Coherence Improvement on Pre-Training Based Cooking Recipe Generation

Group 17053487 (SN:17053487, 17031832, 18137070, 20081688)

Abstract

There has been a growing interest in the automatic generation of cooking recipes thanks to the development of machine learning models and contributed novel datasets. However, there are some drawbacks of the existing recipe generation models. Sometimes some of the artificial cooking instructions do not progress logically and input ingredients are not correctly used. To address these problems, we fine-tuned BART (Lewis et al., 2019), modified the loss function to ensure correct input ingredients identification, and introduced a discriminate module to ensure the instructions coherence. Our final proposed model has outperformed the existing Generative Pretrained Transformer GPT-2 (Radford et al., 2019) qualitatively and quantitatively.

1 Introduction

Automatic generation of cooking recipes can help people find a recipe that is tailored to user-specified ingredients and even other requirements such as a dish with low GI value or containing high protein. Also, it can produce novel recipes by autocompleting the list of given food ingredients, making it possible to cook creatively and spark users' creativity.

Since the publication of the Recipe1M+ dataset (Marin et al., 2019), exploitation of culinary recipe data has become more popular. Some previous studies produced food recommendation systems utilising algorithms such as knowledge graphs (Haussmann et al., 2019), K-nearest neighbour (Bushra and Hasan, 2020), or food representation (Donghyeon et al.). Also, much research has been undertaken regarding cross-modal tasks, including recipe retrieval (Carvalho et al., 2018), image generation from text (El et al., 2019), and recipe generation from images (Salvador et al., 2019).

1.1 Research Question

Recipes are a type of long-form text with a sequence of ordered tasks. It is difficult to preserve contextual coherence in text generation tasks, especially when the generated recipe has many steps. Previous studies treated the generation of recipes as a causal language modelling problem. This is not ideal since it fails to capture long-term dependency between ingredients and directions late in the sequence. For this reason we looked into how conditional language generation models perform in this recipe generating task. We used the term 'coherence' to indicate how multiple sentences globally hold together, which is well-defined in (Cho et al., 2018).

To explore the performance of existing recipe generation models, we produced artificial recipes using 'RecipeNLG's Recipe Generator'(GPT2 based) which is a highly successful model example (an example shown in Figure 1). From testing this recipe generator, two major drawbacks were apparent. Firstly, this model commonly ignores some of the input ingredients and sometimes generates a list of ingredients containing the same item more than once. For example, as shown in Figure 1, the key input ingredient 'tomato' was not used in the cooking instructions, which contradicts the purpose of generating new recipes based on the given ingredients. Also, the 'Worcestershire sauce' was repeatedly listed three times in the generated 'Ingredients'. Secondly, the generated cooking instruction steps don't progress in a way that preserves contextual coherence. As shown in Figure 1, the logical progression between steps is impossible to follow. For example, from step 1 to 2, it asks to serve despite still being in the middle of the cooking process.

For this project, we aim to achieve:

- Develop a model that could correctly utilise the input by taking full usage of ingredients

Title	Oven
Input Ingredients	• tomato • eggs
Ingredients	• 5 eggs yolks • 3 c. Worcestershire sauce • ¼ c. chopped mushrooms • ¼ c. water • Pepper • ½ c. Worcestershire sauce • ¼ tsp. Worcestershire sauce
Instructions	1. Put into a saucepan; add fat down to the water; mix in gravy; add Worcestershire sauce. 2. Serve over water mix and simmer to prevent boiling through. 3. Pour into a eat brown sugar and vegetables. Place in greased bowl to make a bouillon or half. 4. Add flour in top. 5. Season with melted butter and stir. Bake 35 minutes. 6. When 2 hours. 7. Chill ¼ c. water. 8. Bake at 400° for 15 minutes or until golden brown.

Figure 1: An artificial recipe example obtained from the 'RecipeNLG's Recipe Generator.

given in its generation process.

- Improve the global coherence of instruction steps compared to a currently available model (RecipeNLG model (Bień et al., 2020)).

1.2 Main Outcomes

To achieve the aims mentioned above, in this report, we fine-tuned BART (Lewis et al., 2019) and its variants on the dataset 'RecipeNLG', which is a refined version of the 'Recipe1M+' dataset (Marin et al., 2019). The main outcomes that we had were,

- We compared the recipe generation performance between GPT2 (Radford et al., 2019) and BART (Lewis et al., 2019) under the structure of causal language modeling. An improvement was found evaluated on cosine similarity, BLEU and ROUGE.

