Routing Enforced Generative Model for Recipe Generation

Zhiwei Yu[*,1,2], **Hongyu Zang**[*,1] **and Xiaojun Wan**[1]
[1]Wangxuan Institute of Computer Technology, Peking University
[1]The MOE Key Laboratory of Computational Linguistics, Peking University
[2]Microsoft Research Asia
yuzw, zanghy, wanxiaojun@pku.edu.cn

Abstract

One of the most challenging part of recipe generation is to deal with the complex restrictions among the input ingredients. Previous researches simplify the problem by treating the inputs independently and generating recipes containing as much information as possible. In this work, we propose a routing method to dive into the content selection under the internal restrictions. The routing enforced generative model (RGM) can generate appropriate recipes according to the given ingredients and user preferences. Our model yields new state-of-the-art results on the recipe generation task with significant improvements on BLEU, F1 and human evaluation.

1 Introduction

Food is a critical contributor to physical well being, a major source of pleasure, worry and stress, a major occupant of waking time, and across the world, the single greatest category of expenditures for human beings (Rozin et al., 1999). Recipes are a specific genre of instructional language to teach people how to prepare delicious food. They have been gaining interests in recent researches as recipes contain immensely rich information about the real world (Yagcioglu et al., 2018).

Among previous efforts towards computational recipe studies, there are two lines in obtaining cooking recipes for users: recipe retrieval and recipe generation. Recipe retrieval (Chen and Ngo, 2016; Min et al., 2017) matches the entities from the given dish pictures or text inputs to find the corresponding recipes. Provided with ingredients (Yang et al., 2017), recipe titles (Kiddon et al., 2016), or dish photos (Salvador et al., 2019), recipe generation models introduce additional mechanisms to assure the generated recipes containing as much given ingredients as possible. In previous studies, the target recipes are exactly composed of the given ingredients. However, in practice, people usually have a number of ingredients at hand and do not know what to cook. They have difficulty in choosing appropriate set of ingredients. And it can be hard for them to input an accurate recipe title for the models (Kiddon et al., 2016; Majumder et al., 2019). What's more, users may have preferences on some ingredients (e.g. "olive oil") or categories (e.g. "Low Sugar"). As Figure 1 shows, given the same ingredient list, there can be different sets of ingredients contributing to recipes with different user demands. Previous researches have not discussed this common scene of life. There is a clear need in finding suitable cooking recipes that match user demands. As increasing the variety and creativity of daily dishes can promote our happiness, in this work, we manage to obtain the desired recipes in a generation manner.

Ingredients	Low Sugar	Low Sugar + olive oil
beef, cauliflower, celery salt, celery, egg, onions, olive oil, pepper, salmon, sauce, ...	cook onion, celery and pepper until softened. stir in ketchup. combine one half of the sauce with ground beef, breadcrumbs, egg, celery salt. mix gently. bake for 1 hour. slice meat loaf and serve with sauce.	heat the olive oil in a medium saute pan. add beef and cook for a few minutes. add the cauliflower and cook over medium heat for about 8 minutes. add the onions and cook for another 5 minutes. season with celery salt.

Figure 1: Recipe Examples

Our task is generally defined as follows:

> *Given the input of objective background information I and subjective semantic constraints C, our model is to help the machine automatically grounding the inputs to the text output $\boldsymbol{Y}$.*

Such application contexts is common for many specific tasks. For recipe generation, we define I as

[*] The two authors contributed equally to this paper. Contribution was done at Peking University.

Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, pages 3797–3806,
November 16–20, 2020.

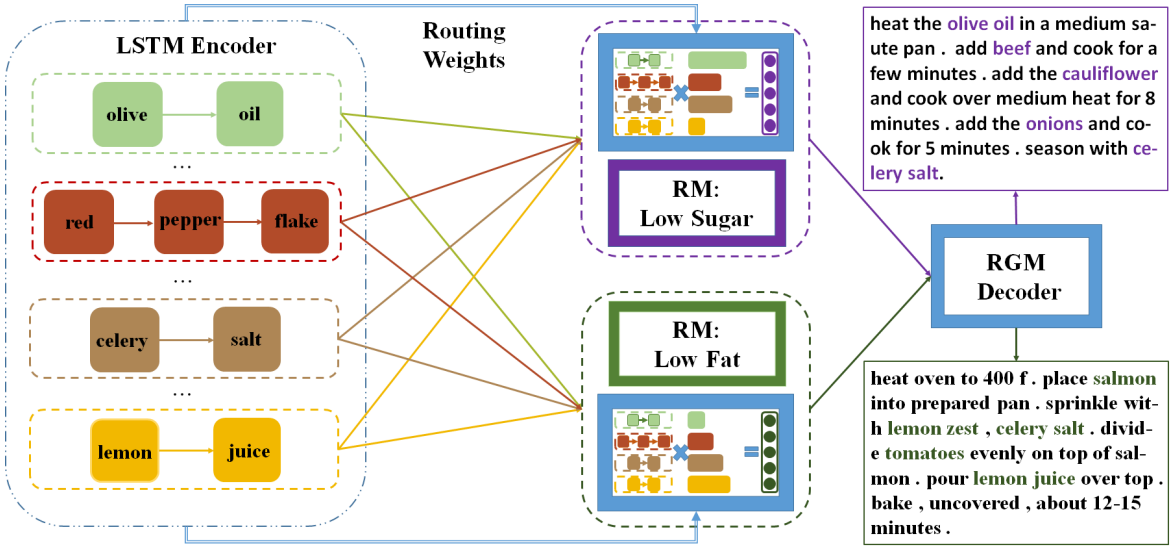

Figure 2: Routing Enforced Generative Model (RGM). LSTM encoder calculates representations of the input ingredients. Routing module computes the routing weights. Attention decoder generates the recipes allowing for routing weights and user demands.

a set of ingredients (e.g. "beef", "olive oil") the user has. And $C = \{CI, CD\}$ denotes user preferences on some ingredients ($CI \subseteq I$) and category demands ($CD \subseteq D$, D represents the set of dish categories. e.g. "Low Fat", "Low Sugar"). The generated $\boldsymbol{Y}$ is the text of the recipe which satisfies the user preferences with the given ingredients. There are two significant challenges: how to select appropriate set of ingredients to satisfy user demands; and how to generate recipes accordingly. We propose a novel approach to solve these problems.

