2023 IEEE International Conference on Systems, Man, and Cybernetics (SMC)
October 1-4, 2023, Oahu, Hawaii, USA

Cook-Gen: Robust Generative Modeling of Cooking Actions from Recipes

Revathy Venkataramanan[1], Kaushik Roy[1], Kanak Raj[1,2], Renjith Prasad[1,3], Yuxin Zi[1], Vignesh Narayanan[1], Amit Sheth[1]

***Abstract*— As people become more aware of their food choices, food computation models have become increasingly popular in assisting people in maintaining healthy eating habits. For example, food recommendation systems analyze recipe instructions to assess nutritional contents and provide recipe recommendations. The recent and remarkable successes of generative AI methods, such as auto-regressive Large Language Models, can enable robust methods for a more comprehensive understanding of recipes for healthy food recommendations beyond surface-level nutrition content assessments. In this study, we investigate the use of generative AI methods to extend current food computation models, primarily involving the analysis of nutrition and ingredients, to also incorporate cooking actions (e.g., add salt, fry the meat, boil the vegetables, etc.). Cooking actions are notoriously hard to model using statistical learning methods due to *irregular data patterns* - significantly varying natural language descriptions for the same action (e.g., marinate the meat vs. marinate the meat and leave overnight) and infrequently occurring patterns (e.g., add salt occurs far more frequently than marinating the meat). The prototypical approach to handling irregular data patterns is to increase the volume of data that the model ingests by orders of magnitude. Unfortunately, in the cooking domain, these problems are further compounded with larger data volumes presenting a unique challenge that is not easily handled by simply scaling up. In this work, we propose novel aggregation-based generative AI methods, Cook-Gen, that reliably generate cooking actions from recipes, despite difficulties with irregular data patterns, while also outperforming Large Language Models and other strong baselines.**

I. INTRODUCTION

Alice Waters once said, "We are what we eat", succinctly summarizing the impact that food can have on people's physical and mental well-being. Increased access to knowledge about the impact of food choices along with rapid advances in health monitoring technology, has led to individuals being more aware and cautious about their dietary choices [1]. There has been a growing trend to utilize AI-based diet management systems that analyze food recipes to provide an enhanced food recommendation for users. [2], [3]. Most of these systems incorporate discriminative models for the identification of nutritional information in recipes to encourage healthy eating habits (e.g., ingredients and their nutritional values). However, these models lack a more comprehensive understanding of the recipe. For example, they often do not account for cooking actions when providing recommendations [4]. It is vital to include cooking actions in the decision-making of diet management systems since they can significantly impact the nutritional value of food [5], ensure food safety [6], and even produce harmful substances. For instance, grilling meat at high temperatures produces cancer-causing compounds [7], [8]. Figure 1 illustrates a recommendation system that analyzes the effect of cooking actions and its impact on the final recommendation. Recent

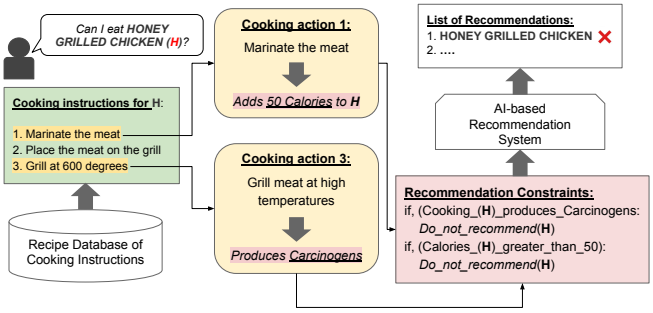

Fig. 1. The figure illustrates how understanding and modeling the cooking actions, not just the ingredients and their nutrition values, significantly impacts the final recommendation. Here meat inherently has nothing unhealthy about it. However, marinating it and grilling at high temperatures causes changes to its "healthiness" value.

successes in Large Language Models (LLMs) have the potential to enable a more comprehensive analysis of recipes, such as calorie count, macro-nutrient content, and ingredient lists. This information can then be used to identify healthy recipes and also suggest ingredient substitutions as per the user's health condition and food preferences [9]. LLMs can further be used to identify common cooking actions, ingredients, and flavor profiles associated with healthy dishes, which can be used to develop healthy eating recommendations and meal plans.