- We investigated the possibility of structuring the recipe generation task as a conditional generation problem. A fine-tuned BART (Lewis et al., 2019) with encoder-decoder structure is used to showcase the feasibility of generating culinary recipes.

- We propose a novel way to improve the logical coherence of instruction steps by introducing sample shuffling, sentence embedding and coherence discriminate modules to the model. Also, the loss function of generation is modified to ensure the correct use of input ingredients and explicitly penalise incoherent instructions.

2 Related Work

Early studies on recipe generation vary from case-based planners (Hammond, 1986) to directed acyclic graphs (DAG) (Mori et al., 2014). Recent works use recurrent neural networks (RNN) (Kiddon et al., 2016a) and other neural network architectures. For example, (Majumder et al., 2019) built an encoder-decoder framework using attention mechanisms (Vaswani et al., 2017) which had three embedding layers: menu name, ingredient, and caloric-level embedding, which are passed to two-layered bidirectional GRUs (BiGRU)(Cho et al., 2014). (Yang et al., 2017) proposed a reference-aware language model using an LSTM encoder and an attention based decoder to generate recipes. More recent literatures often apply transformer-based language models to tackle the task because transformers are good at capturing dependencies and generating grammatically correct and complex sentences. In (H. Lee et al., 2020), a Generative Pre-trained Transformer (GPT) architecture was used. The authors proposed a system able to generate instructions from given ingredients and vice versa. This is different from our work. We focused on generating complete recipes provided with incomplete ingredients (see Section Method). It allows us to compare the ground truth recipe with our model result given a full list of the ingredients. (Bień et al., 2020) also exploits GPT2 architecture but its task is more similar to ours. It also used partial ingredients to generate auto completed recipes. The main difference between our work and (Bień et al., 2020) is that it does not use BART (Lewis et al., 2019) which we believe can enhance the model performance. The closest work to ours (Geluykens et al., 2021) used BART (Lewis et al., 2019) to address the procedure generation task. However, they used more information in its input. It used a full list of ingredients with quantity and units included. Thus the effectiveness of BART (Lewis et al., 2019) in cooking recipe generation using an incomplete list of ingredients has not yet been explored.

Studies have shown that sequential models tend to focus more on the recent input and put less attention to the long-range context (Khandelwal et al., 2018). This phenomenon makes generating coherent long-form texts like recipe more challenging. Former studies attempted to ensure global coherence by creating a generation checklist, and dynamically updating the generation conditions (Kiddon et al., 2016b). Some methods train discriminators

to reward well generated texts by evaluating coherence. The rewards are generated by a discriminator using GAN (Cho et al., 2018) or with reinforcement learning modules (Bosselut et al., 2018).

Some publications manage to improve the generation quality by introducing sentence ordering objectives when training the model, and consequently ensure the coherence of generation. ALBERT (Lan et al., 2019) uses a loss function based on a binary classification task by predicting whether two segments of text are consecutive. Skip-thought (Kiros et al., 2015) and Quick-thought (Logeswaran and Lee, 2018) models are trained to discriminate different orders of sentences, and the hidden state vectors are treated as the sentence embedding.

3 Methods

To improve the on the performance of GPT2, after extensive research, we expected BART to be a better alternative. The bidirectional and autoregressive Transformer model (BART (Lewis et al., 2019)) is a denoising autoencoder for pretraining sequence-to-sequence models, which in addition to a decoder structure comparable to GPT2 it also has a bidirection encoder. As in Figure 2, the architecture is of a (12-layer) bidirectional encoder (like BERT) with a (12-layer) auto-regressive decoder (like GPT). It is pretrained by learning a model to reconstruct the original input after text is corrupted arbitrarily and finetuned on a downstream task afterwards. We used a machine translation framework under BART (Lewis et al., 2019) where an incomplete list of ingredients is the source language and the target language is a list of all required ingredients with corresponding quantities and a sequence of cooking instructions.

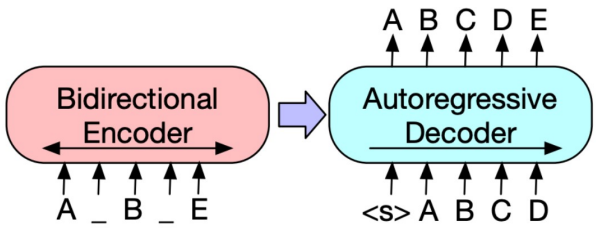

Figure 2: BART model (Lewis et al., 2019)

3.1 Model Structure

The overview of the proposed model is given in Figure 3.