Inspired by (Sabour et al., 2017), we propose a selective routing algorithm to cluster the given ingredients into five categories (*Low Sugar*, *High Fiber*, *Low Fat*, *Grilling* and *Frying*) and get the category vectors. Length of the category vector represents the probability of generated $\boldsymbol{Y}$ belonging to a specific category. We augment attention mechanism to capture ingredient information according to the routing weights between ingredients and categories. Then decoder generates words in sequence. We introduce both manual ways and automatic metrics to evaluate the generated recipes. Experimental results demonstrate the efficacy of our approach. To summarize, the contributions of our work are as follows[1]:

- To our knowledge, our work is the first endeavor to take ingredient selection into consideration in recipe generation process. We propose a novel algorithm to calculate the ingredient collocation weights to enforce recipe generation model.
- Given ingredients with noises, our model can satisfy personalized user demands by taking ingredients and category constraints into consideration.
- Our approach yields significant improvements on both automatic and human evaluation.

2 Related Work

Recipes have been gaining interest in recent researches, including recipe processing (Mori et al., 2012, 2014; Bosselut et al., 2017), recipe parsing (Malmaud et al., 2014; Jermsurawong and Habash, 2015), recipe retrieval (Chen and Ngo, 2016; Min et al., 2017), regional cuisine style transformation (Kazama et al., 2018), recipe QA (Yagcioglu et al., 2018) and recipe generation (Kiddon et al., 2016; Yang et al., 2017). Among previous efforts towards recipes researches, our study is closer to recipe generation. Kiddon et al. (2016) define the task as given a goal (recipe title) and an agenda (ingredient list) to generate a complete recipe. They present the neural checklist model to improve the semantic coverage of the agenda in the generated texts. Yang et al. (2017) develop a language model that treats reference as an explicit stochastic latent variable and create mentions of entities together with their

[1]https://github.com/ArleneYuZhiwei/RGM-for-Recipe-Generation

attributes by accessing external databases. Majumder et al. (2019) take the historical user preference records into consideration. They generate personalized recipes from incomplete input specifications (name and incomplete ingredient details).

Different from the existing methods on recipe generation which focus on covering all of the given ingredients, we extend the previous generation task with a selection of given items. Selective generation is a task to produce the natural language description for a salient subset of a rich records (Mei et al., 2016). A lot of attention has been paid to individual content selection and selective realization sub-problems (Barzilay and Lee, 2004; Barzilay and Lapata, 2005; Liang et al., 2009). Recent works (Chen and Mooney, 2008; Chen et al., 2010; Mei et al., 2016) explore full selective generation and learn alignments between generated texts and input data using a translation model.

We find appropriate set of ingredients by selective routing algorithm. And our model can generate texts according to the user constraints on both ingredients and categories. The core inspiration for our routing module comes from following works. Hinton et al. (2011) propose transformation matrices that learn to encode the intrinsic spatial relationship between a part and a whole constitute viewpoint. Later, Sabour et al. (2017) propose an iterative routing-by-agreement mechanism to learn the intrinsic relationship between two layers. Hinton et al. (2018) propose a new iterative routing procedure based on the EM algorithm. Inspired by the previous work, Yang et al. (2018) firstly investigate the performance of dynamic routing on text classification. They propose three strategies (orphan category, leaky-softmax, coefficient amendment) to stabilize the dynamic routing process and alleviate the disturbance of some noises.

3 Our Approach

The basic structure of our model is shown in Figure 2. Generally speaking, we take two steps to achieve the recipe generation:

Select with Routing: In this part, our task is to select an appropriate set (soft selection as weight distribution) of ingredients to support a dish for each category with the given constraints. The procedure can be defined as

$$\boldsymbol{O} = f_S(I, D, C), \tag{1}$$

where ingredients I and constraints $C = \{CI \subseteq I, CD \subseteq D\}$ are inputs. We propose a selective routing algorithm to cluster the ingredients I into different dish categories D. Assuming that I and D contains n and m items independently, the output $\boldsymbol{O} \in \mathbb{R}^{n \times m}$. We define $\boldsymbol{O}$ as the routing weights, where $o_{i,j}$ stands for the importance of the ingredient i in the category j.

Generate with Attention: After the content selection, we choose a proper category d and use the routing weights $\boldsymbol{o_{*,d}}$ [2] to help with the attention-based generation process. And we get the recipe Y by

$$Y = f_G(I, \boldsymbol{o_{*,d}}), \tag{2}$$

The generative module is an improved encoder-decoder framework with hierarchical attention mechanism.

In the following sections, we will give more details about the model design and training objective.

