same time. Thus, this value indicates the likelihood of the r-NE combination. In addition as the number of elements in the r-NE set increases, the frequency decreases. This is the reason why we introduce $P(\{\boldsymbol{e}\})^{-l}$ considering the sequence length l.

4.4 Recipe Sentence Candidate Generation

For each partial frame sequence, we generate the most likely sentence and its score without referring to the neighboring sentences. Some of these sentences may, however, be discarded in the next step. Thus we call it a recipe sentence candidate. The input to this process is the r-NE sequence and the scores of the r-NEs. And the output is the recipe sentence candidate that maximizes the score for the given partial frame sequence (Figure 2 D).

For the sentence candidate generation we use an LSTM language model. It outputs a sentence and its likelihood. Different from the ordinary LSTM, it takes a set of r-NEs as the input, but not a sequence. In addition, it is trained on the corpus in which r-NEs are recognized and replaced with r-NE tags as summarized in Figure 4. The first step of its training is preprocessing, in which we conduct word segmentation (Neubig et al., 2011) (not necessary for English or some other languages) and r-NE recognition (Sasada et al., 2015) for each recipe sentence in the recipe corpus (Figure 4 A). Then we filter out sentences containing r-NEs other than Ac, F and T and delete Ac tags for the reasons below:

- We cannot get information about r-NEs other than F and T by the object recognition module.
- A predicate denoting an action (Ac) is necessary for a complete sentence.

Putting it in another way, our method guesses a suitable predicate (verb) from the objects (foods and/or tools) and the corpus. From each of the resultant sentences, we generate a sentence in which F and T are replaced with tags and the set of the r-NEs contained in it (Figure 4 B). Finally we train the LSTM language model on the corpus. The LSTM can map a set of r-NEs to a recipe sentence with its likelihood (Figure 4 C).

As the likelihood of this module, our method returns the following score:

$$Score(\boldsymbol{e}) = P_{\text{LSTM}}(\boldsymbol{r}_{max}(\boldsymbol{e})|\boldsymbol{e}) \times P(\boldsymbol{e}),$$

where $\boldsymbol{r}(\boldsymbol{e})$ is the sentence generated by the LSTM language model given $\boldsymbol{e}$ as the input. $P_{\text{LSTM}}(\boldsymbol{r}(\boldsymbol{e})|\boldsymbol{e})$ is the generation probability of $\boldsymbol{r}(\boldsymbol{e})$. $\boldsymbol{r}_{max}(\boldsymbol{e})$ is the sentence that maximizes $P_{\text{LSTM}}(\boldsymbol{r}(\boldsymbol{e})|\boldsymbol{e})$ with the beam search decoder given $\boldsymbol{e}$.

$$\boldsymbol{r}_{max}(\boldsymbol{e}) = \mathop{\textbf{argmax}}_{\boldsymbol{r}(\boldsymbol{e}) \in \mathcal{R}(\boldsymbol{e})} P_{\text{LSTM}}(\boldsymbol{r}(\boldsymbol{e})|\boldsymbol{e}),$$

where $\mathcal{R}(\boldsymbol{e})$ is a set of sentences that can be generated by beam search when $\boldsymbol{e}$ is the input and $\boldsymbol{r}(\boldsymbol{e})$ is the sentence corresponding to it. The generation probability of a sentence is calculated by the following formula:

$$P_{\text{LSTM}}(\boldsymbol{r}(\boldsymbol{e})|\boldsymbol{e}) = \prod_{k=1}^{N_d} P(d_k|d_1, d_2, ..., d_{k-1}; \boldsymbol{e}),$$

where $\boldsymbol{r}(\boldsymbol{e}) = d_1, d_2, ..., d_{N_d}$ is a word string and N_d is the length of the word string. And $P(d_k|d_1, d_2, ..., d_{k-1}; \boldsymbol{e})$ denotes the generation probability of the k-th word d_k, when the input is $\boldsymbol{e}$. The sentence is generated by the LSTM language model by beam search. The sentence is, however, aborted when the word length exceeds 20 or the terminal symbol appears. $P(\boldsymbol{e})$ is introduced to reflect the likelihood that the r-NE sequences $\boldsymbol{e}$ appear (see Equation (3)).