Our approach treated the recipe generation as a conditional generation problem. We encoded the ingredients as the condition and fed it to the BART encoder. Meanwhile, the BART decoder was required to generate the recipe instructions.

Differing from common text generation methods, we proposed to use a Coherence Discriminate Module (CDM) to correctly identify sentence order and check the logic coherence of output texts. More specifically, the last hidden state of the BART decoder was fed to this module for a binary classification task.

The model was designed to complete two tasks: the text generation and the coherence discrimination. The generation loss and the discriminate loss were combined and the model was jointly trained based on this combined loss.

3.2 Conditional Generation for Recipe

In the previous study of (Marin et al., 2019), the generation of recipes was structured as a causal language modelling problem. Specifically, the model was asked to learn a conditional distribution of next-word output w_i, that is $P(w_i \mid w_1, \ldots, w_{i-1})$. The distribution was conditioned on the input words or previously generated text $\{w_1, \ldots, w_{i-1}\}$.

For conditional generation problems, other than the previous text, an extra condition C was input to control the output. Thus it is aiming to estimate,

$$P(w_i \mid w_1, \ldots, w_{i-1}, C)$$

In this report, we managed to generate recipes from a conditional generation perspective, and used the BART model's (Lewis et al., 2019) encoder-decoder framework to process the input. The input of data were divided into 2 parts, the ingredients and the instructions. The ingredients part worked as the generation conditions, which was embedded by the BART encoder. The instructions were then fed into the decoder side which then output the predicted token. The data preprocessing is further discussed in section 4.

3.3 Coherence Discriminate Module

Based on the vanilla BART model (Lewis et al., 2019), we proposed the use of a binary classification module to provide a feedback signal on the value of coherence in a generated text. This module introduced a penalisation on the garbled instruction order and encouraged the generated instructions to be more coherent and consistent.