3.1 Routing Module (RM)

The routing module is designed to find routing weights that contribute to the generation process. The process is shown in Algorithm 1. We first apply an LSTM network as the encoder to obtain the representation of the i-th ingredient $\boldsymbol{h^{(i)}} \in \mathbb{R}^z$ (z is the size of hidden vectors). $\boldsymbol{h^{(i)}}$ is the average of the encoder hidden states $\boldsymbol{H^{(i)}} \in \mathbb{R}^{n_i \times z}$ (n_i is the number of words in the i-th ingredient). We then obtain the corresponding routing vectors $\boldsymbol{U} \in \mathbb{R}^{n \times z}$ from the ingredient representation, where $\boldsymbol{u_{i,*}} = \boldsymbol{h^{(i)}}\boldsymbol{M}$ and $\boldsymbol{M} \in \mathbb{R}^{z \times z}$ is a mapping matrix to map the ingredient semantic information to the routing space. Given routing vectors $\boldsymbol{U} = \{\boldsymbol{u_{1,*}}, \boldsymbol{u_{2,*}}, ...\boldsymbol{u_{n,*}}\}$ and the routing iteration number r, we use selective routing algorithm f_R to obtain the routing weights $\boldsymbol{O} \in \mathbb{R}^{n \times m}$ and category vectors $\boldsymbol{V} \in \mathbb{R}^{m \times z}$.

We define coupling coefficients as $\boldsymbol{B} \in \mathbb{R}^{n \times m}$. The initial values $b_{i,j}$ are the log prior probabilities that ingredient i should be coupled to category j. And then we calculate the routing weights $o_{i,j}$ by Eq 3. Inspired by (Sabour et al., 2017), we use the length of category vector $\boldsymbol{v_{j,*}}$ to represent the probability that an appropriate recipe with a specific category j exists. To get the category vector, we apply a non-linear squash function on the weighted sum $\boldsymbol{s_{j,*}}$ by Eq 4. By this means, short vectors get shrunk to almost zero length and long vectors get

[2] For a matrix $\boldsymbol{A}$, $a_{i,j}$ denotes the item at i-th row, j-th column , $\boldsymbol{a_{i,*}}$, $\boldsymbol{a_{*,j}}$ denotes the i-th row vector and j-th column vector respectively.

Algorithm 1 Selective Routing Algorithm

procedure $f_S(I, D, C)$
  Get the representation of the i-th ingredient by LSTM encoder: $\boldsymbol{h^{(i)}} \leftarrow I_i$
  Map $\boldsymbol{h^{(i)}}$ to the routing space: $\boldsymbol{U} = \{\boldsymbol{u_{1,*}}, \boldsymbol{u_{2,*}}, ...\boldsymbol{u_{n,*}}\}, \boldsymbol{u_{i,*}} \leftarrow \boldsymbol{h^{(i)}M}$
  Initialize all the coupling coefficients of ingredient i and category j: $b_{i,j} \leftarrow -\infty$
  for $di \in CI, j \in D$: $b_{di,j} \leftarrow \alpha$ ▷set ingredients demands
  Get selections by r-iteration routing algorithm: $\boldsymbol{B,O,V} \leftarrow f_R^r(\boldsymbol{U}, I, D, C, \boldsymbol{B})$
  return $\boldsymbol{O}$
procedure f_R $(\boldsymbol{U}, I, D, C, \boldsymbol{B})$
  for all category $j \in D$: $\boldsymbol{o_{*,j}} \leftarrow$softmax$(\boldsymbol{b_{*,j}})$ ▷softmax computes Eq.3
  for all category $j \in D$: $\boldsymbol{s_{j,*}} \leftarrow \sum_i o_{i,j}\boldsymbol{u_{i,*}}$
  for all category $j \in D$: $\boldsymbol{v_{j,*}} \leftarrow$squash$(\boldsymbol{s_{j,*}})$ ▷squash computes Eq.4
  for all ingredient $i \in I$ and category $j \in D$: $b_{i,j} \leftarrow b_{i,j} + \boldsymbol{u_{i,*}} \cdot \boldsymbol{v_{j,*}}$
  for $j \in D \setminus CD$: $\boldsymbol{v_{j,*}} \leftarrow \boldsymbol{0}$ ▷set categories demands
  return $\boldsymbol{B,O,V}$

shrunk to a length slightly below 1 (Sabour et al., 2017). In the training phase (detailed in Section 4.3), the category vectors and routing weights are used to calculate the loss for routing module. When making predictions with user constraints C, we set the prior probabilities of desired ingredients CI to α to emphasize the ingredient preferences. The presetting will lead the routing to converge to a desired category. And we also mask the vectors of undesired category $(D \setminus CD)$ in each iteration. If the preferred category is not specified in CD, we extract the routing weights $\boldsymbol{o_{*,\hat{j}}}$ as the routing weights for generation model, where $\hat{j} = argmax_j||\boldsymbol{v_{j,*}}||$ denotes the most possible category of the target recipe.

$$o_{i,j} = \frac{exp(b_{i,j})}{\Sigma_{k=1}^{m} exp(b_{i,k})}. \quad (3)$$

$$\boldsymbol{v_{j,*}} = \frac{||\boldsymbol{s_{j,*}}||^2}{1 + ||\boldsymbol{s_{j,*}}||^2} \frac{\boldsymbol{s_{j,*}}}{||\boldsymbol{s_{j,*}}||}. \quad (4)$$

