Calorie Estimation in a Real-World Recipe Service

Jun Harashima, Makoto Hiramatsu, Satoshi Sanjo
Cookpad Inc.
Yebisu Garden Place Tower 12F, 4-20-3 Ebisu, Shibuya-ku, Tokyo
{jun-harashima, himkt, satoshi-sanjo}@cookpad.com

Abstract

Cooking recipes play an important role in promoting a healthy lifestyle, and a vast number of user-generated recipes are currently available on the Internet. Allied to this growth in the amount of information is an increase in the number of studies on the use of such data for recipe analysis, recipe generation, and recipe search. However, there have been few attempts to estimate the number of calories per serving in a recipe. This study considers this task and introduces two challenging subtasks: ingredient normalization and serving estimation. The ingredient normalization task aims to convert the ingredients written in a recipe (e.g., 胡麻油 (仕上げ用), which says "sesame oil (for finishing)" in Japanese) into their canonical forms (e.g., ごま油, sesame oil) so that their calorific content can be looked up in an ingredient dictionary. The serving estimation task aims to convert the amount written in the recipe (e.g., N 個分, N pieces) into the number of servings (e.g., M 人分, M people), thus enabling the calories per serving to be calculated. We apply machine learning-based methods to these tasks and describe their practical deployment in Cookpad, the largest recipe service in the world. A series of experiments demonstrate that the performance of our methods is sufficient for use in real-world services.

Introduction

Developments in machine learning (ML)-related technologies continue to advance and focus on an ever-wider range of data and applications. In particular, the field of healthcare is likely to be dramatically changed by the data-processing capabilities of ML.

Cooking recipes are one such source of data, and there are a vast number of recipes currently available on the Internet. For example, Cookpad and Yummly, the largest recipe services in the world, contain more than 5.5 million and 2.0 million recipes, respectively. A considerable proportion of these recipes are homemade—many people write their own recipes and upload them to the Internet.

As the popularity of recipes has grown, so too has the number of ML-related studies that use the data included within them to perform tasks such as recipe analysis, recipe generation, and recipe search. A variety of recipe datasets are available, and related workshops such as CEA and MADiMa are held every year.

However, there has been little effort to address calorie estimation, which is one of the most important tasks in the field of healthcare. More precisely, although some studies have addressed calorie estimation from food photos, few studies have addressed this task from recipes.

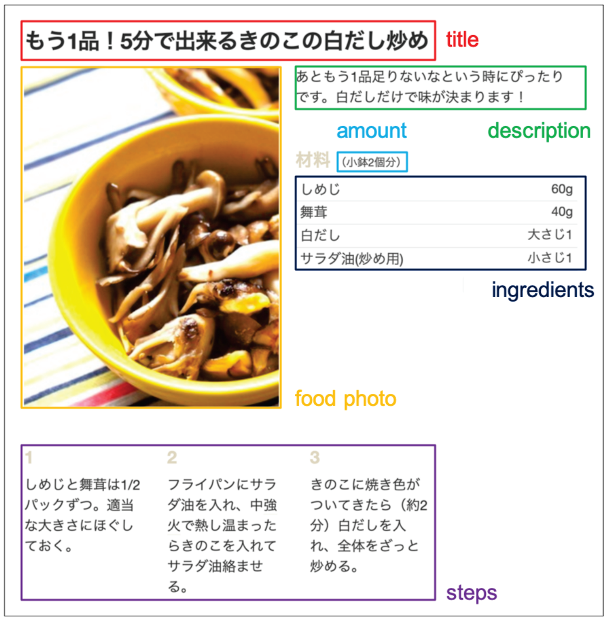
One of our challenges in this study is to address the calorie estimation task for a recipe. To do so, we must tackle two challenging subtasks. The first is ingredient normalization, in which the goal is to convert the ingredients written in a recipe into their canonical forms in an ingredient dictionary. For user-generated recipes especially, we observe a variety of issues in the ingredient lists, such as orthographic variants, abbreviations, symbols, and parentheses. We have to overcome these issues to enable the calorie contents of the ingredients to be looked up in an ingredient dictionary.

The second subtask is serving estimation, which aims to convert an amount written in a recipe into a number of servings. The amount describes how much volume the recipe produces. The amount in a user-generated recipe is often written freely. Thus, an amount may not be expressed in terms of M people but as, for example, N pieces, N slices, or N plates. These amounts need to be converted into the number of servings (i.e., for M people) so that the calories per serving can be calculated.

Another challenge in this study is to deploy ML-based methods for the above tasks in our real-world recipe service. Relatively little is known about the practical issues of introducing ML-based methods into real industrial scenarios. We consider this issue for each task, design a deployment architecture for our methods, and deploy them in our service.

In summary, the contributions of this study are as follows:

- This is a pioneering attempt at calorie estimation for online recipes that exposes the fundamental difficulties of such a task.
- The two subtasks of ingredient normalization and serving estimation are introduced and explored using ML-based methods.
- Practical issues related to deploying our methods in Cookpad, a real-world recipe service, are highlighted through a case study.