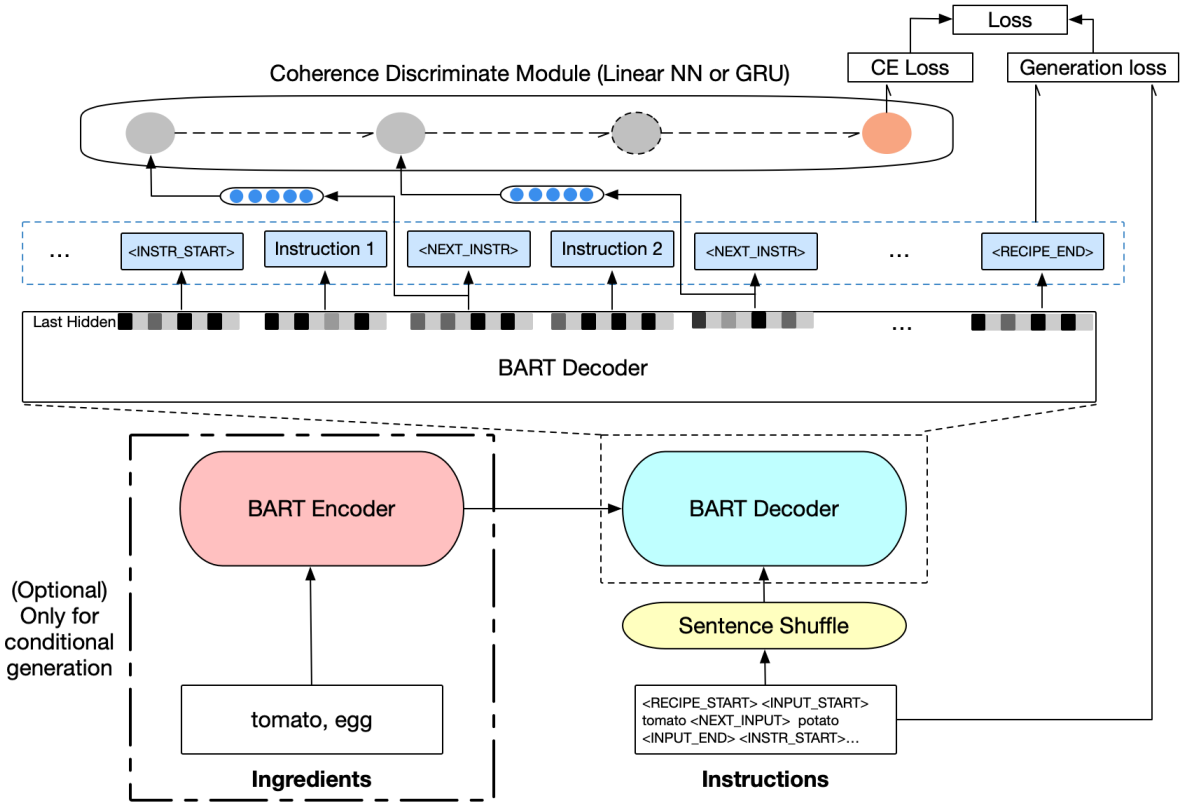

Figure 3: BART with CDM and Modified Loss

We were inspired by Quick-Thought (Logeswaran and Lee, 2018) encoding a sentence to an embedding and we generalised it to process a series of sentences. However, in our approach, we were not attempting to obtain an accurate sentence embedding, but to train a linear encoder that can capture the coherence feature in consecutive segments of the text. The output of the encoder would contain the coherence information, and the following classification module would discriminate the coherence of the generated text. This procedure is visualized in Figure 2.

Our first attempt was to use a simple 2-layer fully-connected, multiple-layer perceptron (MLP) with a sigmoid activation function. Assuming the instruction text size to be N and the last hidden state size to be H, the MLP input size is $N \times H$. And the output is either 1 or 0.

There are two clear drawbacks of MLP: it is not adaptable to the text length and it cannot capture the sequence feature. Thus, a GRU model was used as the coherence discriminate module. In ALBERT (Lan et al., 2019), the hidden state of a special <CLS> token was used to perform sentence order prediction tasks. In our work, the hidden state of <NEXT_INSTR> tokens are trained to represent the coherence feature. A GRU was then used to capture the global coherence feature, as shown in Figure 3.

3.4 Training and Loss Modification

As we had two training objectives, we divided the training procedure into 2 rounds. In the first round, we fine-tuned BART (Lewis et al., 2019) and trained the CDM jointly. In the second round, we fixed the CDM and only fine-tuned BART (Lewis et al., 2019). The overall progress can be modelled as a multi-task learning problem, with two objectives: text generation and coherence checking.

For text generation, we investigated the influence of splitting the total loss into two parts: the ingredient loss ℓ_{ing} and the instruction loss ℓ_{ins}. Our idea was to put more weight on the ingredient loss ℓ_{ing} to ensure it correctly predicted the ingredients, without randomly using unknown ingredients or ignoring some input ingredients.

The coherence checking was a binary classification problem. We used 0 to represent coherent text and 1 for incoherent, and used cross-entropy loss (CE loss) to measure the difference between the predicted value and the label. The label would be 0 if the sample was not shuffled and 1 otherwise. The

loss is simply a binary cross-entropy loss, ℓ_{CDM}. A similar training procedure as in ALBERT (Lan et al., 2019) was applied.

Thus, the total loss was:

$$\ell_{total} = \alpha\ell_{ing} + \ell_{ins} + \lambda\ell_{CDM}, where\ \alpha > 1$$

The weight hyperparameters α and λ were chosen by fitting multiple models and selecting the best combinations.

4 Data Preprocessing

We performed our experiments on the dataset 'RecipeNLG'(Bień et al., 2020), which contains 2,231,142 cooking recipes. Each recipe consists of a title, ingredients, and instructions. The step-by-step nature of the instructions makes the data suitable for semi-structured text generations. An example of the dataset is shown in Figure 4.

Before input to the model, the recipes were broken down into two parts: ingredients and instructions. The ingredients part is the raw text taken directly from the dataset and the instructions were preprocessed by adding control tokens. In the generation process, the model first complements the ingredients by adding necessary condiments and the quantity of ingredients needed, and then generates the instructions step by step.

In several samples, the order of instructions were intentionally shuffled, which was aimed to break the coherence of texts.

title	Best Ever Baked Beans
ingredients	• 5 pounds beans with pork, canned • 1/2 cup onions chopped • 1/2 cup celery chopped • 13 cup sweet red bell peppers chopped ……
NER	• Beans • Onions • Celery • sweet red bell peppers …….
instructions	• Combine all ingredients, except in bacon, in large ovenproof container. • Lay bacon strips on top. • Place on smoker grid and smoke for 2 to 2 1/2 hours.