3.2 Routing Enforced Generative Model (RGM)

The generation module shares the same LSTM encoder with RM. We augment the attention mechanism (Luong et al., 2015) to capture relevant ingredient information to help with predicting the current target word. The alignments $\boldsymbol{a_i} \in \mathbb{R}^{n_i}$ between the last target hidden state $\boldsymbol{h_{t-1}} \in \mathbb{R}^z$ and hidden states $\boldsymbol{H^{(i)}} \in \mathbb{R}^{n_i \times z}$ of the ingredient i is calculated as Eq 5, where $\boldsymbol{W}_T \in \mathbb{R}^{z \times z}$ is a linear transformation matrix for the ingredient representations. Different from the previous attention mechanism, we obtain the context vector $\boldsymbol{c_t} \in \mathbb{R}^z$ taking both routing weights and alignment vector into consideration as Eq 6 shows. In this way, the undesired ingredients (with lower routing weights) get lower attentions. We produce an attention hidden state by concatenating the context vector and the hidden state. And then we use an LSTM decoder to get the word distribution $\boldsymbol{p_t}$ formulated as Eq 7, where $\hat{\boldsymbol{Y}} = \{\hat{Y}_1, ..., \hat{Y}_{|\hat{\boldsymbol{Y}}|}\}$ denotes the word sequences of the target text. The word with highest probability in the distribution is selected as the generated word.

$$\boldsymbol{a_i} = softmax(\boldsymbol{h_{t-1}}\boldsymbol{W}_T(\boldsymbol{H^{(i)}})^\top). \quad (5)$$

$$\boldsymbol{c}_t = \Sigma_{i=1}^{n} o_{i,\hat{j}} \boldsymbol{a_i} \boldsymbol{H^{(i)}}. \quad (6)$$

$$\boldsymbol{p_t} = softmax(LSTM(\hat{Y}_{t-1}, [\boldsymbol{c_t}; \boldsymbol{h_{t-1}}])). \quad (7)$$

3.3 Model Training

We pre-train the routing module. We mix the input ingredients and target categories of n_r recipes as one datum to build a mixed training set $\boldsymbol{MT}$. The loss function of RM consists of two parts: classification loss and routing loss. For multiple classification, we use classification loss $\hat{L}_j$ for each category j as Eq 8, where $e_j = 1$ iff category j exists in the mixed target categories, otherwise $e_j = 0$. As for routing loss, we define the gold routing weights as $\boldsymbol{G} \in \mathbb{R}^{n \times m \times n_r}$. $g_{i,j,k} = 1$ iff the i-th ingredient in the inputs is used for the j-th category in the k-th recipe, otherwise $g_{i,j,k} = 0$. As there may be multiple input combinations for one target category in one mixed datum, we hope our predicted routing weights of the category have a good consistency with one gold combination. Therefore, we

maximize the max sum of weights along the gold combinations as Eq 9. The loss function for RM is calculated on the mixed training set $\boldsymbol{MT}$ as Eq 10.

$$\hat{L}_j = e_j(1 - ||\boldsymbol{v_{j,*}}||)^2 + (1 - e_j)(||\boldsymbol{v_{j,*}}||)^2. \quad (8)$$

$$\hat{L_c} = 1 - \max_{j=1}^{m} \max_{k=1}^{n_r} \left\{ \sum_{i=1}^{n} g_{i,j,k}\, o_{i,j} \right\} \quad (9)$$

$$L_{RM} = \mathbb{E}_{\boldsymbol{MT}} \left(\sum_{j=1}^{m} \hat{L}_j + \hat{L_c} \right). \quad (10)$$

For training RGM, we use the expectation of negative log likelihood loss over the generation training set $\boldsymbol{GT}$ as Eq 11. The probability is modeled by the encoder and decoder of RGM with parameters $\boldsymbol{\theta}$. As RM and generative module (GM) share the encoders, the whole model is jointly trained by $L_{RGM} = L_{RM} + L_{GM}$.

$$L_{GM} = \mathbb{E}_{\boldsymbol{GT}} - logp(\hat{\boldsymbol{Y}}|I, D; \boldsymbol{\theta}). \quad (11)$$

4 Experiments

4.1 Data sets

To keep in line with the previous work on recipe generation (Yang et al., 2017), we use the recipe data from Allrecipe[3] to train the generative model. We exclude the recipes that contain less than 10 tokens or more than 500 tokens. As the vocabulary is limited in recipes, we keep all the words appearing in the training set. The vocabulary sizes of ingredients and recipes are 6,121 and 19,168 respectively. The training set $\boldsymbol{GT}$ contains 73,088 recipe data, while valid set and test set (Standard Inputs) each contains 8,000 recipe data. To explore the generality of our model, we build another test data set (Mixed Inputs). It mixes the ingredients of two recipes for each test case to provide redundant ingredients.

To pre-train the routing module, we use 312,707 ingredients with their corresponding recipe categories from Yummly[4]. We mix the input ingredients and target categories of nr = 2 recipes to build the mixed training set $\boldsymbol{MT}$.

4.2 Baseline Models

In this work, we investigate how to improve recipe generation over strong baselines in both our setting (Mixed Inputs) and common setting (Standard Inputs). We compare our routing enforced generative model against the baseline models below.

Attseq: As the bidirectional LSTM encoder has proved strong representation capability (Devlin et al., 2019). We use the model with bidirectional encoder and Luong attention decoder (Luong et al., 2015) as our baseline model.

Pointer: As the vocabulary used in instructional language is limited and there is a strong relationship between given ingredients and recipes, seq2seq model with the pointer network performs particularly well in previous recipe generation work (Yang et al., 2017). We use the pointer network (Vinyals et al., 2015) with ingredient attention, which provides comparable performance to reference-aware language model (Yang et al., 2017) and higher *BLEU*[5].