もう1品！5分で出来るきのこの白だし炒め

あともう1品足りないなという時にぴったりです。白だしだけで味が決まります！

材料 (小鉢2個分)

しめじ	60g
舞茸	40g
白だし	大さじ1
サラダ油(炒め用)	小さじ1

1 しめじと舞茸は1/2パックずつ。適当な大きさにほぐしておく。

2 フライパンにサラダ油を入れ、中強火で熱し温まったらきのこを入れてサラダ油絡ませる。

3 きのこに焼き色がついてきたら（約2分）白だしを入れ、全体をざっと炒める。

Figure 1: Example of a recipe from Cookpad.

Calorie Estimation

Task Definition

The goal of our task is to estimate the number of calories per serving in a recipe. Generally, recipes on the Internet consist of the following components: title, food photo, description, amount, ingredients, and steps. Figure 1 shows an example of a stir-fried mushroom recipe from our service. Note that we focus on Japanese recipes in this study because over 3.1 million of the 5.5 million recipes on the service are written in Japanese. We address calorie estimation in the recipes using the above components. Note that the important components of this task are the title, amount, and ingredients.

Applied Method

We estimate the calories per serving in a recipe r as follows:

$$c(r) = \frac{\sum_{i \in \boldsymbol{I}_r} c(i) \cdot q(i)/100}{s(r)}, \qquad (1)$$

where $\boldsymbol{I}_r$ represents a set of ingredients in r and i represents an ingredient in $\boldsymbol{I}_r$, $c(i)$ returns the calories per 100 grams of i, $q(i)$ returns the grams of i in r, and $s(r)$ returns the number of servings of r. In summary, $c(r)$ returns the sum of the calories in the ingredients divided by the number of servings of the recipe.

Consider the recipe in Figure 1 as an example. We first obtain $c(i)$ and $q(i)$ for each ingredient in the recipe. For example, the first ingredient in the recipe is しめじ (shimeji mushroom). For this ingredient, $c(i) = 16.2$, as obtained from an ingredient dictionary, and $q(i) = 60$. We then obtain $s(r)$ for the recipe. In this case, as the amount is 小鉢2個分 (2 small bowls), $s(r)$ can be estimated as 2. Finally, we obtain $c(r)$ using Equation (1).

Table 1: Examples of entries in an ingredient dictionary.

(a) shimeji mushroom

composition per 100 grams	kcal	16.2
	salt equivalent	0.0
	...	...
grams per unit	パック (pack)	100.0
	袋 (sack)	100.0
	...	...

(b) white dashi

composition per 100 grams	kcal	49.0
	salt equivalent	13.2
	...	...
grams per unit	大さじ (tablespoon)	18.0
	小さじ (teaspoon)	6.0
	...	...

Two Problems to Solve

Although the idea underlying our method is simple, it involves two problems that need to be solved. First, it is difficult to obtain $c(i)$. Consider again the recipe in Figure 1. We need to look up しめじ in an ingredient dictionary to obtain $c(i)$ for the ingredient. However, we cannot do this without normalizing the word. Suppose that the dictionary has an entry in Table 1 (a). This entry is for シメジ, which is an orthographic variant of しめじ. We first have to normalize しめじ to シメジ to use the information in the entry.

Note that it is not difficult to obtain $q(i)$ if we can look up the ingredients in the dictionary. Consider the third ingredient in Figure 1, which is 白だし (white dashi) with a quantity of 大さじ1 (1 tablespoon). From this quantity, we cannot determine how many grams of this ingredient are needed. However, if the entry in Table 1 (b) is available, we can determine that 18.0 grams are needed.

Second, it is difficult to obtain $s(r)$. There is no standard way of writing the amount in user-generated recipes. Thus, it may not be in terms of M people but in terms of, for example, N pieces, N slices, or N plates. For example, the amount in Figure 1 is written as 小鉢2個分 (2 small bowls). Recipes cannot be compared without knowing the number of servings (i.e., M people served), even if the sum of the calories in their ingredients are known. Thus, we have to convert the amount in each recipe into the number of servings.

Ingredient Normalization

Task Definition

As described in the previous section, the goal of this task is to normalize ingredients in a recipe to their canonical forms. Table 2 presents some examples of this task. Because people freely express the ingredients in their recipes, we have to address a variety of issues in this task.

Table 2 (a) lists several issues related to synonymous expressions. The first and second examples are related to orthographic variants and abbreviations, respectively. The third one is related to spelling mistakes. The final expression is correct, but represents a general synonymous rela-

Table 2: Examples of ingredient normalization.