Figure 4: Data structure of RecipeNLG (Bień et al., 2020)

4.1 Data Cleansing

For the purpose of training stability, four rules were set to remove undesired recipe samples:

- 'Titles' should contain more than four characters.
- 'Instructions' should not contain 'mix all'.
- 'Instructions' should have at least two sentences.
- 'Instructions' should have at least 30 characters, and the tokenised vector should be no longer than 512 tokens.

Recipe samples that did not comply with the standards were removed from the dataset.

4.2 Control Tokens

In the recipe samples, 'Ingredients' were treated as the encoder-side input (equally as the condition within the conditional generation), and 'Instructions' was treated as the generation target. Control tokens (Figure 5) were added to the directions and used to help the model to understand the recipe samples' underlying structures. The tokenised instructions with length shorter than 512 were padded with the token <RECIPE_END> for training purposes.

Token	Description
<INPUT_START>	Beginning of user input - from predefined list
<NEXT_INPUT>	Next element of input list
<INPUT_END>	End of user input
<TITLE_START>	Beginning of recipe title
<TITLE_END>	End of recipe title
<INGR_START>	Beginning of generated ingredients list
<NEXT_INGR>	Next element of generated ingredients list
<INGR_END>	End of last generated ingredient
<INSTR_START>	Beginning of first cooking instruction
<NEXT_STEP>	Next cooking instruction
<INSTR_END>	End of last cooking instruction
<RECIPE_START>	Start of the whole recipe
<RECIPE_END>	End of recipe. Works as end of text delimiter

Figure 5: Control Tokens (Bień et al., 2020)

An example of the input for conditional generation is given in Figure 6. We tested two types of input for the encoder: one with special splitting tokens <NEXT_INPUT> between the input ingredients and one without. We decided to drop the splitting tokens to avoid gradient overflow.

For the causal language modelling, we did not need the encoder-side input. The input format is given in Figure 7.

4.3 Text Encoding

After the data had been cleaned and special tokens were added, the data was tokenised by BART-base's tokeniser and stored as a tokenised H5 database. This was later used directly in the training process to give fast access to the dataset.

Encoder input (Ingredients)	beef <INPUT_END>... <INPUT_END>
Decoder input (Instructions)	<RECIPE_START> <INPUT_START> beef <NEXT_INPUT> <INPUT_END> <INSTR_START> Combine all ingredients. <NEXT_INSTR> ... <INSTR_END> <TITLE_START> Spicy Stuffed Peppers <TITLE_END> <RECIPE_END>
Target	<INGR_START> 3/4 lbs. lean beef <NEXT_INGR> ... <INGR_END> <INSTR_START> Combine all ingredients. <NEXT_INSTR> ... <INSTR_END> <TITLE_START> Spicy Stuffed Peppers <TITLE_END> <RECIPE_END>

Figure 6: Input example for Conditional Generation

Input	<RECIPE_START><INPUT_START> beef <NEXT_INPUT> <INPUT_END>
Target	<INGR_START> 3/4 lbs. lean beef <NEXT_INGR> ... <INGR_END> <INSTR_START> Combine all ingredients.<NEXT_INSTR> ... <INSTR_END> <TITLE_START> Spicy Stuffed Peppers <TITLE_END> <RECIPE_END>

Figure 7: Input example for CLM

4.4 Recipe Instruction Shuffling

While preparing our data, we randomly shuffled the order of instruction steps within each recipe sample. Then an extra signal variable was added to represent whether this sample was shuffled for the following training (1 for being shuffled and 0 otherwise). Those shuffled samples were treated as incoherent samples in the training. This preprocessing method would not intensely influence the generation procedure because the feature persists within a sentence. In addition, each instruction step was explicitly cut off by a special token <NEXT_INSTR>, which would help the model to reserve the local coherence within a sentence. Meanwhile, this shuffling method also forced the model to learn how a coherent recipe should be organised globally.

5 Experiments

We conducted extensive experiments to evaluate our proposed model, and tested each component independently. Our results are summarised in Table 1 and an example is presented in Figure 8.

5.1 Baselines

We trained five baseline models as follows.

1. **GPT2**: GPT2 is an auto-regressive model i.e. the output will be added to the sequence of input to generate the next token. It does not have an encoder block, which is the major difference between it and BART for conditional generation (Lewis et al., 2019).

2. **BART1** (*BART for causal language models*): Only the decoder of vanilla BART (Lewis et al., 2019) without a CDM module (described in Section 3) is used to do generation. The input format is the same as GPT2.

3. **BART2** (*BART (Lewis et al., 2019) for conditional generation*): The vanilla BART without a CDM module described in Section 3.