However, recipes written in natural language pose unique challenges with regard to natural language processing (NLP) that are not observed in traditional NLP tasks. For example, recipes include highly variable textual descriptions for the same cooking action (some people write " marinate the chicken and leave overnight" whereas others simply write "marinate the chicken"). Furthermore, recipes consist of several elements such as ingredients, utensils, temperature, and cooking actions (e.g., "marinate 1/2 cup of sliced chicken in a bowl") and, crucially, the frequency of cooking actions ("marinate") relative to ingredients, quantities, and other information ("1/2, "bowl and chicken") is low. We refer to the issues of low-frequency words and variation in lengths

*This work was not supported by any organization
[1]Artificial Intelligence Research Institute, University of South Carolina, Columbia, USA
[2]Birla Institute of Technology - Mesra, Ranchi, India
[3]Indian Institute of Technology, Jammu, India

2023 IEEE International Conference on Systems, Man, and Cybernetics (SMC) | 979-8-3503-3702-0/23/$31.00 ©2023 IEEE | DOI: 10.1109/SMC53992.2023.10394432

of cooking action descriptions collectively as *irregular data patterns*. Thus, discriminative models struggle due to the lack of distributional patterns with a high discriminatory value for expressing complex relationships between different variables (e.g., the effect of cooking actions).

With the rise of LLMs that have demonstrated remarkable successes with neural network-enabled generative models, AI systems can potentially model more complex phenomena, such as the impacts of cooking actions from the natural language text in recipes [10]. However, these models still make a discriminative model-type decision repeatedly (e.g., continuous next-word prediction), except that they do it using large neural networks accompanied by enormous volumes of training data. Therefore, the problems with *irregular data patterns* for discriminative modeling persist and are, in fact, further compounded at larger scales (e.g., imbalanced frequencies get more imbalanced). Inspired by recent innovations in dynamical systems control that utilize aggregation methods to handle *irregular data patterns* by (1) distilling more regular aggregated information from *irregular data patterns* and (2) leveraging high capacity function approximation capabilities of higher-order polynomial fits to model complex relationships, we propose `Cook-Gen`, a novel generative-AI method to generate cooking actions from recipe texts [11]. We compare `Cook-Gen` with state-of-the-art generative and discriminative baseline models and find that it consistently outperforms these baselines.

II. RELATED WORK

Here we review previous work on modeling cooking actions from recipes. Papadopoulos et al. represented cooking recipes using manually specified program abstractions that include cooking actions, quantities, ingredients, and other information from the recipes [12]. Recipescape, a cooking analysis tool, used Stanford's CoreNLP POS Tagger to extract verbs from recipes as a proxy for cooking actions [13]. Walter et al. also used a POS tagger in a similar context to reconstruct recipes using a workflow diagram of cooking actions. Shidochi et al. manually extracted cooking actions from 15 recipes to suggest substitute actions with similar effects [14]. Silva et al. used an off-the-shelf annotation tool to annotate recipes for cooking actions as part of an information retrieval system [15]. A common theme among the existing works is the manual effort involved in cooking action modeling, underscoring the challenge of automated modeling of cooking actions from recipes.

III. DESCRIPTION OF THE DATASET AND TASK, TRAINING AND EVALUATION METRICS

A. Dataset Construction and Description

We utilize the Recipe1M dataset, which is a comprehensive collection of 1 million high-quality recipes containing titles, ingredients, cooking instructions, and recipe images for each entry [16]. We then ask graduate students between ages 23 and 27 from the University of South Carolina to identify cooking actions in the recipe data for a randomly selected sample of 10000 recipe texts. Examples of recipe texts include, "bring the water to boil", "saute the vegetables in the wok", or "marinate the chicken". Their corresponding cooking actions are, "boil", "saute" and "marinate".

B. Task Description

As mentioned in Section I, we experiment with both discriminative and generative baselines. We describe the task in the discriminative and generative modeling settings.

