CAN THO UNIVERSITY

COLLEGE OF INFORMATION AND COMMUNICATION TECHNOLOGY

PROJECT - FUNDAMENTAL TOPICS
INFORMATION TECHNOLOGY
(HIGH-QUALITY PROGRAM)

COOKING RECIPE GENERATION MODEL FROM INGREDIENTS

Student: Phan Trung Thuan
Student ID: B2111957
Class: 2021-2025 (K47)
Advisor: Dr. Lam Nhut Khang

Can Tho, 05/2024

CAN THO UNIVERSITY
COLLEGE OF INFORMATION AND COMMUNICATION TECHNOLOGY
DEPARTMENT OF INFORMATION TECHNOLOGY

PROJECT - FUNDAMENTAL TOPICS
INFORMATION TECHNOLOGY
(HIGH-QUALITY PROGRAM)

COOKING RECIPE GENERATION MODEL FROM INGREDIENTS

Student: Phan Trung Thuan
Student ID: B2111957
Class: 2021-2025 (K47)
Advisor: Dr. Lam Nhut Khang

Can Tho, 05/2024

ACKNOWLEDGMENTS

ABSTRACT:

Cooking is a unique activity of human which have been developed for thousand years. It’s always a complicated problem, specially with the limit of ingredient at home. So, this study wants to create an ingredient2recipe model using transformer and RecipeNLP dataset [1] which will allow everyone can handle with cooking by detailed instruction generated from the model.

TABLE OF CONTENT:

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION:

With the develop of Large Langue Model (LLM) nowadays with many applications, this study wants to create a model that help housewife prepare meals easily for their family base on what they have in the fridge, named cooking recipe generation model using RecipeNLG dataset [1].

The goal of this study is developing a model can solve the problem which has acceptable performance and accuracy. There are a lot of research about this topic like Inverse cooking: Recipe generation from food images [2] which generate recipe from image using transformer model, RecipeGPT: Generative Pre-training Based Cooking Recipe Generation and Evaluation System [3] show the deep approach in recipe generation problem and specially Cooking Recipe Generation Based on Ingredients Using ViT5 [4] which is the main inspiration of this study.

Our general approach is research and learn from related work, state the theory, implement the model, evaluation and conclusion. This thesis includes 5 parts: Introduction, theory, methods, experiment and conclusion.

CHAPTER 2: THEORY

1. Pipeline

We used GPT2 model [5] for recipe generation task which have proved very good performance in the original paper. This model allow us generate hundreds of tokens which has a very well semantic content, in our problem is the cooking recipe.

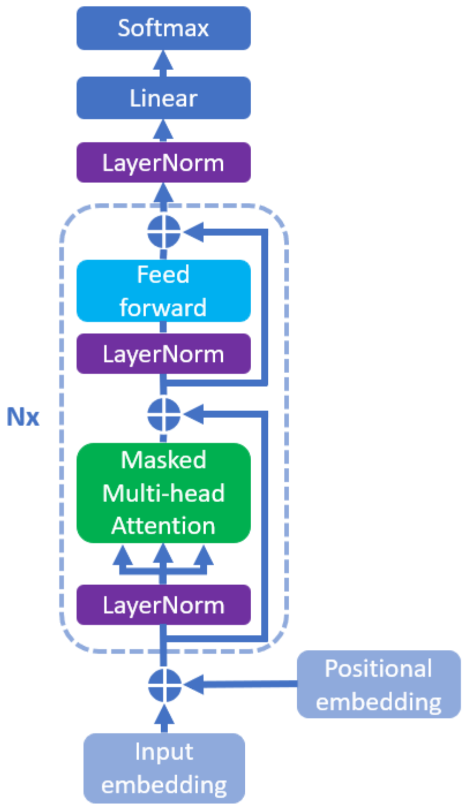

Figure 1: GPT2 Model Architecture

2. Input embedding and positional embedding

We used 2 embedding layer (both are learnable layers) which turn token ids and position values to vectors. By adding them together, we receive a rich information vector contain both token information and its position in the sentence which are necessary for the prediction of the next word in the sequence.

$$x = EM(id_{token}) + EM(pos_{token})$$

Where EM stands for Embedding layer, x is the input of the transformer blocks, id_{token} is token's id base on the vocabulary, pos_{token} is the position of token base in the sequence. This formular will apply for every token in the sequence and return a matric x size of (seq_len, d_{model}).

