RecipeGPT: Generative Pre-training Based Cooking Recipe Generation and Evaluation System

Helena H. Lee
Singapore Management University
helenalee@smu.edu.sg

Ke Shu
Singapore Management University
keshu@smu.edu.sg

Palakorn Achananuparp
Singapore Management University
palakorna@smu.edu.sg

Philips Kokoh Prasetyo
Singapore Management University
pprasetyo@smu.edu.sg

Yue Liu
Singapore Management University
yueliu@smu.edu.sg

Ee-Peng Lim
Singapore Management University
eplim@smu.edu.sg

Lav R. Varshney
University of Illinois at Urbana-Champaign
varshney@illinois.edu

ABSTRACT

Interests in the automatic generation of cooking recipes have been growing steadily over the past few years thanks to a large amount of online cooking recipes. We present RecipeGPT, a novel online recipe generation and evaluation system. The system provides two modes of text generations: (1) instruction generation from given recipe title and ingredients; and (2) ingredient generation from recipe title and cooking instructions. Its back-end text generation module comprises a generative pre-trained language model GPT-2 fine-tuned on a large cooking recipe dataset. Moreover, the recipe evaluation module allows the users to conveniently inspect the quality of the generated recipe contents and store the results for future reference. RecipeGPT can be accessed online at https://recipegpt.org/

CCS CONCEPTS

• **Information systems → Web applications**; • **Computing methodologies → Natural language processing**; • **Applied computing → Consumer health**.

KEYWORDS

recipe generation, natural language generation, web application

ACM Reference Format:
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 Companion), April 20–24, 2020, Taipei, Taiwan.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3366424.3383536

WWW '20 Companion, April 20–24, 2020, Taipei, Taiwan

ACM ISBN 978-1-4503-7024-0/20/04.
https://doi.org/10.1145/3366424.3383536

1 INTRODUCTION

Automatic generation of cooking recipes is an interesting and practical research problem that can help overcome the limitations of standard recipe retrieval systems. Though many online recipe databases, such as Allrecipes and Yummly, allow users to explicitly include and exclude specific ingredients in the recipe search, the users have to use an advanced search interface which can be difficult to understand for novice users. Recipe generation systems can facilitate this process by directly generating cooking instructions for a specific recipe given a list of user-specified ingredients. Next, it can also be used for creative cooking (e.g., IBM Chef Watson [7]), where the system helps the users to come up with novel and practical ways for cooking certain dishes, taking into account the complementarity of ingredients.

A few approaches to recipe text generation have been proposed, such as knowledge-based models [7] and deep neural networks models [1, 3, 6]. Large-scale transformer-based language models have been shown to outperform Recurrent Neural Networks (RNNs) in several natural language processing (NLP) tasks lately. In text generation, transformers are known for their effectiveness in capturing complex dependencies and generating fluent sentences. Among those, OpenAI's GPT-2, pre-trained on a gigaword-scale textual dataset, has achieved impressive results in a variety of text generation tasks [5]. Recent study has also shown that fine-tuning GPT-2 can result in better performance on domain-specific text generations [10]. However, the effectiveness of pre-trained transformer-based language models in cooking recipe generation has not yet been explored.

Similar to other text generation tasks, evaluating the quality of machine-generated recipe texts is challenging. First, most neural text generation models are non-deterministic, thus each generation run produces unique results which are difficult to replicate. Second, machine-evaluation metrics and human evaluation on text generations are not well correlated. Third, a lot of efforts are required to judge the content coverage in the generated text. Finally, the adaptability of the model has not been well-studied, for example, generating cooking instructions from novel inputs of recipe title and ingredient combinations.