Calculating the above scores for all the possible $\boldsymbol{e}$ of a partial frame sequence, we define $\boldsymbol{e}_{max}$ as the r-NE sequences which maximize the score. At this stage, the generated sentence is no more than a recipe sentence candidate $\boldsymbol{r}_{max}(\boldsymbol{e}_{max})$, whose score is $Score(\boldsymbol{e}_{max})$. When the scores earned by partial frame sequences are all 0, no recipe sentence candidate is generated.

4.5 Generating Recipe

As we see above, a set of recipe sentence candidates is generated from the partial frame sequences. The frame sequence is divided into partial sequences so that the overall score of the division, which is the sum of the $Score(\boldsymbol{e})$ in each partial sequence, is maximized (Figure 2 E). The partial sequences cover the entire video, thus the corresponding sentences, sequences of $\boldsymbol{r}(\boldsymbol{e})$, form a complete recipe.

Since it is almost impossible for one chef to perform two operations in parallel, the corresponding partial frame sequences of the recipe sentence candidates must not overlap. In addition, in order to prevent the same recipe sentence from appearing more than once, the score of the recipe sentence candidate which has appeared once in the recipe is set to be 0. Under this condition, the score of a recipe sentence candidate can change. Although it should be totally searched for score maximization, we use the Viterbi algorithm for the calculation, because the change of the score is limited at the time of generation of the same sentence and it is considered that it does not occur so much.

By calculating the path of the recipe sentence candidate sequence for increasing the score, the

Table 2: The BLEU scores.

	BLEU			
Configuration	$N = 1$	$N = 2$	$N = 3$	$N = 4$
w/o $P(\{e\})^{-l}$	22.73	13.13	7.48	4.11
with $P(\{e\})^{-l}$	26.73	15.42	9.09	**5.50**

generated recipe sentence sequence is output as a recipe. The higher the score, the more recipe-like the sentences are.

5 Experiments and Evaluation

In this section we evaluate our method experimentally. We first describe the settings of the experiments, then report the experimental results, and finally evaluate our method. [2]

5.1 Experimental Setting

We used the following dataset to train and evaluate our model.

5.1.1 Test Dataset

KUSK Dataset This dataset contains 20 recipes and corresponding cooking videos.

5.1.2 Train Dataset

KUSK Object Dataset This dataset contains 180 categories of objects in total, which comprise ingredients, cooking tools, and others (bottle cap, dish cloth, and so on), observed in cooking videos in KUSK Dataset. Since all videos are recorded at the same kitchen, exactly the same cooking tools appear through all videos, including ones in the test set. More detailed information and examples are available in (Hashimoto et al., 2016).

Cookpad NII corpus This corpus contains 1720000 recipes collected from cookpad website. 187700 sentences are extracted for training.

Flow Graph Corpus This corpus contains randomly chosen 208 recipes (867 sentences) from Cookpad NII corpus. The text is annotated with the r-NE tags. (Mori et al., 2014).

5.1.3 Training Faster-RNN and Named Entity Recognizer

As the first module, an object recognizer for frames, we use Faster R-CNN(Ren et al., 2015). We fine-tuned Faster R-CNN with KUSK Object Dataset. The dataset contains 180 categories in total, but some categories, for example dish clothes or bottle caps, will not appear in recipe texts. Thus we ignored such categories and used 95 categories to fine-tune the Faster R-CNN model, which is done in the manner of leave-one-video-out.

Because this module is a pre-process of the second module, to achieve higher recall rather than a higher precision, we used any detection proposals from Faster R-CNN with more than 0.01% in confidence score, and set the intersection threshold of Jaccard Index 0.5. This setting earned 78.8% of recall and 22.3% of the precision on average through the 95 categories.

For the second module we trained an NE recognizer PWNER (Sasada et al., 2015), which is based on support vector machines and Viterbi best path search, with Flow Graph Corpus. Its accuracy is about 90% in F-measure (Mori et al., 2014).