3. Layer normalization

Based on the original paper [5], we used Layer normalization [6] which are also has learnable layer. By using the formular below, we got a normalize vector which has stable distribution.

$$x_{norm} = \frac{x - E(x)}{\sqrt{Var(x) + \epsilon}} * \gamma + \beta$$

Where $E(x)$ and $Var(x)$ is the mean and variance calculated over the last dimension, ϵ is the very small epsilon value which is used in order to prevent divided by zero situation. Lastly, γ and β is 2 learnable parameter called scale and shift which is used to allow model to control the output of normalize layer, contribute on the stability of the backpropagation process.

4. Masked multi-head attention

Same as other transformer decoder block which has the core is masked multi-head attention. This part of the decoder is used for achieving the relationship between words.

$$(Q, K, V) = x_{norm} \cdot (W_Q, W_K, W_V)$$

First of all, we dot product x_{norm} received from Layernorm [6] with 3 weight matrices to achieve Q, K and V stand for query, key and value respectively. W_Q, W_K and W_V are weight matrices which have the same size is (d_{model}, d_{model}). Therefore, the Q, K and V have the shape of x_{norm} is (seq_len, d_{model}).

$$(Q_i, K_i, V_i) = hsplit(Q, K, V)$$

Secondly, we split Q, K and V matrices horizontally into num_head triplets in order to calculate attention score for each head where i is the value between 0 and (num_head-1) stand for i-th head. Now, the shape of Q_i, K_i and V_i diminish to $(seq\_len, \frac{d_{model}}{num\_head})$.

$$mask = \begin{bmatrix} 0 & -\infty & -\infty & \dots & -\infty & -\infty \\ 0 & 0 & -\infty & \dots & -\infty & -\infty \\ 0 & 0 & 0 & \dots & -\infty & -\infty \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 0 & -\infty \\ 0 & 0 & 0 & \dots & 0 & 0 \end{bmatrix}$$

$$att\_score_i = softmax(\frac{Q_i \cdot K_i^T}{\sqrt{d_k}} + mask) \cdot V_i$$

Moreover, we calculate attention score for each head with the above formular where Q_i, K_i^T and V_i are query matrix, transposed key matrix and value matrix of i-th head. The mask matrix shape is (seq_len, seq_len) was applied allow each word shouldn't attention to the following word in the sentence because those haven't generated. The att_score_i matric of i-th head has the same size of V_i which will help us stacking other block without concern about the input and output shape because it's equal.

$$x_{att} = hstack(att\_score_i) \cdot W$$

Finally, to get the output of the masked multi-head attention, we stack every attention score of each head horizontally and dot product with the final weight matrix W shape (d_{model}, d_{model}). As a result, we receive a matrix which has the same size of x_{norm} and have the information present the relationship between word and word in the sentence, this plays an indispensable role in our model.

5. Skip connection

The skip connection is a technique first introduce in ResNet [7] which not only help our model prevent or reduce the gradient vanishing problem in a very deep learning model like transformer but also allow the input vector can move a small range to fit the true position in the vector space which is the meaning of the word.

6. Feed forward

Feed forward layer is a multilayer perceptron which has an input layer, a hidden layer has size equal 4 times of input layer size and output layer has the same size of the input layer according to the original transformer paper and using GELU activation function after feeding forward the hidden layer.

$$GELU(x) = \frac{1}{2}x\left\{1 + tanh\left[\sqrt{\frac{2}{\pi}} * (x + 0.044715x^3)\right]\right\}$$

$$z_{hidden} = GELU(W_{layer} \cdot x_{norm} + b_{layer})$$

$$x_{ff} = W_{output} \cdot z_{hidden} + b_{output}$$

Where $W_{layer}, b_{layer}, W_{output}$ and b_{output} are learnable parameters. x_{ff} is the output of the feed forward block which has the same size of x_{norm}. It helps words' features transform the information of themselves without attention into other words which is the role of the masked multi-head attention.

CHAPTER 3: METHODS

1. Dataset

We used a mini version of RecipeNLG dataset which is 500000 samples because of the limitation of resource. we only use the ingredients and directions columns as the input of the model. Each sample of ingredient and direction are lists. Therefore, we convert them into a long string by merge each element of the list with a space character. Lastly, we combine 2 achieved long string together.

$$input = ','.join(ingredients) + '\backslash nInstructions:' + ''.join(directions)$$

Where join function same as join function in python which turn a list into a string separate by a specific character, in our case is a space. We place $'Instructions:'$ string between ingredients and directions as a separator allows model to know which is which and help us extract the generated cooking recipe easier.