(a) Examples related to synonymous expressions.

ingredient	canonical form	note
人参 (carrot)	にんじん (carrot)	orthographic variants
新たま (spring onion)	新たまねぎ (spring onion)	abbreviations
アボガド (avocado)	アボカド (avocado)	spelling mistakes
馬鈴薯 (potato)	じゃがいも (potato)	other

(b) Examples related to supplementary expressions.

ingredient	canonical form	note
♡味噌 (♡ miso)	味噌 (miso)	symbols
油 (炒め用) (oil (for frying))	油 (oil)	parentheses
お砂糖 (sugar)	砂糖 (sugar)	prefix
あればローズマリー (rosemary, if any)	ローズマリー (rosemary)	other

tionship where the latter word is commonly used. Note that, unlike examples for orthographic variants, those for the last category have different pronunciations. Whereas both 人参 and にんじん are pronounced "ninjin," 馬鈴薯 and じゃがいも are pronounced "bareisyo" and "jagaimo," respectively.

Table 2 (b) lists issues related to supplementary expressions. The first and second examples contain symbols and parentheses that should be removed in this task. The third example contains the prefix お, which adds a degree of politeness to some words and should be removed prior to looking up the words in a dictionary. We sometimes need to remove other unnecessary expressions, as in the last example.

As these phenomena can occur simultaneously, each ingredient may have a large number of expressions in a real-world recipe service. For example, しょうゆ (soy sauce) has more than 100 expressions in Cookpad, such as ♡しょう油. This makes it more difficult to normalize ingredients into their canonical forms.

Applied Method

We regard this task as a translation problem to which a neural translation model (encoder–decoder model) can be applied. An overview of our method is illustrated in Figure 2. The method takes an ingredient as input. The string is split into characters or subwords, and these are fed into the encoder individually. The decoder receives the output from the encoder and generates pieces one by one. We obtain a canonical form by concatenating the generated pieces. We do not use attention mechanisms (Bahdanau, Cho, and Bengio 2015) for this task because of a pilot study (Harashima and Yamada 2018), which revealed that the ingredients are short strings for which such mechanisms are ineffective.

An important point underlying the design of our normalization method is that we are strongly motivtated by its

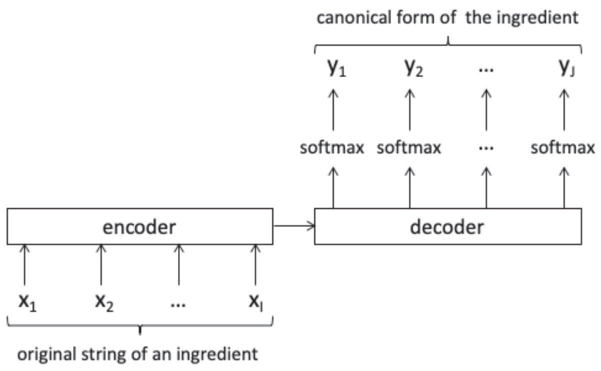


Figure 2: Neural translation for ingredient normalization.

performance when deployed. In other words, we keep the method as simple as possible. There is a vast number of mechanisms that can be added to encoder–decoder models, especially for machine translation. For ingredient normalization, we also investigated mechanisms for controlling the output of the decoder (Harashima and Yamada 2018). However, we do not use such mechanisms in our service because they make our method complex and do not contribute proportionately to its improvement.

Serving Estimation

Task Definition

The goal of serving estimation is to convert the amount stated in a recipe into the number of servings. Table 3 presents examples of this task. Similar to the ingredient list in a recipe, the amount is written freely by the recipe author, and so a variety of issues must be addressed in this task.

The first example in Table 3 is one of the easiest cases, in which the value in the amount is just the number of servings. The second example is more difficult because we need to know that 合 (go), a traditional Japanese unit, is used for two servings of rice. Note that 分 (worth) is a Japanese suffix that is attached to various units such as 皿 (plate) and 合. Because the suffix does not contain important information, we can ignore it in this task. In the third example, more human-like thinking is needed. Fourteen cookies are too much for one person, while one cookie is (generally) not enough. Three dietitians who were recruited for the annotation task considered two cookies per person to be a suitable serving. The final example needs some additional thinking: a tart is usually divided into 8, 6, or 4 parts depending on its size, and the annotators selected 8 parts for an 18-cm mold.

Note that there are several answers for some recipes. Although our annotators estimated the number of servings as eight in the last example, other people could estimate it as six, which cannot be considered incorrect. We do not pursue this issue in our service; our aim is to develop a method that estimates the number of servings in a similar manner to that of the people who use the recipes.