Discriminative Modeling Setting: Let R be the set of all recipes. Each recipe $r \in R$ is written in natural language using a list of words W_r and also consists of a title T_r, a list of ingredients I_r, and a list of cooking actions A_r. Let $S = \{w_m \mid m \leftarrow \{1, \ldots, M\}\}$ denote the set of all M words across the N recipes. Thus we have a recipe set defined as:

$$R = \{(W_r, T_r, L_r, A_r) \mid W_r \subseteq S, T_r \subset S, I_r \subset S, A_r \subset S\} \tag{1}$$

We define a function $f : S \rightarrow \{0, 1\}$ that predicts whether a given word $w_m \in S$ is a cooking action or not. Thus we define the task of predicting cooking actions in a recipe as a binary classification task, where for each recipe r, we want to predict a sequence of binary labels $y_r = \{y_1, \ldots, y_M\}$, where $y_m = (f(w_m) \wedge \mathbb{I}(w_m \in W_r))$. Here $\mathbb{I}(w_m \in W_r)$ is an indicator function that evaluates to 1 if word w_m is present in the recipe r and evaluates to 0 otherwise.

Generative Modeling Setting: Consider the same definition of a set of recipes R as in Eq (1). Let $X_r = \{w_1, \ldots, w_L\}$ denote a partially complete natural language description of the recipe r consisting of L words, i.e., $X_r \subset W_r$ Formally, for a recipe $r \in R$, given a partial description X_r, the task is to learn a function $f : X_r \rightarrow G_r$ that can generate the set,

$$G_r = X_r \cup W_r \setminus X_r \cup A_r \tag{2}$$

C. Training and Evaluation Metrics

The training metrics we consider in our experiments are cross-entropy (CE) loss and weighted cross-entropy (WCE) loss. We use the WCE loss as there is a class imbalance issue, i.e., the number of cooking action words across all recipes compared to other words is relatively low. In the discriminative modeling case, the CE and WCE loss reduces to the binary CE and WCE losses, respectively. In the generative modeling case, only the CE loss (during word prediction of the words in the set defined in Eq (2)) is considered. Since there is a class imbalance in the dataset, we do not use accuracy and instead use precision and recall as our evaluation metrics.

Time and Space Efficiency Considerations: Despite the remarkable progress made by state-of-the-art AI models in recent years, they are not easily available to the general public. A key reason for this is their significant resource requirements, such as long training and inference times and high memory usage. For an AI system that can assist any individual in making healthy food choices, it is necessary for it to be resource-efficient, such as being able to run on a mobile phone and handle daily queries without overburdening the system. Consequently, we also evaluate the

number of parameters and memory usage of the models used for experimentation in both discriminative and generative modeling settings.

IV. METHODS AND EVALUATION

We divide our methods into two categories, (A) methods that work in the discriminative modeling setting and (B) methods that work in the generative modeling setting. For each category of methods, we describe the methods, detail the experimental setup, and discuss their evaluation metric scores.

A. Discriminative Modeling Methods and Evaluation

Data Preprocessing: Given the discriminative modeling task description in Section III-B, our aim is to learn the binary function $f : S \rightarrow \{0, 1\}$ that decides if a recipe word $w_m \in W_r$ is a cooking action or not. Since our dataset annotations consist of the full description of the cooking actions, e.g., for a recipe consisting of "marinate the meat and leave overnight", the annotation would be "marinate" (see Section III-A). We manually preprocess the annotations with binary ground truth labels based on if a word $w_m \in W_r$ is a cooking action or not. For example, the annotation "marinate" is preprocessed as a one-hot-encoded vector of size M for all the M words in S, with 1 in the places corresponding to the words "marinate" and 0 for every other word.

Method Descriptions: We try a rule-based method (R), where we use a simple lookup from a predefined list of cooking actions for each word to label the word as a cooking action or not (the predefined list can be found here). Next, we use the fine-tuned language models ELECTRA and XLNET [17], [18]. As mentioned in Section III-C, the discriminative modeling task suffers a class imbalance issue. The models XLNET and ELECTRA have specifically been trained to be robust to class imbalances in the data. Finally, we also try industry standards that have shown particular success in "binary tag prediction" due to the discriminative modeling task essentially boiling down to predicting the tag of a word as either a cooking action or not a cooking action. We experiment with Stanford's NLP tools and explosion AI's SpaCy [19], [20].

Experiments and Evaluation:

Model Configurations for the language models: For ELECTRA, we fine-tuned the electra-base-discriminator model obtained from hugging face. Early stopping was employed to prevent the model from overfitting. For XLNET, we fine-tuned the xlnet-large-cased model from hugging face. Similar to ELECTRA, early stopping was also implemented during training to prevent overfitting. For both ELECTRA and XLNET, the default pytorch implementations for CE and WCE loss were used. For the WCE loss, the weight assignment was 0.881 for label 1 and 0.119 for label 0.