Retrieve: The model retrieves recipes from the training set according to the overlap of the input ingredients.

To explore the importance of routing algorithm, we report the performance of our model without routing module (**w/o RM**).

4.3 Training Details

Attseq, **Pointer** and our model **RGM** all use 2-layer LSTM encoders and decoders. All the hidden sizes in three generation models are 512. To avoid over-fitting, we set the dropout rates to 0.3 in these models. We use Adam (Kingma and Ba, 2014) as the optimizer and the learning rate is 0.0001. As for hyper parameters, we set the routing iteration $r = 3$,[6] weight increase factor $\alpha = 100$.

4.4 Automatic Evaluation

Metrics In order to evaluate the effectiveness of our methods, we introduce the automatic metrics as follows: *BLEU-4* (Papineni et al., 2002) is a commonly used metric to measure the quality of machine generated texts. *Dis2* (Li et al., 2016) is the ratio of distinct bi-grams in the generated recipes, which depicts the diversity[7]. We define the set of used ingredients in the model outputs and gold reference are $\boldsymbol{SO}$ and $\boldsymbol{SG}$ respectively. *Prec.* denotes the ratio of $\boldsymbol{SO} \cap \boldsymbol{SG}$ in $\boldsymbol{SO}$. *Rec.* denotes

[3] www.allrecipes.com
[4] www.yummly.com
[5] The performances of two models are also comparable in original settings (Yang et al., 2017)
[6] We discuss the performance of model with different iterations in appendix
[7] *Dis1* of all the models are quite small with negligible differences. So we only discuss *Dis2* in the paper.

	Mixed Inputs					Standard Inputs				
Models	*BLEU-4*	*Dis2*	*Prec.*	*Rec.*	*F1*	*BLEU-4*	*Dis2*	*Prec.*	*Rec.*	*F1*
Attseq	13.67	0.10	40.24	36.28	38.16	8.54	0.08	45.60	37.13	40.93
Pointer	14.73	0.23	**50.05**	40.55	44.80	7.50	0.18	51.60	40.69	45.50
Retrieve	11.45	**0.61**	30.10	**63.02**	40.74	10.54	**0.50**	32.38	**65.22**	43.27
RGM	**20.16**	0.35	46.11	55.90	**50.54**	**16.02**	0.35	**58.11**	62.26	**60.11**
w/o RM	18.77	0.24	45.80	43.61	44.68	13.28	0.18	54.41	51.36	52.84

Table 1: Results of Automatic Evaluation (%).

the ratio of $\boldsymbol{SO} \cap \boldsymbol{SG}$ in $\boldsymbol{SG}$. *F1* is the harmonic average of the *Prec.* and *Rec.*, which is an overall measurement. For all the metrics, a higher value means better.

Analysis Results of generated texts from all the models when given Mixed Inputs or Standard Inputs are shown in Table 1. As we mixed ingredients of two recipes to create Mixed Inputs, we take each recipe as the ground truth respectively and report the max score over the two ground truths via automatic evaluation metrics. The general trends are consistent in both Standard Inputs and Mixed Inputs, so we discuss them together.

As the vocabulary used in instructional language is limited and there is a strong relationship between given ingredients and recipes, **Pointer** performs better than **Attseq** in almost all the evaluation metrics. Sometimes **Pointer** may directly copy ingredients from the given inputs in the generation process, and it achieves a rather high *Prec.* With overmuch attention on the limited ingredients, **Pointer** generates recipes of low *Rec.* On the contrary, **Retrieve** finds the recipes that contain most ingredients in the training set as outputs, which results in the highest *Rec.* But gaps exist in training set and test set, and the gold recipes from two sets might be different even if the inputs are the same, letting alone the retrieved outputs are always corresponding to the super sets of the given inputs in the test set. And thus **Retrieve** obtains the lowest *Prec.* In all collected data sets, the ingredient names in the ingredient lists may disagree with corresponding expressions in the recipes. We use ingredient mapping rather than word mapping to calculate the automatic evaluation metrics, because "green onion" is not the same ingredient as "onion". However, "basil leaf" in the ingredient list is used as "basil" in the recipe and both expressions represent the same ingredient. Due to this inconformity, *Rec.* of **Retrieve** is not 1. Besides, the outputs are all handcrafted recipes with a higher diversity compared to the generative models.

Models	*Read.*	*Acc.*	*Crea.*	*Feas.*	*Overall*
Attseq	3.07*	2.70*	2.82*	2.76*	2.73*
Pointer	2.83*	2.57*	2.67*	2.75*	2.57*
Retrieve	3.87*	3.89*	3.69	3.63*	3.64*
RGM	**4.13**	**4.23**	**3.77**	**4.12**	**3.98**
w/o RM	4.01	3.61*	3.21*	3.84	3.52*
Gold	4.24	4.37	3.82	4.31	4.13

Table 2: Results of Human Evaluation. * denotes that our model outperforms the baseline model significantly in this aspect, based on two-tailed paired t-test with p < 0.01.

To explore the efficacy of selective routing, we remove the routing module in the test phase. Due to the strong representation capability of shared encoder, *BLUE-4* , *Prec.* and *F1* of **w/o RM** achieve evident promotion compared to **Attseq**. Combined with **RM**, our **RGM** achieves the best performance on *BLUE-4* and *F1* in both cases. And the recipes generated by **RGM** use words with the highest diversity among all the automatically generated recipes. This ablation study demonstrates the empirical contribution of routing algorithm. Automatic evaluation results demonstrate that our model learns the internal relationships of the ingredients well and can generate recipes with high quality.