Table 3: Examples of serving estimation.

title	amount	serving
ココナッツオイル入りほうれん草カレー (Spinach Curry with Coconut Oil)	6皿分 (6 plates)	6
こんぶ・ひじき・ほたてごはん (Mixed Rice with Kombu · Hijiki · Scallop)	3合分 (3 go)	6
グラノーラクッキー (Granola Cookies)	14枚 (14 pieces)	7
パンプキンタルト (Pumpkin Tart)	18cmタルト型 (18-cm tart mold)	8

Applied Method

We evaluated two neural classification models for serving estimation. Figure 3 gives overviews of the models. Whereas the model in Figure 3 (a) uses either the title or the amount of a recipe, that in Figure 3 (b) uses both the recipe's title and amount. In the models, encoders take this information as their inputs and generate a vector for the text as their output. Again, each piece of text is split into characters or subwords. The output vectors are then concatenated in the multi-source model. Next, the dense layers receive the (concatenated) vector and transform it into a K-dimensional vector, where K represents the number of categories, such as one and two, which correspond with the possible serving values. The category with the highest probability is selected as the final number of servings for the recipe.

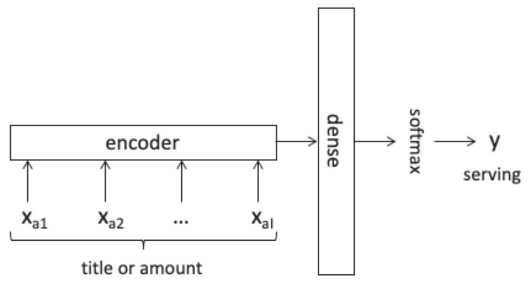
It is important to note that serving estimation should not be formulated as a regression problem, but instead treated as a classification problem. The number of servings tends to be a multiple or divisor of the value of the amount. In the second example in Table 3, the value is 3 and the number of servings is 6, which is a multiple of 3. In the third example, the value is 14 and the number of servings is 7, which is a divisor of 14. Multiples and divisors of a value are discrete; hence, classification models, which handle discrete categories, are more suitable for serving estimation than regression models, which handle continuous values.

Deployment

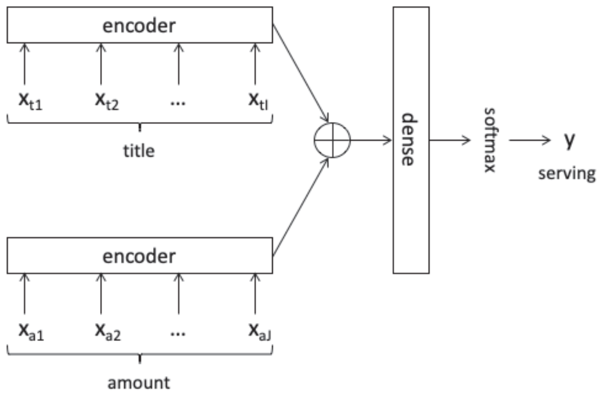
The deployment of ML-based methods is an important issue in real-world services. Figure 4 shows an overview of our deployment architecture for the methods described in the previous sections. In our case, the issue of deployment can be divided into the following three sub-issues.

The first is how to construct training sets (as well as development and test sets) for our models, as indicated by the red arrows in Figure 4. This task involves collecting examples for ingredient normalization and serving estimation, as described in Tables 2 and 3, respectively. Thus, we asked our three dietitians to manually annotate several recipes every day using a simple annotation tool. The annotation results are double-checked and then inserted into the databases.

The second sub-issue is how to construct and update our models using the training sets, as indicated by the green arrows in Figure 4. Because GPUs and deep-learning frameworks are convenient for training, we use AWS GPU instances and PyTorch, respectively. Of course, other tools, such as GitHub and Jenkins, are also used to develop the models and update them when our code or data change.



(a) Single-source model.



(b) Multi-source model.

Figure 3: Neural classification for serving estimation.

The third sub-issue is how to run the models in our service, as indicated by the blue arrows in Figure 4. Every day, we select the ingredients, especially those in newly uploaded recipes, that have not yet been normalized, use our model to normalize them, and insert the results into our normalization database. Similarly, we select recipes for which the number of servings has not yet been estimated, use our model to estimate the numbers, and insert the results into our estimation database. We use AWS CPU instances for inference because this has a relatively low computational cost. The normalization and estimation results are roughly checked by our annotators, and any errors that are located are then modified. These manually modified results are used as part of our training sets to update our models.

After overcoming the above sub-issues, we can finally estimate the calories in a recipe using Equation (1) and use this information in our service, as indicated by the yellow arrows in Figure 4. Every day, we calculate the calories of recipes for which the normalized ingredients and number of servings are newly available, and provide the results to our subscribers. Note that the results are basically accurate because the normalization and estimation results are manually checked as described above.