Specifics of the SpaCy and Stanford's NLP model: We use Stanford's NER with its default configuration with the recognized named entities being labeled with 1 (other words 0), and SpaCy's part-of-speech (POS) tagger model 'en-core-web-lg' to label cooking actions (label 1 if POS is a "verb" for a word and 0 for other words).

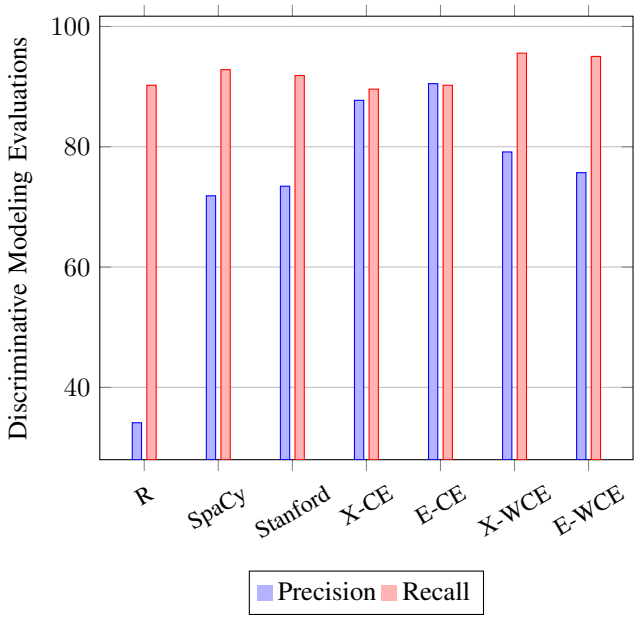

Fig. 2. Performance results of various discriminative modeling methods to predict cooking actions. X-CE, E-CE, X-WCE, E-WCE, R, SpaCy, Stanford represents XLNET with CE loss, ELECTRA with CE loss, XLNET with WCE loss, ELECTRA with WCE loss, The rule-based system (R), SpaCy's POS tagger, and Stanford's NER model respectively.

The evaluation scores for various discriminative modeling methods are presented in Figure 2. The rule-based system (R) exhibits the worst performance, which is unsurprising given the challenges of creating precise rules for determining cooking actions from recipes, particularly when the data is characterized by irregular patterns. Although SpaCy's POS tagger and Stanford's NER model, the two industry standards, perform somewhat better, they employ Convolutional Neural Network and Conditional Random Field-based approximation models that can only capture linear chain structured contextual dependencies between the recipe words. As a result, we anticipate that self-attention transformer-based techniques such as XLNET and ELECTRA will outperform these models since they can model more complex interactions or dependencies between the recipe words and are also designed to be robust to *irregular data patterns*. As expected, XLNET and ELECTRA demonstrate significant performance improvements. Nonetheless, employing WCE instead of CE in the training loss results in decreased precision, indicating that the model with CE loss may have overfit spurious data patterns. We conclude from these experiments that discriminative modeling may not be suitable for cooking action prediction since discriminatory patterns in the data are not present with regularity. Generative modeling has been demonstrated to be highly effective in situations where easily recognizable discriminatory patterns are absent, with the most notable example being large self-attention transformer-based models for language modeling. As a result, we will

now move on to an experimental analysis in the generative modeling context.

B. Generative Modeling Methods and Evaluation

Data Preprocessing: Given the generative modeling task description in Section III-B, our aim is to learn the function $f : X_r \rightarrow g_r$, that generates g_r as defined in Eq (2). Therefore we first tokenize recipe r into a set of words W_r and randomly sample the subset X_r from W_r. The set of cooking actions A_r is already part of the annotated dataset.