5.1.4 Recipe Named Entity Sequence and Recipe Sentence Candidate Generation

When generating the r-NE sequences, we should specify the sequence length l. Most of the sentences in our recipe corpus contain no more than three r-NEs of F or T[3]. So we set the length of frame sequences as $l = 1 \sim 3$.

The training data of the LSTM language model consists of 11,705 sentences and the number of r-NE tokens is 4,025. These training data are a set of recipe sentences extracted so as to satisfy the following conditions:

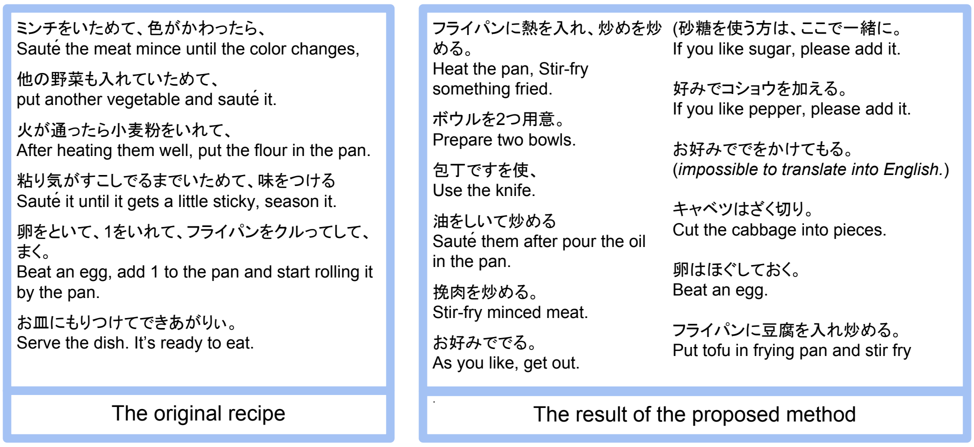
- The total number of F and T is between 1 and 3,
- Each sentence does not contain any r-NE other than Ac, F, and T (see Section 4.4).

As a result the LSTM language model has a tendency not to generate sentences containing 4 or more r-NEs.

The setting of the LSTM language model training is as follows. The epoch number is 100, the batch size is 100, and the number of units of LSTM is 1,000. The objective function is the softmax cross entropy and the optimization algorithm is Adam (Kingma and Ba, 2014). The beam width for recipe sentence candidate generation is set to be 1.

[2] The code used in our experiment is available on our website. http://www.ar.media.kyoto-u.ac.jp/member/hayato/procedural-text-generation/

[3] The percentage is slightly less than 75%.



ミンチをいためて、色がかわったら、
Sauté the meat mince until the color changes,

他の野菜も入れていためて、
put another vegetable and sauté it.

火が通ったら小麦粉をいれて、
After heating them well, put the flour in the pan.

粘り気がすこしでるまでいためて、味をつける
Sauté it until it gets a little sticky, season it.

卵をといて、1をいれて、フライパンをクルってして、まく。
Beat an egg, add 1 to the pan and start rolling it by the pan.

お皿にもりつけてできあがりぃ。
Serve the dish. It's ready to eat.

The original recipe

フライパンに熱を入れ、炒めを炒める。
Heat the pan, Stir-fry something fried.

ボウルを2つ用意。
Prepare two bowls.

包丁ですを使、
Use the knife.

油をしいて炒める
Sauté them after pour the oil in the pan.

挽肉を炒める。
Stir-fry minced meat.

お好みででる。
As you like, get out.

(砂糖を使う方は、ここで一緒に。
If you like sugar, please add it.

好みでコショウを加える。
If you like pepper, please add it.

お好みででをかけてもる。
(*impossible to translate into English.*)

キャベツはざく切り。
Cut the cabbage into pieces.

卵はほぐしておく。
Beat an egg.

フライパンに豆腐を入れ炒める。
Put tofu in frying pan and stir fry

The result of the proposed method

Figure 5: The original recipe for a cooking video and the generated recipe by the proposed method.

5.1.5 Evaluation Metrics

We generated a recipe for each of 16 cooking videos corresponding to seven recipes in KUSK Dataset. As we mentioned in Section 3 they are excluded from the training data. In order to investigate the effectiveness of $P(\{e\})^{-l}$, we compared the results of the models with and without it.