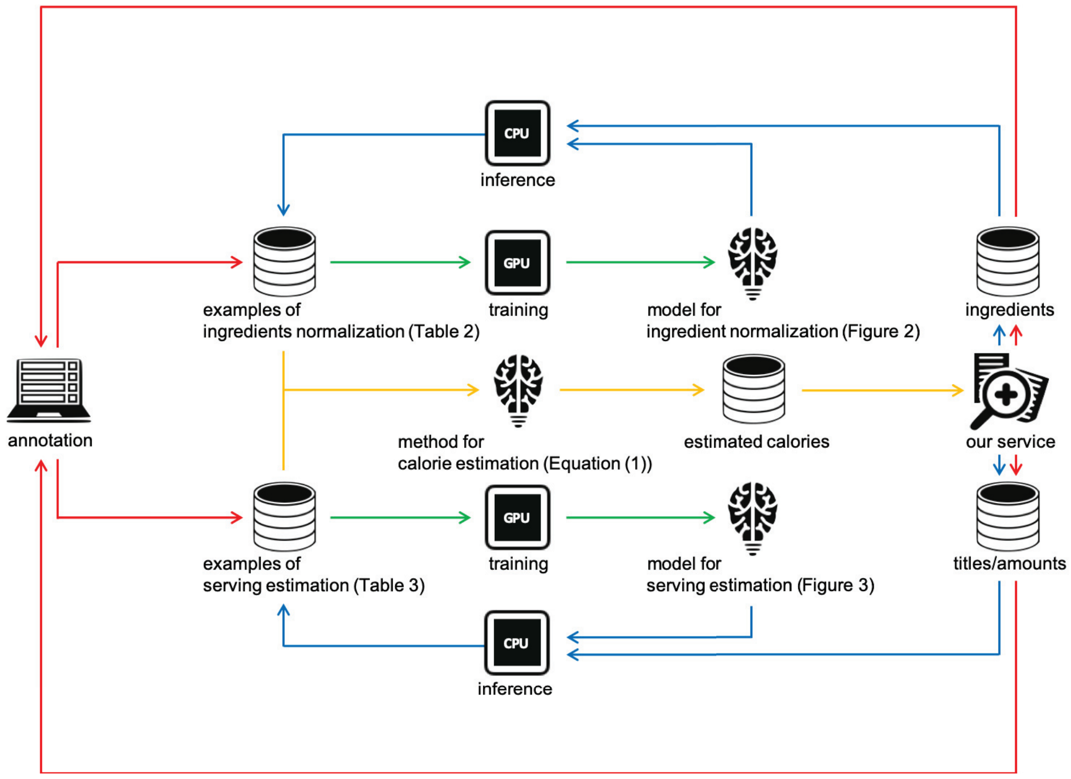

CPU
inference
examples of
ingredients normalization (Table 2)
GPU
training
model for
ingredient normalization (Figure 2)
ingredients
annotation
method for
calorie estimation (Equation (1))
estimated calories
our service
examples of
serving estimation (Table 3)
GPU
training
model for
serving estimation (Figure 3)
titles/amounts
CPU
inference

Figure 4: Overview of our deployment architecture for ingredient normalization, serving estimation, and calorie estimation.

Experiments

Ingredient Normalization

First, we evaluate our method for ingredient normalization. We collected $18,805$ distinct ingredients from our service and manually normalized them based on our internal ingredient dictionary. We then removed $2,086$ ingredients whose normalized versions were the same as their original strings and used the remaining $16,719$ ingredients in our experiment. We divided the pairs of the ingredients and their normalized versions into training, development, and test sets containing $13,375$, $1,672$, and $1,672$ pairs, respectively.

The configuration of our method was as follows. For the encoder and decoder, we used long short-term memories (LSTMs) with 2 layers, 500 units, and dropout rates of 0.2. We also used an embedding layer with 500 units before the LSTMs. These values were set using our development set. We used all characters and subwords in our training set. The character-based vocabularies for the encoder and decoder contained $1,252$ and 497 entries and the subword-based vocabularies contained $2,992$ and $1,174$ entries, respectively. The segmentation model for subwords (Kudo 2018) was trained using the 3.1 million Japanese recipes in our service.

Table 4 presents the results. The accuracy is the success rate based on the number of exact matches between the output of our methods and that of the annotators. Here, “RE” refers to a regular expression-based method, which removes symbols and parentheses from the beginning and end of each input, and “ML” refers to our ML-based method.

Table 4: Accuracy of ingredient normalization.

base	segmentation	accuracy
RE	n/a	0.20
ML	character	0.73
	subword	0.71

We can see that our ML-based method successfully normalized over 70% of the ingredients in the test set. The method solved a variety of issues, as shown in Table 5, and overcame hybrid issues related to synonymous and supplementary expressions. For example, 胡麻油 (仕上げ用) (sesame oil (for finishing)) was successfully normalized as ごま油 (sesame oil), overcoming issues related to orthographic variants and parentheses. In contrast, the RE-based method only solved issues related to supplementary expressions. In this experiment, the character-based segmentation was more effective than that based on subwords because the vocabularies for subwords were too large for the information to be efficiently embedded.

Our method made two types of normalization errors. The first was to wrongly generate nonexistent ingredients. Consider the first example in Table 6. The method wrongly normalized ニガウリ (bitter melon) to にががり, which is a nonexistent word. The second error was to wrongly generate different ingredients. Consider the second example in the table. The method normalized 薯蕷 (yam) to じゃがいも (potato). Note that these errors are manually modified in our architec-

Table 5: Successful examples of ingredient normalization.