Method Descriptions: While self-attention transformer-based generative models have shown impressive success in various tasks, their high resource usage makes them challenging for ordinary users. Additionally, scaling up the model parameters and data does not necessarily solve the problem of *irregular data patterns*. To address this issue, aggregation-based methods can be utilized to enhance the robustness of the model to such irregularities. These methods handle data pattern irregularities by learning "aggregate patterns" that regularly appear in the data. To create small-scale generative models that can also handle *irregular data patterns*, we conducted experiments using two parametric models that have proven universal approximation capabilities: (1) a two-layer feedforward neural network with ReLU activations that aggregates the layer outputs across input words, and (2) a polynomial approximation-based aggregation method that raises word representations to higher-order polynomial powers before aggregating them. We also incorporated a position embedding layer to provide information about the sequence of words in the recipes. In general, the proposed generative method `Cook-Gen` employs aggregation operations in a generative modeling setup and allows toggling between different generative modeling architectures, e.g., neural network-based approximators vs. polynomial powers-based approximators. We will now formally describe the forward pass of the implemented generative models that can be used with `Cook-Gen` to generate the cooking actions for A_r for a recipe r from a partial natural language description (set of words) of the recipe X_r.

`Cook-Gen`-NN using an Aggregated Two-Layer Feedforward Neural Network Approximation: Using the same notation as in Section III-B, denoting X_r as the input to the forward pass, we can describe the sequence of forward pass steps as follows:

$$E_r = \mathbf{E}X_r,\ \mathbf{E} \in \mathbb{R}^{M \times d} \tag{3}$$

$$H_r = E_r + \mathbf{P_e}X_r,\ \mathbf{P_e} \in \mathbb{R}^{L \times d} \tag{4}$$

$$H_r = \max(\mathbf{W_z}^T H_r, 0),\ \mathbf{W_1} \in \mathbb{R}^{L \times d_1},\ \mathbf{W_z} \in \mathbb{R}^{d_1 \times d_2},\ z \in \{1, 2\} \tag{5}$$

$$H_r = \mathbb{E}(H_r[i, :]),\ i \in \{1, 2, \ldots d_1\} \tag{6}$$

$$O_r = \mathbf{W_o}^T H_r + \mathbf{B_o},\ \mathbf{W_o} \in \mathbb{R}^{d_2 \times M},\ \mathbf{B_o} \in \mathbb{R}^{d_2 \times M} \tag{7}$$

The final logits (before applying softmax) are obtained by extracting the last column of M values from O_r. Eq (6) in the forward pass denotes the aggregation operation, i.e., taking the average (expected value) of all d_1 rows of the second hidden layer output matrix. Here d represents the embedding dimension, $\mathbf{E}$ denotes the embedding matrix, $\mathbf{P_e}$ denotes the position embedding matrix, and the d_z corresponds to the dimensions of the hidden layer weight matrices.

`Cook-Gen`-PF using an Aggregated Polynomials-based Function Approximation: Neural networks are popular because the steps to obtain the hidden layer output matrix before the application of the aggregation operation in Eq (6) can be thought of as obtaining a feature map of the inputs X_r using the neural network. Such a feature map is *adaptive* because it is not predefined. Also, functions of feature maps obtained using neural networks, as described in Section IV-B (two layers with ReLU activations), have universal approximation capabilities. However, we also know that a set of polynomial powers of the input X_r is also a feature map, functions of which can also have universal approximation capabilities. But we first need to provide the function approximator knowledge of embeddings and positions of words in X_r. Thus we compute aggregates over polynomial powers of the rows in matrix H_r obtained just before Eq (5), i.e., H_r obtained after applying the embedding and position embedding matrix operations to X_r. The aggregation of polynomial powers of the rows of matrix H_r is described as follows:

$$H_r = \{\mathbb{E}(H_r[i, :]^j),\ i \in \{1, 2, \ldots L\} \mid j \in \{1, 2, \ldots, J\}\} \tag{8}$$

We then get a set of J vectors that are compiled into a new matrix H_r of dimension $\mathbb{R}^{J \times d}$. We also need to correspondingly modify the dimensions of weights and biases $\mathbf{W_o}$ and $\mathbf{B_o}$ of the output layer to be of dimension $\mathbb{R}^{d \times M}$.