The evaluation metrics is BLEU ($N = 1 \sim 4$) (Papineni et al., 2002) taking the original human-written recipes as the reference. The cooking actions in the KUSK Dataset video part were performed with following these recipes. Unlike BLEU calculation in MT, we treat the entire recipe, a sequence of sentences, as the unit instead of a single sentence. This is because one can describe the same actions in various ways with different number of sentences. An example pair is "cut onions and potatoes." and "cut onions. then cut potatoes."

5.2 Results and Discussion

Since our task is quite novel and existing end-to-end video captioning methods do not obviously work because of lack of large training data, there is no direct baseline. Thus we discuss absolute BLEU scores of some settings and examples of generated sentences.

Table 2 shows the BLEU scores. The absolute BLEU values (ex. 5.50 for $N = 4$) are much higher than the results of cinema caption generation (Kaufman et al., 2016) (0.8 for $N = 4$), which is regarded as one of the state-of-the-arts of text generation for videos longer than video clips. This result is worth noting considering that cooking videos are raw recording of execution and not edited nor divided into scenes, while input of cinema caption generation is an *edited* video and scene segmentation is available. Our higher accuracy may be due to a large amount of text data in the target domain.

We then examined generated recipes and the original recipes. Figure 5 presents a recipe example actually generated by the proposed method and its original recipe used in the cooking video recording. We see that there are suitable sentences such as "挽肉を炒める。" ("Stir-fry minced meat."), "卵はほぐしておく。" ("Beat an egg.") in the result. These sentences correspond to "ミンチをいためて、色がかわったら、" ("Saute the meat mince until the color changes") and "卵をといて、" ("Beat an egg,") in the original recipe. On the other hand, the result contains some unnecessary sentences. For example, in the third line of the generation result, "包丁ですを使" ("Use the knife is."). The sentence itself is semantically correct, but is not suitable for a recipe (and grammatically wrong). This is actually the difference from the existing video clip description research.

Even if the object recognition functions perfectly, the sentence generation part has to ignore some objects focusing only on the actions to be taken. Such errors can be alleviated by considering the recipe structure such as relations of r-NEs. There are also ungrammatical sentences such as "お好みででを" ("pour over it that if if you like and serve") in the result. This sort of errors are caused by the LSTM language model. We may need a language model incorporating grammatical structures (Chelba and Jelinek, 2000).

Despite the errors mentioned above, our method solves the novel problem, procedural text generation from execution video in a certain accuracy. As it is clear from the explanation of our method, it has the correspondence between the sentence and the video frame region. Thus one can use our method for various practical multimedia applications, such as multimedia document generation from an execution video.

6 Conclusion

In this paper, we have proposed a novel task of procedural text generation from an execution video and the first attempt at solving it. Contrary to the ordinary video captioning task, it requires some kind of abstraction, that is, selecting objects to be mentioned. In addition, no existing end-to-end method is applicable due to the limited amount of video/text pairs for training. Instead, our method decomposes the problem into object recognition and sentence generation. Then we train the models for them independently with maximum available resources for each one. Finally we search for the best procedural text referring to them at once. For evaluation, we conduct recipe generation from cooking videos as an example case. The quality was as good as or better than the state-of-the-art scenario description for cinemas. Thus we can say that our method is promising to solve this novel task. We also gave some error analyses to allow further improvements in solutions of this difficult but interesting task.

Acknowledgement

In this paper, we used recipe data provided by Cookpad and the National Institute of Informatics. The work is supported by JSPS Grants-in-Aid for Scientific Research Grant Number 26280084.

References

Ciprian Chelba and Frederick Jelinek. 2000. Structured language modeling. *Computer Speech and Language* 14:283–332.

David L Chen and William B Dolan. 2011. Collecting highly parallel data for paraphrase evaluation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics, pages 190–200.

Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. 2015. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. pages 2625–2634.