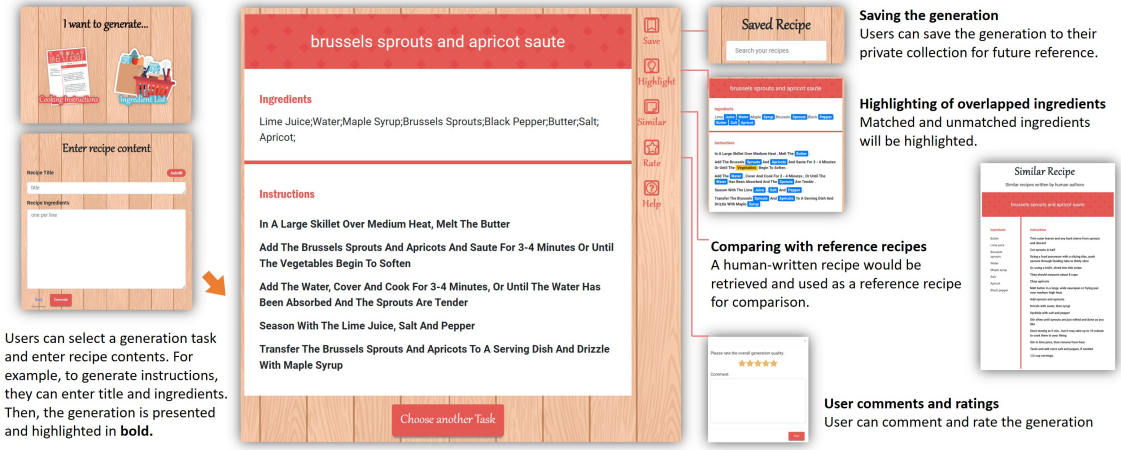

Figure 1: Overview of RecipeGPT

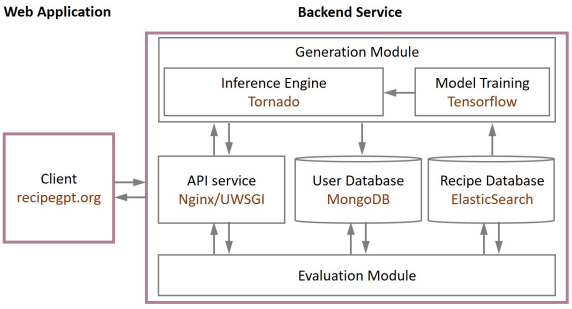

Figure 2: System Architecture

In this paper, we introduce RecipeGPT, a novel web application for recipe generation and evaluation, to demonstrate the feasibility of generative pre-trained transformer in cooking recipe generation and to assist users in evaluating the generation quality more easily as illustrated in Figure 1. Users can utilize RecipeGPT to: 1) generate cooking instructions according to given recipe title and ingredient texts; and 2) generate ingredients given recipe title and cooking instruction texts. The system allows users to rate, comment, and store the generated results. Furthermore, the system also helps the users compare machine-generated recipes with similar human-written recipes retrieved from the recipe database.

2 SYSTEM OVERVIEW

Figure 2 shows the system architecture of RecipeGPT. First, the user client is developed as a web application (presented in Figure 1). Next, the backend service provides core RESTFul web services which can be used by clients with an API key. These services include accepting user inputs, handling user interactions, and returning the generated results. More specifically, the backend service is composed of two following modules:

2.1 Generation Module

The generation module relies on a generative pre-trained transformer GPT-2, fine-tuned on Recipe1M dataset [4]. We utilize the fine-tuned model to perform two tasks: ingredient generation and instruction generation. The details of model training are described in Section 3. The generation model is trained using Tensorflow and deployed to the inference engine to handle live requests. Then, utilizing an API service built upon Nginx and UWSGI server, the requests are encapsulated and sent to the inference engine accelerated by GPU to increase the request throughput. Lastly, MongoDB is used as a data storage to optionally save the generated outputs for future reference and inspection.

2.2 Evaluation Module

The aim of the evaluation module is to provide functionalities to assist the users in individually inspecting the quality of the generated recipe texts. These are implemented in the following features:

Highlighting of overlapped ingredients. To quickly check the quality of the generated recipe texts, users may want to compare the overlap of ingredient words in the specified recipe contexts and the generated texts. High-quality generations are those that have the highest degree of overlapped ingredients, i.e., all ingredient words specified by the users appear in the generated texts. RecipeGPT facilitates the inspection of overlapped ingredients through a word highlighting feature. Since recipe authors tend to use different word variations to refer to the same ingredients (e.g., simply *cheese* instead of *provolone cheese*), we only consider root nouns of ingredients in the comparison.