`Cook-Gen`-GPT using ChatGPT: Given that ChatGPT is a generative model, we aim to incorporate it into the suggested "Cook-Gen" approach. Despite ChatGPT requiring significant computational resources, it can be made accessible to the general public through API calls, which eliminates the need to store the model on their personal devices. Additionally, by leveraging backend infrastructure, the API calls can reduce inference times and latency. Figure 3 illustrates how ChatGPT can be utilized to generate cooking instructions through API calls.

```
inst = 'Brush some melted butter in a 4 inch baking dish or oven safe bowl.'
prompt = 'Get cooking verbs from:' + ' ' + str(inst) + '.' + 'If no cooking
verbs, return only none. Return cooking verbs with no special characters.'

completion = openai.ChatCompletion.create(model="gpt-3.5-turbo", messages=[
{"role": "user", "content": prompt}])

result = completion.choices[0].message.content
```

Fig. 3. The figure shows how ChatGPT is prompted to obtain cooking actions from recipes using OpenAI API calls.

Experiments and Evaluation:

Model Configurations for `Cook-Gen`-NN, and `Cook-Gen`-PF: The `Cook-Gen`-NN model is trained using a configuration with $d = d_z = 200$ for z values of 1 and 2, $L = 512$, and $M = 33000$. This results in a

total of 13.38 million parameters. On the other hand, the Cook-Gen-PF model is trained with $d = 200$, $L = 512$, $J = 3$, and $M = 33000$, resulting in a total of 13.3 million parameters. Both models have trainable matrices consisting of $\mathbf{E}$, $\mathbf{P_e}$, $\mathbf{W_z}$ for $z \in 1, 2$, $\mathbf{W_o}$, and $\mathbf{B_o}$. Unlike the Cook-Gen-NN model, the number of parameters for the Cook-Gen-PF model does not depend on the polynomial order J. This means that the Cook-Gen-PF model has an advantage over the Cook-Gen-NN model in that it can handle more complex functions without increasing the parameter count.

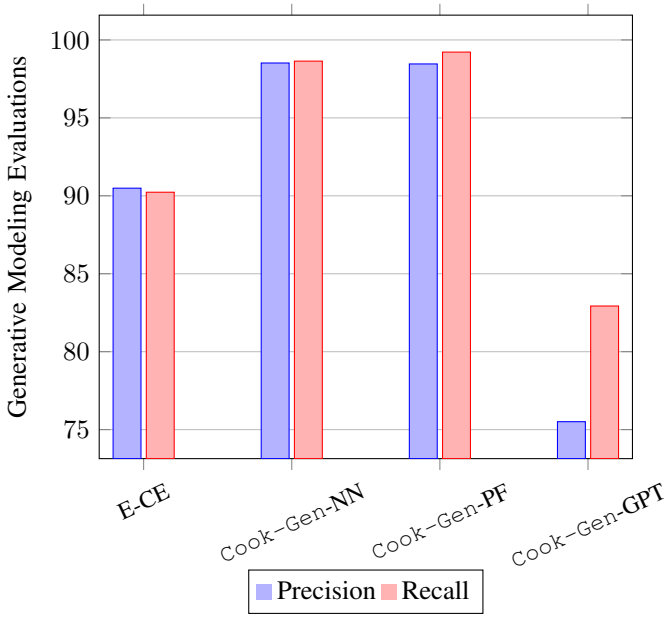

Fig. 4. Performance results of various generative modeling methods used to generate cooking actions as part of Cook-Gen compared to the best performing discriminative modeling method E-CE (ELECTRA with CE loss). Cook-Gen-NN, Cook-Gen-PF, Cook-Gen-GPT, represents Cook-Gen with a two-layer neural network approximation, with a polynomial powers approximation, and with access to ChatGPT API calls, respectively.

The evaluation scores for the different variants of Cook-Gen is shown in Figure 4. The figure also shows the best-performing discriminative modeling method for comparison. We notice that Cook-Gen-PF performs slightly better than Cook-Gen-NN. Since ChatGPT is very likely trained on abundantly available online recipe data, we expect it to perform reasonably well. However, ChatGPT performs rather poorly. This could be due to the fact that ChatGPT is not trained for specific tasks. It is worth noting that we do not have access to the model and are unable to fine-tune it on our dataset. Fine-tuning ChatGPT could have resulted in improved performance. Thus, we see that generative modeling using our proposed aggregation methods significantly outperforms strong baselines, demonstrating the effectiveness of Cook-Gen for modeling cooking actions despite issues and challenges caused by irregular data patterns.

C. Time and Space Efficiency Considerations

In Section III-C, we discussed the need for efficient modeling of cooking actions to allow easy access for ordinary users that do not have access to the resources required to run large state-of-the-art models such as ChatGPT.