4.5 Human Evaluation

Settings Because recipes are a kind of instructional language to teach people how to cook, performing well on the automatic metrics is not enough. For a more comprehensive evaluation, we sample 50 inputs and obtain 50 recipes generated by each model above and then get 50 groups of recipes (25 for each case) to do the human evaluation. We ask 9 judges[8] on Amazon Mechanical Turk to rate the recipes in a Likert scale ($\in [1, 5]$). Three native English speakers are asked to give a score on each recipe with the following descrip-

[8]For quality assurance, we choose the judges whose HIT approval rate is greater than 90.

Mixed Inputs			
agave nectar , red wine vinegar , ice , strawberry , mayonnaise , carrot , blue cheese , celery salt , banana , broccoli floret , raisin , white sugar , mustard , cabbage , onion powder , tofu , red pepper flake , olive oil			
Gold	**Attseq**	**Retrieve**	**RGM**
mix **mayonnaise** , **blue cheese** , **sugar** , **olive oil** , **mustard** , **vinegar** , **red pepper flakes** , **onion powder** , and **celery salt** together in a large bowl . add **cabbage** , **carrots** , **broccoli** , and **raisins** ; stir to coat . chill in refrigerator at least 2 hours before serving .	in a large bowl , combine the first , **lime juice** , **sugar** , and **salt** . add the **dressing** , and cook until smooth , about 1 minute . add the **lime juice** , and toss to coat . pour into the prepared pan . bake in the preheated oven until the fish is no longer pink in the center of the center should read .	in a large bowl , whisk together the **sugar** , **salt** , **pepper** , **onion powder** , **mustard** , **garlic** , **basil** , **oregano** , **cilantro** , **hot pepper sauce** , **red wine vinegar** , **olive oil** , **lemon juice** , and **parmesan cheese** . add the **tomatoes** , **green onions** ...**pine nuts** . refrigerate for at least 1 hour before serving .	place the **tofu** , **broccoli** , and **carrots** into a large bowl . pour the **olive oil** , **vinegar** , and **sugar** over the mixture ; stir to coat . cover and refrigerate for at least 1 hour . stir the **mayonnaise** , **red wine vinegar** , and **red pepper flakes** into the mixture . stir in the **blue cheese** . cover and refrigerate for at least 1 hour before serving .
Standard Inputs			
lime , coffee , cocoa powder , water , white sugar , vanilla ice cream			
Gold	**Attseq**	**Retrieve**	**RGM**
cut **limes** in half and squeeze lime juice into a 2 quart pitcher . add **sugar** and stir until dissolved . fill pitcher with cold **water** , stir and chill in refrigerator . in a blender , combine **ice cream** , **coffee** and **cocoa** . blend until smooth . pour into glasses and serve .	in a large bowl , combine the cream , **sugar** , and *sugar* . mix well . pour into a bowl . pour in the sugar mixture . pour over the **ice cream** . refrigerate until chilled .	place **water** , **cocoa** , **sugar** and instant **coffee** in blender container . cover ; blend briefly on low speed . add **milk** ... **ice cream** . cover ; blend until smooth . serve immediately .	place the **ice cream** , **cocoa powder** , **sugar** , **water** , and **coffee** into a blender . cover , and puree until smooth . pour into glasses and serve .

Figure 3: Examples of Generated Recipes.

tions: *Readability* denotes whether the recipe is fluent and easy to understand. *Accuracy* denotes whether the given ingredients are correctly used in the recipe. *Feasibility* denotes whether the recipe is feasible. *Creativity* denotes whether the recipe is innovative. *Overall* denotes the overall quality of the recipe.

Analysis The results in Table 2 show that there is a large gap between automatic evaluation and human evaluation on **Pointer**. It outperforms **Attseq** on automatic evaluation but gets the worst rating scores on human evaluation. It may be due to that **Pointer** sometimes reuses the same ingredients in a recipe for several times or even repeats phrases, which does not affect the automatic metrics much but discourages people from reading. This inconsistency suggests the deficiencies of the automatic evaluation. Another interesting discovery is that **Retrieve** achieves lower scores on all the aspects compared to our model. As **Retrieve** returns handcrafted recipes as outputs, scores of its *Readability* and *Feasibility* should have been higher than our model's. We compare the outputs of both models and find **Retrieve** outputs rather long recipes. On the test sets, the average word numbers of the outputs are 173.63 and 96.95 for **Retrieve** and **RGM** respectively, while gold recipes contain 104.41 words averagely. Longer texts mean that there are more operation steps or more ingredients used, which makes the recipe difficult to understand and follow. What's more, **Retrieve** outputs common recipes in the data set, while **RGM** generates recipes with novelty. Therefore our model also beats **Retrieve** on *Creativity*. Human evaluation demonstrates that effectively calculating the routing weights of ingredients is informative for recipe generation.