(a) Examples related to synonymous expressions.

input	system output	note
かたくり粉 (potato starch)	片栗粉 (potato starch)	orthographic variants
牛ひき (ground beef)	牛ひき肉 (ground beef)	abbreviations
水名 (mizuna)	水菜 (mizuna)	spelling mistakes
糸うり (spaghetti squash)	そうめんかぼちゃ (spaghetti squash)	other

(b) Examples related to supplementary expressions.

input	system output	note
☆小麦粉 (⋆ flour)	小麦粉 (flour)	symbols
レーズン(お好みで) (raisin (as you like))	レーズン (raisin)	parentheses
お豆腐 (tofu)	豆腐 (tofu)	prefix
冷えたトマト (chilled tomato)	トマト (tomato)	other

Table 6: Examples of failed ingredient normalization.

input	human outpu	system output
ニガウリ (bitter melon)	ゴーヤ (bitter melon)	にががり (nonexistent word)
薯蕷 (yam)	山芋 (yam)	じゃがいも (potato)

Table 7: Agreement rates on the sub-dataset.

A	B	C	rate
✓	✓		0.64
	✓	✓	0.69
✓		✓	0.64
✓	✓	✓	0.55

Table 8: Accuracy of serving estimation.

base	source	title	amount	segmentation	accuracy
RE	single		✓	n/a	0.47
ML	single	✓		character	0.27
	single	✓		subword	0.28
	single		✓	character	0.63
	single		✓	subword	0.63
	multi	✓	✓	character	0.61
	multi	✓	✓	subword	0.62

ture, as described in the previous section.

There are two possible solutions that would overcome these errors. The first is to revise our method. In our pilot study (Harashima and Yamada 2018), for example, we introduced some mechanisms to consider the existence of the output and its similarity to the input. However, they made our method more complex. The second solution is thus more promising: expand the training set. This solution would overcome errors while preserving the simplicity of our method. We leave this task for our future work.

Serving Estimation

Next, we evaluate our method for serving estimation. In this experiment, we first constructed a sub-dataset to investigate the agreement rate among human annotations for this task. We randomly selected 160 recipes from our service and asked our three annotators to estimate the number of servings for them individually. As a result, this dataset contained 160 sets of three estimation results produced by humans.

We then constructed a main dataset to train, develop, and test our method. We collected another $5,279$ recipes from our service and asked our annotators to estimate the number of servings for them. Unlike the results in the sub-dataset, each estimate in the main dataset was produced by one of the annotators. The estimation results ranged from 1 to 20 (i.e., 20 categories). Note that the annotators used not only the title and amount from the recipes, but also other information such as the ingredient list when producing their estimations. We divided the recipes into training, development, and test sets containing $4,223$, 528, and 528 recipes, respectively.

The configuration of our method was as follows. We used LSTMs for the encoders, with 2 layers, 20 units, and dropout rates of 0.2 based on the results obtained using the development set. Similarly, each embedding layer before the LSTMs had 20 units. We used all characters and subwords in the training set. The character-based vocabularies for the title and amount encoders contained $1,323$ and 252 entries, and those based on subwords contained $2,620$ and 304 entries, respectively. The subword segmentation model was the same as that used in the ingredient normalization experiment. There were 2 dense layers, one containing 40 units and the other containing 20 units for multi-source models. For single-source models, both layers contained 20 units each.

Table 7 presents the results for the sub-dataset to investigate the annotator agreement rate. A, B, and C denote our three annotators. As shown in the table, the agreement rates between two annotators and between three annotators were $0.64-0.69$ and 0.55, respectively. There are several answers for some recipes. Sweets recipes, especially those for large amounts, are a typical example. For instance, the number of servings for a recipe titled フィナンシェ (financier) with a stated amount of 14個 (14 pieces) was estimated as 14 by two annotators and 7 by the other, both of which are potentially correct. Conversely, the annotators estimated the same number of servings for over half of the recipes, although there were many possible answers.

Table 8 presents the results for the main dataset. The RE-based method, which extracted a value from each amount, achieved an accuracy of 0.47. This suggests that approximately half of the recipes in our test set require smarter methods than this simple approach. Our ML-based method that used the amount information achieved accuracy rates greater than 0.60, and thus is better suited to this task. Although the subword-based models achieve slightly higher performance than the character-based ones, the differences

Table 9: Successful and failed examples of serving estimation.

title	amount	human	system (single)	system (multi)
シュガーパン (Sugar Bread)	2個 (2 pieces)	2	2	2
素朴なレーズンパン (Simple Raisin Bread)	1斤 (1 loaf)	8	8	8
もちもち野菜チヂミ (Chewy Korean Vegetable Pancake)	フライパン2枚分 (2 frying pans)	2	2	4
ツナポテトのミニコロッケ☆お弁当にも (Small Tsuna and Potato Croquettes ⋆ also for Lunch Box)	8個分 (8 pieces)	4	8	4
甘さ控えめのクッキー (Cookies with Less Sugar)	鉄板1枚分 (1 iron plate)	16	10	10

between them are small in this experiment.