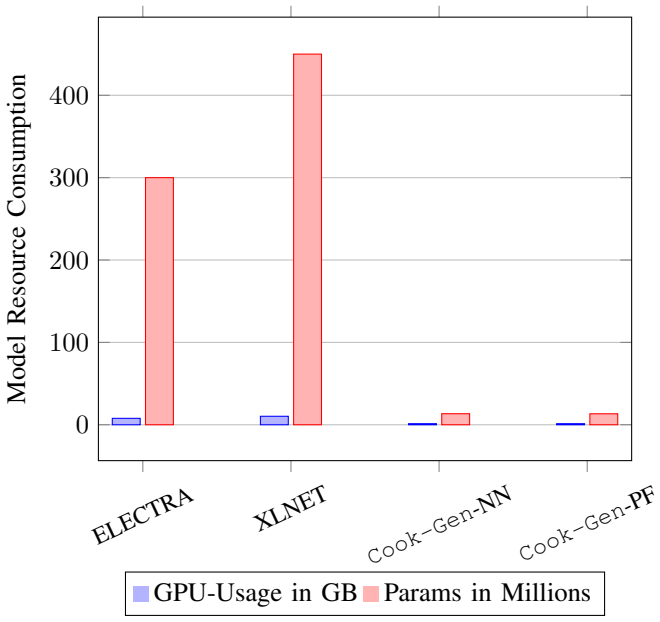

Fig. 5. The graph displays the usage of GPU memory (in GBs) and the number of parameters (in millions) for the ELECTRA and XLNET discriminative model methods in comparison to the Cook-Gen-NN and Cook-Gen-PF models. The resource consumption of the Cook-Gen models is considerably lower, indicating that Cook-Gen is highly effective for low-resource cooking action modeling.

Figure 5 shows the resource consumption in terms of GPU memory usage (in GBs) and the number of parameters in millions for the discriminative model methods ELECTRA and XLNET compared to the Cook-Gen-NN and Cook-Gen-PF models. The Cook-Gen models consume significantly fewer resources showing the effectiveness of Cook-Gen for low-resource cooking action modeling that is accessible to ordinary users on resource-constrained devices (e.g., mobile phones).

V. CONCLUSION

In this paper, we introduce Cook-Gen, which performs generative modeling of cooking actions that has the following salient features: (1) Cook-Gen is robust to *irregular data patterns* in recipes caused by variations in cooking action descriptions and imbalanced word frequencies; (2) Cook-Gen uses significantly lower memory and parameters for modeling while still outperforming much larger baselines. Our vision entails leveraging Cook-Gen as an integral component within a comprehensive automated AI system designed to support individuals in cultivating and sustaining healthy eating habits. This system may encompass various features, including a sophisticated recommendation system exemplified in Figure 1, where Cook-Gen proves to be remarkably beneficial.

Code and Data: All the code and data used to produce the results in the paper will be made available at this link.

ACKNOWLEDGEMENTS

This work was supported in part by the National Science Foundation (NSF) Awards 2133842 "EAGER: Advancing Neuro-symbolic AI with Deep Knowledge- infused Learning," and 2119654 "RII Track 2 FEC: Enabling Factory to Factory (F2F) Networking for Future Manufacturing" and was carried out under the advisement of Prof. Amit Sheth [21]–[23]. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] P. Stehr, V. Karnowski, and C. Rossmann, "The multi-faceted usage patterns of nutrition apps: a survey on the appropriation of nutrition apps among german-speaking users of myfitnesspal," *BMC medical informatics and decision making*, vol. 20, no. 1, pp. 1–13, 2020.

[2] M. R. Jospe, K. A. Fairbairn, P. Green, and T. L. Perry, "Diet app use by sports dietitians: a survey in five countries," *JMIR mHealth and uHealth*, vol. 3, no. 1, p. e3345, 2015.

[3] Q. Wang, B. Egelandsdal, G. V. Amdam, V. L. Almli, and M. Oostindjer, "Diet and physical activity apps: perceived effectiveness by app users," *JMIR mHealth and uHealth*, vol. 4, no. 2, p. e5114, 2016.

[4] R. Y. Toledo, A. A. Alzahrani, and L. Martinez, "A food recommender system considering nutritional information and user preferences," *IEEE Access*, vol. 7, pp. 96 695–96 711, 2019.