Comparing with reference recipes. For each generation run, RecipeGPT also retrieves the most similar recipe given the specified recipe contexts (e.g., title, ingredients, etc.) from Recipe1M, stored in ElasticSearch data storage, to be used as a reference to compare against the generated recipe texts. We use ElasticSearch's built-in search functions for ranking recipes by similarity.

User comments and ratings. Lastly, ReciptGPT provides basic annotation functions, such as user commenting and rating, to facilitate human evaluation of the generated recipe texts. The annotation data are stored together with the user-saved recipes.

3 TRAINING RECIPE GENERATION MODEL

In this section, we describe in detail the steps to build our recipe generative pre-training transformer, one of the core components of the recipe generation module, by fine-tuning GPT-2 on Recipe1M

dataset. These involve (1) preprocessing of Recipe1M data; and (2) training multi-field recipe generation model.

3.1 Data Preprocessing

First, we filter out numerals, quantity words, and other comment texts in the ingredients using an ingredient phrase parser[1] and our own rule-based filtering. In addition, we remove recipes that contain non-ingredients and non-instructional sentences (e.g., nutritional or author information). Finally, we keep the remaining recipes with at least 2 ingredients, 2 instructional sentences, and 20 words in the instructions (N = 904,401). From those, two disjointed sets of 4,000 recipes each are reserved for the validation set and the testing set.

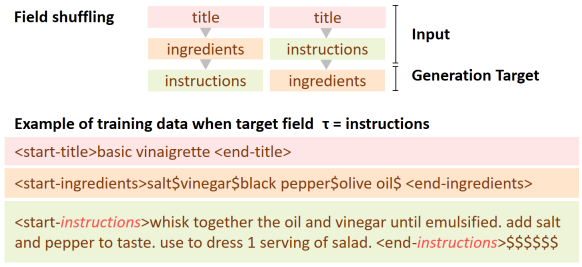

Figure 3: Illustration of Multi-field Generation

3.2 Multi-field Learning and Generation

Our aim is to train GPT-2 transformer provided by Radford et al [5] to generate multi-field recipe documents. Specifically, each recipe in Recipe1M consists of three fields: title, ingredients, and cooking instructions. To that end, we follow Zeller et al.'s approach [9] to perform multi-field learning and generation. Each field is encapsulated by field-specific start and end tokens. From the original training set, we further constructed a multi-field training set by shuffling the input and the target fields. We also shuffle the order of ingredients. Figure 3 illustrates how we shuffle the training data to enable the generation of target fields given recipe contexts as input fields.

To generate a target field τ, we append the field-specific start token <start-τ> to the input followed by sampling from the model until we hit <end-τ>. Note that we might have padding tokens '$' after <end-$\tau$>. During each sampling step, we select a token among the k highest-ranked tokens of the entire vocabulary sorted by the prediction probability. This sampling strategy is also known as top-k sampling. Although the generation process is non-deterministic, we can control the generation diversity by the hyperparameter k. The higher the k, the larger the diversity of generated texts.

We select the optimal learning rate based on perplexity using the validation set. Next, we follow the Byte-Pair Encoding used in GPT-2 to process the inputs. The maximum number of tokens is set to 512 since it fully covers 98.6% recipes in Recipes1M. We add padding tokens at the end of each recipe to unify the sampling length. For recipes exceeding maximum tokens, a randomly selected chunk of 512 tokens is sampled in each training iteration. Due to the memory limitation in our training environment (16GB), we set the batch size to 8.

[1]https://github.com/nytimes/ingredient-phrase-tagger

4 EXPERIMENTS

In this section, we describe the experimental setup to evaluate the performance of the fine-tuned recipe GPT-2 models on the 4,000 recipe data in the test set. Specifically, two recipe text generation tasks are focused: (1) generating ingredient texts from recipe title and cooking instructions; and (2) generating cooking instruction texts from title and ingredients. We try to find the best sampling hyperparameters and select the most suitable model for live deployment. Lastly, we further examine the degree of coherence of ingredients mentioned in different fields within the same recipes. A high-quality recipe should consistently refer to the same set of ingredients across all fields.