4. **BART1+MLP**: BART1 combined with a 2-layer MLP based CDM

5. **BART1+GRU**: BART1 combined with a GRU based CDM.

6. **BART2+GRU**: BART2 combined with a GRU based CDM.

GPT2 and BART1 were trained as causal language models and their input follows the form in Figure 7. The BART2 is a conditional generation model with the input form in Figure 6.

5.2 Experiment Setup

Due to the limitation of computational resources (colab pro only allows for 24 hours processing time), we decided to form a dataset of 350,000 training samples and another 100 testing samples. The idea of picking 100 golden standard testing samples was taken from (Bień et al., 2020), we initially also suspected this to be a small sample size. We then realised that recipe generation is an open-ended task with only minor information fed into the model. So rather than blindly increasing the test sample size, it could be more reasonable to generate multiple recipes under one input. This could remarkably avoid the problem of experimental contingency, and give a rational evaluation of the model performance.

To make our model comparisons meaningful, we used the 'BART-base' model that our proposed model also uses to construct different models which can be found in Table 1. Because of this, we frequently suffered from the problem of gradient and memory outflow. Hence we were forced to set batch size to be 6 and automatic loss scaling was applied to make the gradient descent smooth.

Original Input Ingredients	Full Input Ingredients [a]	Instructions Generated by GPT2	Full Input Ingredients [b]	Instructions Generated by BART1+GRU
Requirements • Tomato • Beef • Potato	**Requirements** • 1 tomato packed with water • 1 potato packed with water • 6 cup beef broth • 2 tablespoons Dijon mustard • 1 cup dry white wine	**Target Product** Tomato Tomato Compote **Tasks** ▪ Remove from grill. ▪ Peel plate and thread tomato. ▪ Peel and dice. ▪ Mix tomatoes, and water, and draft through potato until appearancebrates are thoroughly handled. ▪ Season as desired. ▪ Ready to serve and serve with sauce on hand.	**Requirements** • 1 tomato • 10 potato (smoked) • beef (smoked)	**Target Product:** Molbeleannoapi-Kitty di Pulver (Mock-Finaglio) **Tasks** ▪ Fry diced tomato. ▪ Debone beef and dice potato. ▪ Put meat on a large board, cut pan in a blender; add half the canned tomatoes and mix. ▪ Push the pot to the side and set aside until all the ingredients have been mixed up. ▪ In the meantime, by the time all ingredients are mixed up, grill the potato. ▪ When all sides of potatoes are white and crispy, remove with a large spoon. ▪ Let sit in a bowl 10 minutes. ▪ When all potatoes are cooked, add fat to taste. ▪ Optional: generously sprinkle the remaining tomatoes and pineapple with some salt and pepper to taste. ▪ Serve warm.

Figure 8: Recipes Generated by GPT2 vs. BART1+GRU

Model	**Cosine Similarity**	**BLEU-1**	**ROUGE-1**	**ROUGE-L**
GPT2	0.539	0.770	0.301	0.176
BART1[a]	0.535	0.761	0.309	0.186
BART2[b]	0.503	0.672		
BART1+MLP	0.534	0.771	0.310	0.185
BART1+GRU	**0.564**	0.772	**0.357**	**0.219**
BART2+GRU	0.546	**0.786**	0.346	0.212

[a] BART1 for causal language model, [b] BART2 for conditional generation

Table 1: Evaluation results for 4 trained BART models and the GPT2 model

5.3 Evaluation and results

Among the recipe samples not used in the model training, 100 of them were selected at random to form the testing set. Evaluation of the four trained BART models (Lewis et al., 2019) and comparison against the GPT2 model were done by generating 1000 cooking instructions with each model. These cooking instructions were generated using the reference ingredients from the recipe samples, with each model generating 10 new cooking instructions for each recipe sample.

Four evaluation approaches were utilised. Firstly, cosine similarity was calculated with a TF-IDF representation to measure the similarity between the generated cooking instructions and the instructions from the corresponding recipe sample in the testing set. Secondly, standard evaluation metrics 1-gram BLEU and ROUGE-1 were calculated for each model by comparing word-for-word similarity between the generated cooking instructions and the instructions from the corresponding recipe samples. ROUGE-L was used to measure the sentence-level similarity. The results of these evaluations are summarised in Table 1.

5.4 Qualitative Review

A selected recipe example each generated by GPT2 and BART1+GRU are presented in Figure 8.