Surprisingly, the single-source models that used only amount information outperformed the multi-source models. Essentially, the number of servings for a recipe cannot be estimated without its title information. Suppose that a recipe has 1個分 (1 piece) as an amount. If its title is マフィン (muffin), the number of servings is probably one. However, if the title is ケーキ (cake), the number is probably 8, 6, or 4. Therefore, it seems strange that the single-source models outperformed the multi-source ones.

There are two reasons for the relative success of the single-source models. One is that, in many recipes, the number of servings is often N when the amount information states, for example, N pieces, N slices, or N plates. From the results for the RE-based method, we can see that this was true for approximately half of our recipes, so it is clearly possible to estimate the number of servings for the recipes using only the amount information. The second reason is a lack of training examples. The title information was necessary for the other half of the recipes. However, the number of such examples in our training set was too small (approximately half of the $4,223$ examples) to train two encoders using the title and amount information.

Based on the above findings, we implemented the models in our service according to the following strategy. First, we introduced our single-source model with amount information because it achieved the best performance in our experiment. Then, we expanded our training set based on our deployment architecture, as indicated by the red arrows in Figure 4. Finally, we plan to replace the model with our multi-source model when the latter can outperform the former using the collected examples. We are now in the phase of data expansion, and collecting sufficient examples is an ongoing task.

Table 9 presents some examples of successes and failures of our method. Here, "human," "system (single)," and "system (multi)" refer to the output from our annotators, the single-source model, and the multi-source model, respectively. The results of both models are based on subwords. Whereas the single- and multi-source models used only the title and/or amount information in a recipe for their estimations, our annotators used all information in the recipe.

The first example in the table is an easy case for which both models successfully estimated a value of 2 for the 2個 (2 pieces) of bread recipe. Although both models succeeded in the second example, they additionally learned through our training set that 1斤 (1 loaf) of bread frequently serves eight people. The multi-source model wrongly estimated the number of servings using the title information in the third example. This is because our training set contained some examples in which one Korean pancake serves two people. In contrast, in the absence of title information, the single-source model straightforwardly estimated a value of eight in the fourth example. However, one small croquette is generally not sufficient for one person. The final example is a difficult case in which both the single- and multi-source models failed. The number of servings for this recipe depends on the size of an 鉄板 (iron plate). We might need quantity information about the ingredients, such as the amount of flour in the recipe, to output the same estimate as that of a human.

Calorie Estimation

Finally, we consider an example recipe for which the number of calories was estimated by our methods. Table 10 (a) presents the values of $c(i)$ and $q(i)$ for ingredients in an asari clam rice recipe. For each ingredient, we translated the original string in the recipe into its canonical form using our ingredient normalization method. In this example, the method successfully solved an orthographic variant issue in あさり (asari clam), prefix issues in お米 (rice), お塩 (salt), and お酒 (sake), and a hybrid issue in お醤油 (soy sauce). We then obtained the calories per 100 grams of ingredients and the grams per unit in the recipes from our ingredient dictionary.

Table 10 (b) lists a value of $s(r)$ for the recipe. Using our single-source model with amount information, the number of servings for the recipe was estimated as six. This is correct, because the quantity is 3合分 (3 go) and 1合 of rice serves two people. The model learned this knowledge from our training set and used it in the estimation.

Using Equation (1), the number of calories in the recipe was calculated at 306.6. Note that we cannot estimate the calories in recipes for which the ingredients cannot be normalized or the number of servings cannot be estimated. Nevertheless, as of 2019, we have estimated the number of calories in over $100,000$ recipes provided by our service.

Related Work

Many researchers have recently focused on recipe-related tasks such as recipe analysis (Jermsurawong and Habash

Table 10: Calorie estimation for an asari clam rice recipe.

(a) $c(i)$ and $q(i)$.

original	canonical	quantity	$c(i)$	$q(i)$
あさり (asari clam)	アサリ (asari clam)	1パック (1 pack)	12.0	200.0
お米 (rice)	米 (rice)	3合分 (3 go)	358.0	450.0
お塩 (salt)	塩 (salt)	小さじ1 (1 teaspoon)	0.0	6.0
お酒 (sake)	酒 (sake)	大さじ3 (3 tablespoons)	109.0	45.0
お醤油 (soy sauce)	しょうゆ (soy sauce)	大さじ2 (2 tablespoons)	71.0	36.0
みりん (sweet sake)	みりん (sweet sake)	大さじ3 (3 tablespoons)	241.0	54.0

(b) $s(r)$.

amount	$s(r)$
3合分 (3 go)	6

2015; Kiddon et al. 2015), recipe generation (Kiddon, Zettlemoyer, and Choi 2016; Salvador et al. 2019), and recipe search (Salvador et al. 2017; Carvalho et al. 2018). A variety of recipe datasets are now available (Harashima et al. 2016; Yagcioglu et al. 2018), and related workshops such as CEA and MADiMa are held every year.