[5] N. I. of Health, "Retinoids: Basic science and clinical applications," U.S. Department of Agriculture, Agricultural Research Service, Tech. Rep., 2006. [Online]. Available: https://www.ars.usda.gov/ARSUserFiles/80400530/pdf/retn06.pdf

[6] B. J. McCabe-Sellers and S. E. Beattie, "Food safety: emerging trends in foodborne illness surveillance and prevention," *Journal of the American Dietetic Association*, vol. 104, no. 11, pp. 1708–1717, 2004.

[7] A. Farhadian, S. Jinap, H. Hanifah, and I. Zaidul, "Effects of meat preheating and wrapping on the levels of polycyclic aromatic hydrocarbons in charcoal-grilled meat," *Food chemistry*, vol. 124, no. 1, pp. 141–146, 2011.

[8] P. Das, D. Kemisetti, F. I. Jahan, S. E. Spriha, and S. C. Raka, "A comprehensive review on the formation of carcinogens from food products with respect to different cooking methods."

[9] S. S. Shirai, O. Seneviratne, M. E. Gordon, C.-H. Chen, and D. L. McGuinness, "Identifying ingredient substitutions using a knowledge graph of food," *Frontiers in Artificial Intelligence*, vol. 3, p. 621766, 2021.

[10] Ö. Aydın and E. Karaarslan, "Is chatgpt leading generative ai? what is beyond expectations?" *What is beyond expectations*, 2023.

[11] Y.-C. Yu, V. Narayanan, and J.-S. Li, "Moment-based reinforcement learning for ensemble control," *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[12] D. P. Papadopoulos, E. Mora, N. Chepurko, K. W. Huang, F. Ofli, and A. Torralba, "Learning program representations for food images and cooking recipes," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 16 559–16 569.

[13] M. Chang, L. V. Guillain, H. Jung, V. M. Hare, J. Kim, and M. Agrawala, "Recipescape: An interactive tool for analyzing cooking instructions at scale," in *Proceedings of the 2018 CHI conference on human factors in computing systems*, 2018, pp. 1–12.

[14] Y. Shidochi, T. Takahashi, I. Ide, and H. Murase, "Finding replaceable materials in cooking recipe texts considering characteristic cooking actions," in *Proceedings of the ACM multimedia 2009 workshop on Multimedia for cooking and eating activities*, 2009, pp. 9–14.

[15] N. Silva, D. Ribeiro, and L. Ferreira, "Information extraction from unstructured recipe data," in *Proceedings of the 2019 5th international conference on computer and technology applications*, 2019, pp. 165–168.

[16] A. Salvador, N. Hynes, Y. Aytar, J. Marin, F. Ofli, I. Weber, and A. Torralba, "Learning cross-modal embeddings for cooking recipes and food images," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 3020–3028.

[17] K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning, "Electra: Pre-training text encoders as discriminators rather than generators," *arXiv preprint arXiv:2003.10555*, 2020.

[18] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, "Xlnet: Generalized autoregressive pretraining for language understanding," *Advances in neural information processing systems*, vol. 32, 2019.

[19] J. D. Choi and M. Palmer, "Guidelines for the clear style constituent to dependency conversion," *Center for Computational Language and Education Research, University of Colorado Boulder, Institute of Cognitive Science, Technical Report*, vol. 1, p. 12, 2012.

[20] J. R. Finkel, T. Grenager, and C. D. Manning, "Incorporating non-local information into information extraction systems by gibbs sampling," in *Proceedings of the 43rd annual meeting of the association for computational linguistics (ACL'05)*, 2005, pp. 363–370.

[21] A. Sheth, K. Roy, and M. Gaur, "Neurosymbolic ai-why, what, and how," *arXiv preprint arXiv:2305.00813*, 2023.

[22] A. Sheth, M. Gaur, K. Roy, and K. Faldu, "Knowledge-intensive language understanding for explainable ai," *IEEE Internet Computing*, vol. 25, no. 5, pp. 19–24, 2021.

[23] A. Sheth, M. Gaur, K. Roy, R. Venkataraman, and V. Khandelwal, "Process knowledge-infused ai: Toward user-level explainability, interpretability, and safety," *IEEE Internet Computing*, vol. 26, no. 5, pp. 76–84, 2022.