In this example, we have input the same ingredients for both models (as control). A full list of required ingredients was generated alongside cooking instructions by each model.

Looking into the instructions generated by GPT2, the recipe does not follow clear logic. Immediately, the first instruction step is nonsensical and they progress incoherently. In contrast, the recipe generated by BART1+GRU follows clear logic and does not violate any common sense. It brings a very obvious improvement on the coherence of recipes generated.

The example given was hand selected, however it is representative of the performance of the GPT2 model during testing with varying input ingredients. The BART1+GRU model performed better in most cases.

5.5 Discussion

Being trained and evaluated on the same dataset, GPT2 and BART1 (BART for causal language modelling) have similar evaluation scores. This is expected because both GPT2 and BART1 use the same decoder only structure.

Our proposed approach (BART1 with a discriminate module of GRU) grants an improvement in all evaluation metrics, especially in the ROUGE score. This result suggests that our model has the ability to capture the coherence feature and the modified training loss has the potency to better supervise text generation. However, the performance using a simple MLP is not satisfactory. This suggests that only sequential models could successfully capture the sentence coherence feature, and the adaptability to sentence length does matter.

It is surprising that the conditional generation model (BART2) (Lewis et al., 2019) does not bring improvement to the recipe generation in terms of Cosine Similarity and BLEU. We suspect that this is caused by two reasons,

- **Unseen tokens**: We look into the input tokens of BART encoder and decoder, and find some of the ingredient tokens are not consistent with the corresponding ones in the instructions. Because most ingredients rarely appear in the pretraining vocabulary of BART, the tokens of these ingredients are rather arbitrary. For example, an ingredient 'okra' would be decomposed into two tokens by the BART tokeniser, 'ok' and 'ra' as it never appears in the pretraining procedure. Even though it can be reformed as a whole word, the underlying meanings would be completely different. Additionally, 'ok' within different contexts would also have various tokens adding further confusion. So sometimes the condition in the encoder side cannot be helpful with text generation, or even be harmful.

- **Underfitting**: Due to the limitation of computation resources, we were limited to use only 350,000 samples in the dataset for 24 hours training. This poor performance might be caused by the underfitting of the model since the training of BART2 was much harder with more parameters. We can only confirm this by further study.

6 Conclusions and Future Work

In this report, by considering the recipe generation problem as a machine translation task, we examined the feasibility of using a conditional generation model to generate recipes. We also proposed a Coherence Discriminate Module (CDM) that helps improve coherence in the instructions of generated recipes. We found that BART2 (Lewis et al., 2019) (BART for conditional generation model) does not perform well (even combined with CDM) and BART1 (BART for causal language model) with a GRU based CDM achieved the best performance among all models we constructed.

6.1 Future work

As mentioned in the discussion section, the tokenisation of unseen ingredients would largely influence the model performance. Thus, to avoid the problem of mistakenly tokenising these ingredients and further improving the performance of recipe generation, a customised tokeniser that could identify the ingredients should be constructed.

In theory, for recipe generation tasks we expect conditional generation models would outperform causal language models because it has input ingredients as an extra condition to generate recipes. However, we got a result that is different from our expectation. We suspect it is because both parameters of encoder and decoder in BART2 (BART for conditional generation model) need to be trained and therefore it requires more data to be fine tuned to a better result. While for BART1 (BART for causal language model) only the decoder needs to be fine tuned and thus gets a better result than BART2. Due to time and computational resource limitation, we only used 350,000 recipe samples to fine tune models which may not be sufficient. For example (Bień et al., 2020) used 2,231,142 recipe samples to fine tune GPT2. Hence one area of future work could be to fine tune BART2 (conditional generation model) on a larger dataset.

Again due to time and computational resource limitation, we were not able to do random or grid-search for optimal hyperparameter combinations for the coefficients of losses α and λ. Another future work would be to find (local) optimal hyperparameters to improve the model performance.

References

Michał Bień, Michał Gilski, Martyna Maciejewska, Wojciech Taisner, Dawid Wisniewski, and Agnieszka Lawrynowicz. 2020. RecipeNLG: A cooking recipes dataset for semi-structured text generation. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 22–28, Dublin, Ireland. Association for Computational Linguistics.

Antoine Bosselut, Asli Celikyilmaz, Xiaodong He, Jianfeng Gao, Po-Sen Huang, and Yejin Choi. 2018. Discourse-aware neural rewards for coherent text generation. *arXiv preprint arXiv:1805.03766*.