Nevertheless, few studies have addressed calorie estimation. Some studies have attempted to estimate calorie information from an image (i.e., food photo) (Myers et al. 2015; Ege and Yanai 2017). However, there have been no studies on the estimation of calories from text (i.e., recipe). This may be because the task involves two challenging subtasks: ingredient normalization and serving estimation.

Of these two tasks, ingredient normalization has been addressed by several researchers. There are a few dictionaries (Nanba et al. 2014) that can be used to normalize ingredients. Additionally, we introduced some mechanisms into an encoder–decoder model for this task (Harashima and Yamada 2018), although they were found to be impractical.

To the best of our knowledge, there have been no studies that address serving estimation. Thus, this study is the first to focus on this important task. Our models are based on neural machine translation (NMT) techniques (Cho et al. 2014), particularly multi-source NMT (Zoph and Knight 2016), although our focus is classification rather than translation.

Conclusion

This study is the first to attempt to combine calorie estimation for online recipes and deploy ML-based methods for this task in a real-world recipe service. We described the task and its difficulties, and introduced two challenging subtasks: ingredient normalization and serving estimation. For the former, we applied a neural translation model; for the latter, neural classification models were used. The internal architecture for deploying the methods in our service was described, and experimental results were presented to demonstrate that our methods perform sufficiently well to be used in a real-world service. We also reported that they cannot be applied to certain ingredients and recipes. Therefore, future work will focus on the efficient expansion of the number of ingredients and recipes to which the methods can be applied.

References

Bahdanau, D.; Cho, K.; and Bengio, Y. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *Proceedings of ICLR 2015*, 1–15.

Carvalho, M.; Cadène, R.; Picard, D.; Soulier, L.; Thome, N.; and Cord, M. 2018. Cross-Modal Retrieval in the Cooking Context: Learning Semantic Text-Image Embeddings. In *Proceedings of SIGIR 2018*, 35–44.

Cho, K.; van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *Proceedings of EMNLP 2014*, 1724–1734.

Ege, T., and Yanai, K. 2017. Simultaneous Estimation of Food Categories and Calories with Multi-task CNN. In *Proceedings of MVA 2017*, 172–175.

Harashima, J., and Yamada, Y. 2018. Two-StepValidation in Character-based IngredientNormalization. In *Proceedings of CEA 2018*, 29–32.

Harashima, J.; Ariga, M.; Murata, K.; and Ioki, M. 2016. A Large-Scale Recipe and Meal Data Collection as Infrastructure for Food Research. In *Proceedings of LREC 2016*, 2455–2459.

Jermsurawong, J., and Habash, N. 2015. Predicting the Structure of Cooking Recipes. In *Proceedings of EMNLP 2015*, 781–786.

Kiddon, C.; Ponnuraj, G. T.; Zettlemoyer, L.; and Choi, Y. 2015. Mise en Place: Unsupervised Interpretation of Instructional Recipes. In *Proceedings of EMNLP 2015*, 982–992.

Kiddon, C.; Zettlemoyer, L.; and Choi, Y. 2016. Globally Coherent Text Generation with Neural Checklist Models. In *Proceedings of EMNLP 2016*, 329–339.

Kudo, T. 2018. Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates. In *Proceedings of ACL 2018*, 66–75.

Myers, A.; Johnston, N.; Rathod, V.; Korattikara, A.; and Gorban, A. 2015. Im2Calories: towards an automated mobile vision food diary. In *Proceedings of ICCV 2015*, 1233–1241.

Nanba, H.; Doi, Y.; Tsujita, M.; Takezawa, T.; and Sumiya, K. 2014. Construction of a Cooking Ontology from Cooking Recipes and Patents. In *Proceedings of CEA 2014*, 507–516.

Salvador, A.; Hynes, N.; Aytar, Y.; Marin, J.; Ofli, F.; Weber, I.; and Torralba, A. 2017. Learning Cross-modal Embeddings for Cooking Recipes and Food Images. In *Proceedings of CVPR 2017*.

Salvador, A.; Drozdzal, M.; i Nieto, X. G.; and Romero, A. 2019. Inverse Cooking: Recipe Generation from Food Images. In *Proceedings of CVPR 2019*.

Yagcioglu, S.; Erdem, A.; Erdem, E.; and Ikizler-Cinbis, N. 2018. RecipeQA: A Challenge Dataset for Multimodal Comprehension of Cooking Recipes. In *Proceedings of EMNLP 2018*.

Zoph, B., and Knight, K. 2016. Multi-Source Neural Translation. In *Proceedings of NAACL 2016*, 30–34.