4.6 Case Study

For an intuitive comparison, we show some examples in Figure 3. As **Pointer** achieves rather low scores in human evaluation, we only show the recipes generated by two stronger baseline models here. We apply some ellipsis because **Retrieve** always outputs long texts[9]. The noise ingredients (given in the Mixed Inputs but not expected to use) are in purple and extraneous ingredients (not in the given inputs) are in red. The ingredients which are correctly used in the recipes are in bold black. Italic words denote that the ingredients are supposed to appear in the recipes and have been used before. In both cases, **Attseq** generates recipes containing few expected ingredients. Particularly, it introduces extraneous ingredients when given Mixed Inputs and incorrectly reuses "sugar" given Standard Inputs. As for **Retrieve**, it introduces quite a few extraneous ingredients in both cases. **Retrieve** outputs complicated recipes with overmuch ingredients and lengthy operation steps, which are not practical for most people. As a contrast, **RGM** uses most of the expected ingredients and only one noise ingredient,

[9]Please refer to appendix for details

Ingredients	celery seed , mustard , mayonnaise , horseradish , sour cream , green zucchini , dill weed , apple cider vinegar , cayenne pepper , yellow bell pepper , dark brown sugar , salt , cabbage	
Constraints	None	mustard
Recipes	in a large bowl , mix together the **cabbage** , **green onions** , and green **bell pepper** . in a separate bowl , whisk together the **mayonnaise** , **sour cream** , **salt** , **celery seed** , and **brown sugar** . pour over the cabbage mixture , and toss to coat . cover , and refrigerate at least 1 hour before serving .	in a medium bowl , mix **mayonnaise** , **sour cream** , **horseradish** , **brown sugar** , **mustard** , **celery seed** , **salt** , and **cayenne pepper** . mix in the **cabbage** and **green onions** . cover , and refrigerate at least 1 hour before serving .
Ingredients	flour , butter , thyme , lucerne egg , lucerne milk , black pepper , garlic , salt , chive , roast , yorkshire pudding	
Constraints	None	thyme , butter , roast
Recipes	preheat oven to 325 degrees f . in a medium bowl , whisk together **flour** , **salt** , and **pepper** . in a large bowl , whisk together **eggs** , **milk** , and **garlic** . add flour mixture to flour mixture , whisking until just combined . pour into a 9-inch pie plate . bake in the preheated oven until a toothpick inserted into the center comes out clean , about 45 minutes . let stand for 10 minutes before slicing .	preheat oven to 350 degrees f . place the **roast** in a roasting pan . in a small bowl , whisk together the **butter** , **flour** , **salt** , **pepper** , and **thyme** . pour the egg mixture over the *roast* . bake for 1 hour , or until a meat thermometer inserted into the center of the roast registers 140 degrees f for medium-rare . remove the roast from the oven and let stand for 10 minutes before slicing . meanwhile , in a medium bowl , whisk together the **eggs** , **milk** , *butter* , and **garlic** . add the *flour* and whisk until smooth . pour the sauce over the roast .
Ingredients	olive oil , salt , honey , curry powder , the dressing , garbanzo bean , apple cider vinegar , raisin , quinoa , black pepper , water , salted butter , couscou , green onion , celery , carrot , vegetable soup mix , almond , turmeric powder , parsley , preserved lemon , vegetable stock	
Constraints	None	Low Carb
Recipes	bring **water** and **quinoa** to a boil in a saucepan . reduce heat to medium low , cover , and simmer until *quinoa* is tender and liquid has been absorbed , 20 to 25 minutes . heat **olive oil** in a skillet over medium heat ; cook and stir **onion** and **celery** until tender , about 5 minutes . stir in **couscous** , **carrots** , *celery* , and **raisins** . cook and stir until *quinoa* is tender , about 5 minutes . stir **vegetable stock** and **honey** into *quinoa* mixture ; bring to a boil . reduce heat to low and simmer until *quinoa* is tender , about 10 minutes . stir *cous* , *raisins* , and **parsley** into *couscous* mixture ; season with **salt** and **pepper** .	heat **olive oil** in a large saucepan over medium heat . add **carrots** , **celery** , and **onion** ; cook and stir until tender , about 5 minutes . stir in **quinoa** , **vegetable stock** , and **water** . bring to a boil . reduce heat to low , cover , and simmer until *quinoa* is tender , about 15 minutes . stir in **couscous** , **raisins** , and **parsley** . season with **salt** and **pepper** .

Figure 4: Examples of Controllable Recipe Generation.

which proves that our routing module is effective for content selection. What's more, compared to attention-based model **Attseq**, **RGM** alleviates the problem of inappropriately repeating words.

4.7 Controllable Recipe Generation

Our model is different from existing methods mainly in two aspects: routing based selection model and recipe generation in accordance with constraints (user demands). For all the results discussed above, **RGM** selects ingredients and generates the recipes following the routing weights. In this section, we assign the constraints of ingredients or categories as Figure 4 shows. In the first example, the generated recipe does not use "mustard". We constrain the ingredient by promoting the initial weight of it in the selective routing algorithm. As a result, **RGM** generates the recipe with expected mustard. Further, we conduct experiments on constraining a number of ingredients and the results confirm the validity of the selective routing algorithm. Considering the second example, **RGM** uses 6 ingredients we input without any constraints. If a user especially prefers some ingredients like:" thyme", "butter" and "roast", we set corresponding constraints on them. The generated recipe exactly contains the assigned ingredients. For people having special tastes or demands, they need dishes of certain categories, like: "Low Sugar". In the third example, **RGM** first generates the recipe following the maximum likelihood without any constraints. The generated recipe is a "High Fiber" one. We then give a constraint as "Low Sugar". The new recipe contains almost the same ingredients as before except for "honey". It is well known that "honey" is a sweet produced by bees and some related insects. It should not appear in a "Low Sugar" recipe. And the operation steps are also adjusted accordingly. The results of extended experiments show that **RGM** is able to generate reasonable recipes with user demands.

5 Conclusion and Future Work

In this paper, we make an effort on introducing routing algorithm to enforce the recipe generation model. We model the internal relationships between ingredients by selective routing algorithm. Given ingredients with noises, our model selects reasonable ingredient collocations and generates recipes based on user demands. Extensive experiments shows that the generated recipes are not only fluent and feasible, but also creative. There are several directions to explore in the future. For example, the clustering ability of routing algorithm can be used to control the style of generated texts.