Naila Bushra and Mehedi Hasan. 2020. Quicklycook: A user-friendly recipe recommender. In *Machine Learning and Metaheuristics Algorithms, and Applications*, pages 245–254, Singapore. Springer Singapore.

Micael Carvalho, Rémi Cadène, David Picard, Laure Soulier, Nicolas Thome, and Matthieu Cord. 2018. Cross-modal retrieval in the cooking context: Learning semantic text-image embeddings. In *The 41st International ACM SIGIR Conference on Research amp; Development in Information Retrieval*, SIGIR '18, page 35–44, New York, NY, USA. Association for Computing Machinery.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation.

Woon Sang Cho, Pengchuan Zhang, Yizhe Zhang, Xiujun Li, Michel Galley, Chris Brockett, Mengdi Wang, and Jianfeng Gao. 2018. Towards coherent and cohesive long-form text generation. *arXiv preprint arXiv:1811.00511*.

Park Donghyeon, Kim Keonwoo, Kim Seoyoon, Michael Spranger, and Kang Jaewoo. Flavorgraph: a large-scale food-chemical graph for generating food representations and recommending food pairings. *Scientific Reports (Nature Publisher Group)*, 11(1).

Ori Bar El, Ori Licht, and Netanel Yosephian. 2019. Gilt: Generating images from long text.

Joppe Geluykens, Sandra Mitrović, Carlos Eduardo Ortega Vázquez, Teodoro Laino, Alain Vaucher, and Jochen De Weerdt. 2021. Neural machine translation for conditional generation of novel procedures. In *Proceedings of the 54th Hawaii International Conference on System Sciences*, page 1091.

Helena H. Lee, Ke Shu, Palakorn Achananuparp, Philips Kokoh Prasetyo, Yue Liu, Ee-Peng Lim, and Lav R. Varshney. 2020. Recipegpt: Generative pre-training based cooking recipe generation and evaluation system. In *Companion Proceedings of the Web Conference 2020*, WWW '20, page 181–184, New York, NY, USA. Association for Computing Machinery.

K. Hammond. 1986. Chef: A model of case-based planning. In *AAAI*.

Steven Haussmann, Oshani Seneviratne, Yu Chen, Yarden Ne'eman, James Codella, Ching-Hua Chen, Deborah L. McGuinness, and Mohammed J. Zaki. 2019. Foodkg: A semantics-driven knowledge graph for food recommendation. In *The Semantic Web – ISWC 2019*, pages 146–162, Cham. Springer International Publishing.

Urvashi Khandelwal, He He, Peng Qi, and Dan Jurafsky. 2018. Sharp Nearby, Fuzzy Far Away: How Neural Language Models Use Context. In *Association of Computational Linguistics (ACL)*.

Chloé Kiddon, Luke Zettlemoyer, and Yejin Choi. 2016a. Globally coherent text generation with neural checklist models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 329–339, Austin, Texas. Association for Computational Linguistics.

Chloé Kiddon, Luke Zettlemoyer, and Yejin Choi. 2016b. Globally coherent text generation with neural checklist models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 329–339.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-thought vectors. *arXiv preprint arXiv:1506.06726*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension.

Lajanugen Logeswaran and Honglak Lee. 2018. An efficient framework for learning sentence representations. *arXiv preprint arXiv:1803.02893*.

Bodhisattwa Prasad Majumder, Shuyang Li, Jianmo Ni, and Julian McAuley. 2019. Generating personalized recipes from historical user preferences.

Javier Marin, Aritro Biswas, Ferda Ofli, Nicholas Hynes, Amaia Salvador, Yusuf Aytar, Ingmar Weber, and Antonio Torralba. 2019. Recipe1m+: A dataset for learning cross-modal embeddings for cooking recipes and food images.

Shinsuke Mori, Hirokuni Maeta, Tetsuro Sasada, Koichiro Yoshino, Atsushi Hashimoto, Takuya Funatomi, and Yoko Yamakata. 2014. Flow-Graph2Text: Automatic sentence skeleton compilation for procedural text generation. In *Proceedings of the 8th International Natural Language Generation Conference (INLG)*, pages 118–122, Philadelphia, Pennsylvania, U.S.A. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Amaia Salvador, Michal Drozdzal, Xavier Giro i Nieto, and Adriana Romero. 2019. Inverse cooking: Recipe generation from food images.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.

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*, pages 1850–1859, Copenhagen, Denmark. Association for Computational Linguistics.