Sergio Guadarrama, Niveda Krishnamoorthy, Girish Malkarnenkar, Subhashini Venugopalan, Raymond Mooney, Trevor Darrell, and Kate Saenko. 2013. Youtube2text: Recognizing and describing arbitrary activities using semantic hierarchies and zero-shot recognition. In *Proceedings of the 14th International Conference on Computer Vision*. Sydney, Australia, pages 2712–2719.

Zhao Guo, Lianli Gao, Jingkuan Song, Xing Xu, Jie Shao, and Heng Tao Shen. 2016. Attention-based lstm with semantic consistency for videos captioning. In *Proceedings of the 2016 ACM on Multimedia Conference*. ACM, pages 357–361.

Atsushi Hashimoto, Shinsuke Mori, Masaaki Iiyama, and Michihiko Minoh. 2016. Kusk object dataset: Recording access to objects in food preparation. In *Proc. of IEEE International Conference on Multimedia and Expo Workshops*. IEEE.

Atsushi Hashimoto, Sasada Tetsuro, Yoko Yamakata, Shinsuke Mori, and Michihiko Minoh. 2014. KUSK Dataset: Toward a direct understanding of recipe text and human cooking activity. In *Workshop on Smart Technology for Cooking and Eating Activities*. pages 583–588.

Justin Johnson, Andrej Karpathy, and Li Fei-Fei. 2016. Densecap: Fully convolutional localization networks for dense captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pages 4565–4574.

Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pages 3128–3137.

Dotan Kaufman, Gil Levi, Tal Hassner, and Lior Wolf. 2016. Temporal tessellation for video annotation and summarization. *arXiv preprint arXiv:1612.06950*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Natsuda Laokulrat, Sang Phan, Noriki Nishida, Raphael Shu, Yo Ehara, Naoaki Okazaki, Yusuke Miyao, and Hideki Nakayama. 2016. Generating video description using sequence-to-sequence model with temporal attention.

Guang Li, Shubo Ma, and Yahong Han. 2015. Summarization-based video caption via deep neural networks. In *Proceedings of the 23rd ACM international conference on Multimedia*. ACM, pages 1191–1194.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*. Springer, pages 740–755.

Shinsuke Mori, Hirokuni Maeta, Yoko Yamakata, and Tetsuro Sasada. 2014. Flow graph corpus from recipe texts. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation*.

Graham Neubig, Yosuke Nakata, and Shinsuke Mori. 2011. Pointwise prediction for robust, adaptable japanese morphological analysis. pages 529–533.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics, pages 311–318.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*. pages 91–99.

Anna Rohrbach, Marcus Rohrbach, Niket Tandon, and Bernt Schiele. 2015. A dataset for movie description. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Marcus Rohrbach, Wei Qiu, Ivan Titov, Stefan Thater, Manfred Pinkal, and Bernt Schiele. 2013. Translating video content to natural language descriptions. In *Proceedings of the IEEE International Conference on Computer Vision*. pages 433–440.

Tetsuro Sasada, Shinsuke Mori, Tatsuya Kawahara, and Yoko Yamakata. 2015. Named entity recognizer trainable from partially annotated data. In *Proceedings of the Eleventh International Conference Pacific Association for Computational Linguistics. 2015.*. ACM, pages 10–17.

Rakshith Shetty and Jorma Laaksonen. 2016. Frame- and segment-level features and candidate pool evaluation for video caption generation. In *Proceedings of the 2016 ACM on Multimedia Conference*. ACM, pages 1073–1076.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. pages 3104–3112.

Atousa Torabi, Christopher Pal, Hugo Larochelle, and Aaron Courville. 2015. Using descriptive video services to create a large data source for video annotation research. *arXiv preprint arXiv:1503.01070*.

Subhashini Venugopalan, Marcus Rohrbach, Jeffrey Donahue, Raymond Mooney, Trevor Darrell, and Kate Saenko. 2015. Sequence to sequence-video to text. In *Proceedings of the IEEE International Conference on Computer Vision*. pages 4534–4542.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C Courville, Ruslan Salakhutdinov, Richard S Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*. volume 14, pages 77–81.

Yezhou Yang, Ching Lik Teo, Hal Daumé III, and Yiannis Aloimonos. 2011. Corpus-guided sentence generation of natural images. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 444–454.