Acknowledgements

This work was supported by National Natural Science Foundation of China (61772036), Beijing Academy of Artificial IntelligenceBAAI) and Key Laboratory of Science, Technology and Standard in Press Industry (Key Laboratory of Intelligent Press Media Technology). We appreciate the anonymous reviewers for their helpful comments. Xiaojun Wan is the corresponding author.

References

Regina Barzilay and Mirella Lapata. 2005. Collective content selection for concept-to-text generation. In *HLT/EMNLP 2005, Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference, 6-8 October 2005, Vancouver, British Columbia, Canada*, pages 331–338.

Regina Barzilay and Lillian Lee. 2004. Catching the drift: Probabilistic content models, with applications to generation and summarization. In *Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics, HLT-NAACL 2004, Boston, Massachusetts, USA, May 2-7, 2004*, pages 113–120.

Antoine Bosselut, Omer Levy, Ari Holtzman, Corin Ennis, Dieter Fox, and Yejin Choi. 2017. Simulating action dynamics with neural process networks. *CoRR*, abs/1711.05313.

David L. Chen, Joohyun Kim, and Raymond J. Mooney. 2010. Training a multilingual sportscaster: Using perceptual context to learn language. *J. Artif. Intell. Res.*, 37:397–435.

David L. Chen and Raymond J. Mooney. 2008. Learning to sportscast: a test of grounded language acquisition. In *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008*, pages 128–135.

Jingjing Chen and Chong-Wah Ngo. 2016. Deep-based ingredient recognition for cooking recipe retrieval. In *Proceedings of the 2016 ACM Conference on Multimedia Conference, MM 2016, Amsterdam, The Netherlands, October 15-19, 2016*, pages 32–41.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Geoffrey E. Hinton, Alex Krizhevsky, and Sida D. Wang. 2011. Transforming auto-encoders. In *Artificial Neural Networks and Machine Learning - ICANN 2011 - 21st International Conference on Artificial Neural Networks, Espoo, Finland, June 14-17, 2011, Proceedings, Part I*, pages 44–51.

Geoffrey E Hinton, Sara Sabour, and Nicholas Frosst. 2018. Matrix capsules with em routing.

Jermsak Jermsurawong and Nizar Habash. 2015. Predicting the structure of cooking recipes. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 781–786.

Masahiro Kazama, Minami Sugimoto, Chizuru Hosokawa, Keisuke Matsushima, Lav R Varshney, and Yoshiki Ishikawa. 2018. A neural network system for transformation of regional cuisine style. *Frontiers in ICT*, 5:14.

Chloé Kiddon, Luke Zettlemoyer, and Yejin Choi. 2016. Globally coherent text generation with neural checklist models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 329–339.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 110–119.

Percy Liang, Michael I. Jordan, and Dan Klein. 2009. Learning semantic correspondences with less supervision. In *ACL 2009, Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the AFNLP, 2-7 August 2009, Singapore*, pages 91–99.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1412–1421.

Bodhisattwa Prasad Majumder, Shuyang Li, Jianmo Ni, and Julian McAuley. 2019. Generating personalized recipes from historical user preferences. In *EMNLP*, pages 5975–5981.

Jonathan Malmaud, Earl Wagner, Nancy Chang, and Kevin Murphy. 2014. Cooking with semantics. In *Proceedings of the ACL 2014 Workshop on Semantic Parsing*, pages 33–38.

Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. 2016. What to talk about and how? selective generation using lstms with coarse-to-fine alignment. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 720–730.

Weiqing Min, Shuqiang Jiang, Jitao Sang, Huayang Wang, Xinda Liu, and Luis Herranz. 2017. Being a supercook: Joint food attributes and multimodal content modeling for recipe retrieval and exploration. *IEEE Trans. Multimedia*, 19(5):1100–1113.

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, LREC 2014, Reykjavik, Iceland, May 26-31, 2014.*, pages 2370–2377.

Shinsuke Mori, Tetsuro Sasada, Yoko Yamakata, and Koichiro Yoshino. 2012. A machine learning approach to recipe text processing. In *Proceedings of the 1st Cooking with Computer Workshop*, pages 29–34.

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 of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA.*, pages 311–318.

Paul Rozin, Claude Fischler, Sumio Imada, Allison Sarubin, and Amy Wrzesniewski. 1999. Attitudes to food and the role of food in life in the usa, japan, flemish belgium and france: Possible implications for the diet–health debate. *Appetite*, 33(2):163–180.

Sara Sabour, Nicholas Frosst, and Geoffrey E. Hinton. 2017. Dynamic routing between capsules. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 3859–3869.

Amaia Salvador, Michal Drozdzal, Xavier Giro-i Nieto, and Adriana Romero. 2019. Inverse cooking: Recipe generation from food images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10453–10462.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 2692–2700.

Semih Yagcioglu, Aykut Erdem, Erkut Erdem, and Nazli Ikizler-Cinbis. 2018. Recipeqa: A challenge dataset for multimodal comprehension of cooking recipes. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 1358–1368.

Min Yang, Wei Zhao, Jianbo Ye, Zeyang Lei, Zhou Zhao, and Soufei Zhang. 2018. Investigating capsule networks with dynamic routing for text classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 3110–3119.

Zichao Yang, Phil Blunsom, Chris Dyer, and Wang Ling. 2017. Reference-aware language models. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 1850–1859.