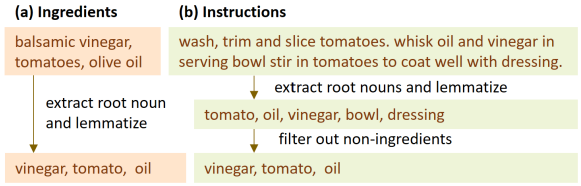

Figure 4: Extracting Ingredient-containing Root Nouns

4.1 Evaluation Metrics

Ingredient generation. We compute standard F1 scores between the set of generated ingredients and ground-truth ingredients. More specifically, we only consider lemmatized root nouns of ingredients as units of analysis instead of the whole noun phrases in the evaluation. we utilize spaCy[2] to identify the lemmatized root nouns as illustrated in Figure 4(a).

Instruction generation. We compute two standard n-gram overlap based metrics, BLEU[3] and ROUGE[4], as well as normalized tree edit distance (NTED) to measure the overall quality of the generated instruction texts with respect to the ground-truth instructions. For NTED, we employ similar procedures used in Chang et al. [2], i.e., we represent instructions as a tree structure where the verbs and nouns (extracted by spaCy) are respectively on stems and leaves representing the relation of actions (i.e. cooking methods) and subjects (i.e. ingredients and cooking tools). We then utilize Zhang-Shasha algorithm to score the edit distance which counts the numbers of INSERT, REMOVE, and REPLACE operations required to get the gold reference from the generated instruction. Lastly, we normalize the tree edit distance by the total number of nodes in both generated and reference trees.

Between-field ingredient coherence Typically, we expect that all ingredient words should be consistently referred to across all fields. To investigate how well the model captures this phenomenon, we measure the overlap between a set of ingredient words extracted from the generated instructions (I_g) and those in the input ingredient texts (I) using Jaccard similarity. First, we extract the ingredient-containing root nouns from the generated instructions and the input ingredient texts. We use two extraction methods for those root nouns as illustrated in Figure 4. The first method has been applied previously in the F1 calculation. The second method

[2]https://spacy.io
[3]https://github.com/moses-smt/mosesdecoder/
[4]https://pypi.org/project/rouge

extracts root nouns from instructions using an ingredient dictionary to filter out non-ingredients. The dictionary is derived from [8] with 89 additional ingredients added by a member of the research team to increase ingredients coverage with respect to Recipe1M dataset. In total, the vocabulary contains 1,992 root nouns. Then, we compute Jaccard similarity scores between the two sets of root nouns. To establish a baseline, we also perform the same procedures to compare the ingredient coherence between human-written instructions (I_h) and human-written ingredients (I).

4.2 Results

Ingredient and instruction generations. Overall, the results confirm the effectiveness of RecipeGPT in generating ingredient and cooking instruction texts. Table 1 displays the model performances on different sampling hyperparameters (k) for top-k sampling. The best model is subsequently selected for live deployment. In Table 1(a), we experiment with different k to examine its effects on the generation diversity and quality. As we can see, models with lower k consistently perform better across all evaluation metrics. Next, k is also positively associated with the length of generation as indicated by the average number of ingredients and the brevity penalty. Ultimately, we set k to 3 in the live deployment to achieve a good balance of quality and diversity.

We also evaluate different training approaches as presented in Table 1(b). As we can see, all models show similar performance levels, suggesting that the knowledge captured in the pre-trained weights are not that helpful when training with a large dataset. Next, fine-tuning on a more complex model (355M vs. 124M) produces a superior model according to perplexity, which is consistent with previous findings [9, 10]. As the generation results among those models are indistinguishable and it takes approximately 9 seconds for 124M and 12 seconds for 355M to process a test case, we deploy the 117M fine-tuned GPT-2 in the live version of RecipeGPT[5].

Ingredient coherence. The Jaccard similarity scores between (I_g, I) and (I_h, I) are 0.53 and 0.49, respectively. This suggests that RecipeGPT captures the between-field ingredient coherence as well as human authors, if not better. Surprisingly, the score for human-written instructions is slightly lower than that of the generated instructions. Upon further inspection, we found that human authors sometimes refer to the input ingredients as a whole when writing the instructions instead of mentioning each individual ingredients, e.g., "combine all ingredients" v.s. "combine vodka and orange juice", thus lowering the overlap.