2. Tokenizer

Because of the limit of words in the ingredients and directions, we create our own tokenizer using Byte Pair Encoding which is the similar to GPT2 model. We limit our vocabulary size is 10000 different tokens which is enough for our task.

3. Initialize model

We create the smaller version of GPT2 model which has embedding dimension is 512, 8 decoder and 8 heads for each Masked multi-head attention block. In order to generalize model, we use dropout layer [8] with 0.2 dropout rate after calculating the attention score and after the output layer of the feed forward block which can be beneficial for regularization and improving generalization in our model.

Moreover, we configurate sequence length is 400 tokens, vocabulary size is 10000 which is discussed above and initialize the weights of our model using Glorot initializer [9] which is the default initializer in pytorch and training it from scratch with the dataset above.

4. Training parameters, loss function and optimizer

We train our model in this dataset by 20 epochs and batch size is 40 because of the limit of resource we have. The base of text generation transformer is classification, so we use cross entropy as loss function. Furthermore, we use AdamW optimizer [10] with 10^{-3} learning rate and weight decay equal 0.1 because this theoretically is the best optimizer we know because it combines RMSprop, Momentum and Weight Decay technic.

$$L = \sum_i y_i * \log(\hat{y}_i)$$

Where y_i is the ground truth value we want model to predict and $\hat{y}_i$ is the predicted value as output of the model. The optimizer will calculate the gradient of each learnable parameters in order to update them with AdamW algorithms.

CHAPTER 4: EXPERIMENT

Because of the limited of resources, we just train the model with 500000 samples and 20 epochs. According to table 1, the result we achieved satisfy us despite BLEU score smaller a little bit because of too few training iteration and less parameters compare to GPT2, while the WER score seems extremely high means it's very different with the reference.

Model	BLEU (N = 4)	WER
Our GPT model	10.912856	3.829499
GPT2 (small)	**11.071026**	**0.996978**

Table 1: Evaluation result of our GPT model and GPT2 model without fine tuning on RecipeNLG dataset (500000 samples)

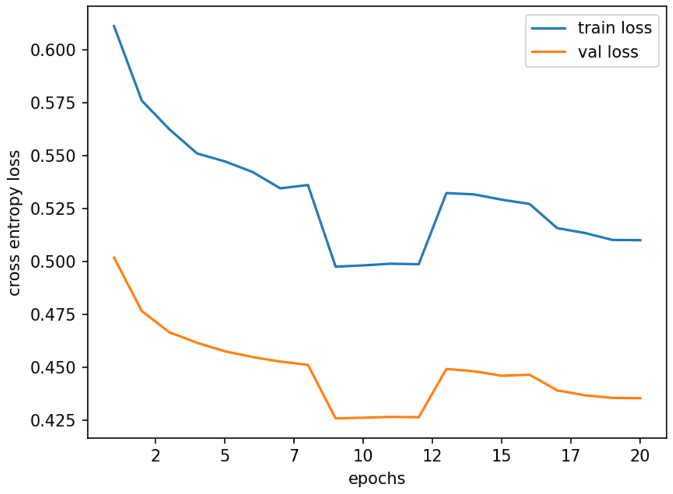

Figure 2: Cross entropy loss graph of our model while training in 20 epochs

Because GPT2 model maybe haven't train on this kind of dataset so its result doesn't better with our model too much. According to table 2, both models have some outcoming ingredients like onions, garlic, etc which is not what we want. We want to solve the model generate cooking recipe step by step and use all or less than the ingredients it gets in order to help everyone clear their fridge.

	Our GPT model	GPT2 (small)
Ingredients	["3 chicken breasts", "2 bananas", "2 tablespoons curry powder", "3 tablespoons butter", "1 teaspoon parsley", "2 teaspoons ground coriander", "4 teaspoons lime juice", "1 teaspoon dry mustard", "1 lime zest", "1 teaspoon salt", "1/2 teaspoon pepper", "3/4 cup water", "1 tablespoon fresh coriander"]	
Directions	One chicken per each breast half, 4 ounces each of skinned, halved breasts of the bananas, 4 ounce packages ain another piece of heavy-duty plastic wrap, pound slightly with mallet or rolling pin. Melt 1/4 cup butter, in electric skillet over medium-hot (not deep).	Preheat oven to 350°F. In a small saucepan over medium heat, melt coconut oil. Add chicken and saute until tender. Add bananas, curry powder and butter. Cook on low to medium heat for about 15 minutes. Add chicken and saute until soft. Add onions and fry for a few minutes until translucent. Add salt and pepper and cook for a minute or two more. Remove from