5 DEMONSTRATION & CONCLUSION

Demonstration. During the demo session, we will demonstrate the whole recipe generation and evaluation process as shown in Figure 1. Both cooking instruction generation and ingredient generation tasks will be available for demonstration. For each task, users will provide specific recipe contexts (e.g., recipe title and ingredients/cooking instructions) to generate respective recipe texts. Single recipe evaluation features are also available to use. RecipeGPT is accessible online at https://recipegpt.org/ [6]

[5]Using a learning rate of 1e-4, the deployed model in our system takes 5 days, approximately 5 epochs, to converge in a single NVIDIA Tesla V100 GPU.

[6]We share the code used to run the experiments at https://github.com/LARC-CMU-SMU/RecipeGPT-exp.

Table 1: Model Performances

Generation output	Ingredients		Instructions				
	F1	# Ingr.	BLEU	BP	R-L	NTED	PPL
(a) Performances of fine-tuned GPT-2 (124M) on validation set							
Top-k sampling with k = 1	0.79	7.6	9.81	0.62	0.39	0.51	
k = 3	0.76	7.9	8.29	0.70	0.37	0.52	
k = 5	0.7	8.0	7.81	0.75	0.36	0.53	
k = 10	0.74	8.3	7.42	0.83	0.35	0.53	
k = 30	0.71	8.7	7.15	0.94	0.34	0.54	
(b) Performances of different RecipeGPT models on test set							
Trained from scratch (124M)	0.75	7.6	8.58	0.71	0.37	0.52	3.77
Fine-tuned GPT-2 (124M)	0.76	7.8	8.34	0.71	0.36	0.52	3.70
Fine-tuned GPT-2 (355M)	0.77	7.9	8.29	0.70	0.37	0.52	3.63

F1: F1 of two sets of lemmatized root nouns, # Ingr: average number of ingredients, BP: Brevity Penalty estimated in BLEU, R-L: ROUGE-L, NTED: Normalized Tree Edit Distance, PPL: BPE Perplexity

Conclusion. RecipeGPT is a novel generative pre-trained transformer based recipe generation and evaluation system. The evaluations illustrate its potential in automatic recipe generation. As the system is publicly accessible online, we hope that this will encourage others to try experimenting with different recipe contexts to further investigate its potentials and behaviors. Lastly, we also provide several user-interface features in RecipeGPT to assist users in examining the quality of the generation at the recipe level, and suggest potential ways to improve recipe generation models.

ACKNOWLEDGMENTS

This research is supported by the National Research Foundation, Prime Minister's Office, Singapore under its International Research Centres in Singapore Funding Initiative.

REFERENCES

[1] Antoine Bosselut, Omer Levy, Ari Holtzman, Corin Ennis, Dieter Fox, and Yejin Choi. 2017. Simulating action dynamics with neural process networks. *arXiv preprint arXiv:1711.05313* (2017).

[2] Minsuk Chang, Léonore V Guillain, Hyeungshik Jung, Vivian M Hare, Juho Kim, and Maneesh Agrawala. 2018. Recipescape: An interactive tool for analyzing cooking instructions at scale. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, 451.

[3] Bodhisattwa Prasad Majumder, Shuyang Li, Jianmo Ni, and Julian McAuley. 2019. Generating Personalized Recipes from Historical User Preferences. In *EMNLP*. 5975–5981. https://www.aclweb.org/anthology/D19-1613

[4] 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. *IEEE transactions on pattern analysis and machine intelligence* (2019).

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

[6] 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*. 10453–10462.

[7] Lav R Varshney, F Pinel, KR Varshney, D Bhattacharjya, A Schörgendorfer, and Y-M Chee. 2019. A big data approach to computational creativity: The curious case of Chef Watson. *IBM Journal of Research and Development* 63, 1 (2019), 7–1.

[8] Ingmar Weber and Palakorn Achananuparp. 2016. Insights from machine-learned diet success prediction. In *Biocomputing 2016: Proceedings of the Pacific Symposium*. World Scientific, 540–551.

[9] Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. 2019. Defending Against Neural Fake News. *arXiv preprint arXiv:1905.12616* (2019).

[10] Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, and Bill Dolan. 2019. DIALOGPT: Large-Scale Generative Pre-training for Conversational Response Generation. *arXiv preprint arXiv:1911.00536* (2019).