	Brown chicken breasts on both sides. Remove toothpicks and keep warm. To the drippings, add the garlic and onions and saute until onions are clear, but not brown. Add the parsley and lime juice and mustard and mix well. Add this mixture, stirring until smooth and thick. Add pineapple chunks and heat through. Pour sauce over chicken and serve hot with grill.	heat and add chicken and saute for another minute or two more. Remove from heat and stir well with a wooden spoon. Serve with rice and a few other garnishes. Recipe from The Cook's Illustrated If you have a small amount of curry powder, add a little more and it will add a nice little crunch to your dish. If you don't have much, you can add it to your soup or even your chicken. Recipe from The Cook's Illustrated
References	["HEAT oven to 450 degrees.", "MIX in blender - Bananas, Curry Powder, Coraindor, Mustard, Butter, Lime Juice, Lime Zest, Salt, Pepper, Parsley and 1/4 cup water.", "DICE chicken and place in a roasting pan.", "MAKE several cuts throughtout each piece of the chicken. NOTE: Can just put in whole breasts not diced.", "POUR sauce over chicken.", "NOTE: I made several more cuts into the chicken to make sure sauce gets inches.", "ROAST in over for about 20 - 25 minutes on lower part of the oven.", "REMOVE pan from oven and REMOVE chicken (make sure is cooked).", "PLACE pan over moderare heat on stovetop and WHISK in remaining 1/2 cup water.", "WHISK until sauce is smooth and heated through.", "NOTE: Can add more water if you like sauce thinner.", "GARNISH with fresh coriander.", "ENJOY!"]	

Table 2: A sample generated directions from both models with the same ingredients.

CHAPTER 5: CONCLUSION

This is a very simple experiment with a smaller version GPT2 model on a dataset like RecipeNLG which is very weak compared to the other research about this topic. In the future, we want to do more research about Re-attention mechanism [11], Rotary position embedding technic [12], Kolmogorov-Arnold network [13], Mamba architecture [14], etc. Moreover, I am longing to dive deep into Large Langue Models (LLMs), do my best to enhance the performance with a bigger dataset and better configuration.

REFERENCES

[1] Michał Bień, Michał Gilski, Martyna Maciejewska, Wojciech Taisner, Dawid Wisniewski, Agnieszka Lawrynowicz, RecipeNLG: a cooking recipes dataset for semi-structured text generation, 2020.

[2] Amaia Salvador, Michal Drozdzal, Xavier Giro-i-Nieto, Adriana Romero, Inverse Cooking: Recipe Generation From Food Images, 2019.

[3] Helena H. Lee, Ke Shu, Palakorn Achananuparp, Philips Kokoh Prasetyo, Yue Liu, Ee-Peng Lim, Lav R. Varshney, RecipeGPT: Generative Pre-training Based Cooking Recipe Generation and Evaluation System, 2020.

[4] Khang Nhut Lam, Y-Nhi Thi Pham. Jugal Kalita, Cooking Recipe Generation Based on Ingredients Using ViT5, 2023.

[5] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Language Models are Unsupervised Multitask Learners, 2019.

[6] Jimmy Lei Ba, Jamie Ryan Kiros, Geoffrey E. Hinton, Layer Normalization, 2016.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, Deep Residual Learning for Image Recognition, 2015.

[8] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov, Dropout: A Simple Way to Prevent Neural Networks from, 2014.

[9] Xavier Glorot, Yoshua Bengio, Understanding the difficulty of training deep feedforward neural networks, 2010.

[10] Hutter, Ilya Loshchilov & Frank, DECOUPLED WEIGHT DECAY REGULARIZATION, 2019.

[11] Daquan Zhou, Bingyi Kang, Xiaojie Jin, Linjie Yang, Xiaochen Lian, Zihang Jiang, Qibin Hou, Jiashi Feng, DeepViT: Towards Deeper Vision Transformer, 2021.

[12] Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, Yunfeng Liu, RoFormer: Enhanced Transformer with Rotary Position Embedding, 2021.

[13] Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljacic, Thomas Y. Hou, Max Tegmark, KAN: Kolmogorov–Arnold Networks, 2024.

[14] Albert Gu, Tri Dao, Mamba: Linear-Time Sequence Modeling with Selective State Spaces, 2023.