IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

RecipeBowl: A Cooking Recommender for Ingredients and Recipes using Set Transformer

KEONWOO KIM[1]†, DONGHYEON PARK[1]†, MICHAEL SPRANGER[2], KANA MARUYAMA[2], and JAEWOO KANG[1],

[1]Department of Computer Science and Engineering, Korea University, Seoul, South Korea (email: akim,parkdh,kangj@korea.ac.kr)
[2]SONY AI, SONY Corporation, Tokyo 108-0075, Japan (e-mail: Michael.Spranger,kana.maruyama@sony.com)

Corresponding author: Jaewoo Kang (e-mail: kangj@korea.ac.kr)

† Equal contributors of this work.

ABSTRACT Countless possibilities of recipe combinations challenge us to determine which additional ingredient goes well with others. In this work, we propose RecipeBowl which is a cooking recommendation system that takes a set of ingredients and cooking tags as input and suggests possible ingredient and recipe choices. We formulate a recipe completion task to train RecipeBowl on our constructed dataset where the model predicts a target ingredient previously eliminated from the original recipe. The RecipeBowl consists of a set encoder and a 2-way decoder for prediction. For the set encoder, we utilize the Set Transformer that builds meaningful set representations. Overall, our model builds a set representation of an leave-one-out recipe and maps it to the ingredient and recipe embedding space. Experimental results demonstrate the effectiveness of our approach. Furthermore, analysis on model predictions and interpretations show interesting insights related to cooking knowledge.

INDEX TERMS Food Ingredient Combination, Food Ingredient Recommendation, Food Ingredient Relations, Recipe Context Learning, Recipe Recommendation, Set Representation Learning

I. INTRODUCTION

Finding the right additional ingredients and sample recipes is an essential, yet challenging task in the culinary world due to vast cooking possibilities [1]. Previous works have attempted to build food recommendation systems [2], [3] using small recipe datasets and shallow data-driven approaches. Food pairing tasks [4]–[6] have been proposed, but were limited to one-to-one ingredient recommendation. With multiple ingredients available, a system that is able to provide reasonable ingredient and candidate recipe choices based on sophisticated cooking knowledge may be desirable.

In this work, we propose RecipeBowl, a set-based model that jointly recommends ingredients and recipes. For example in Figure 1, given lime, chicken breasts, olive oil and garlic as input set, the user desires to cook an 'easy', 'main dish' grilled in an 'oven' using 'chicken'. In this case, the RecipeBowl suggests ingredients (e.g., balsamic vinegar, cilantro, white wine, rosemary and so on) that are likely to go well with the input set and satisfy the user's needs. Moreover, candidate recipes (e.g. *Easy Garlic Chicken*, *Grilled Pesto Chicken* and so on) are also provided to guide the user's decisions on cooking.

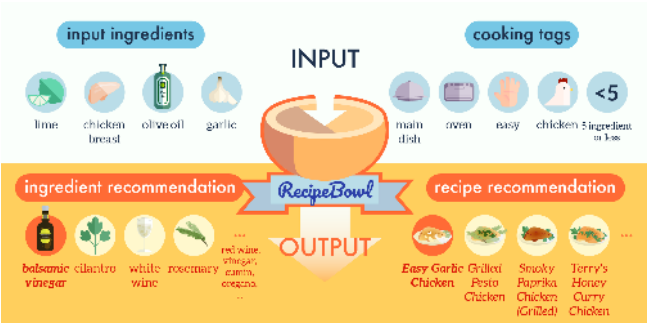

FIGURE 1: Overview of RecipeBowl Cooking Recommender. RecipeBowl takes two types of input then recommends additional ingredients and sample recipes. The bold-faced ingredient (balsamic vinegar) and recipe (*Easy Garlic Chicken*) are the targets selected from their original recipe.

We formulated a recipe completion task where the model is

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3120265, IEEE Access

given a *leave-one-out* set of ingredients and tag information to predict one target ingredient previously excluded from the *original* set. We constructed a dataset based on a large recipe corpus Recipe1M [7]–[9] where each instance consists of an leave-one-out set as input and target ingredient as output. We then trained the model in a supervised learning setting where it has to predict the target ingredient and its corresponding recipe given the *leave-one-out* set. The main objective is to simultaneously learn two different embedding spaces and push its vector projections towards the actual vector representations in each space. The trained model provides recommendations based on similarity-based rankings calculated between its predicted ingredient/recipe with the actual ones in each of their embedding spaces.

We performed quantitative and qualitative analysis on our model's recommendations to demonstrate the viability of our approach. Experimental results show that our model suggests reasonable ingredients that are relevant to recipe context. Observations on the predicted embedding space in t-sne visualizations, set context vectors in clustermaps and attention weights in heatmaps provide insight of how RecipeBowl utilizes recipe contextual knowledge and derives it from various ingredient combinations.

The major contributions are summarized as follows.

- We formulate a recipe completion task that trains a model on set-to-one prediction in a supervised learning setting.
- We propose RecipeBowl, a two-way cooking recommender model that adopts the Set Transformer [10] framework for building representations of ingredient sets [1].
- We introduce a large-scale recipe completion dataset [8], [9] using Tf-Idf scores for selecting optimal target ingredients.
- Both quantitative and qualitative analysis show that RecipeBowl suggests practical choices based on recipe context and ingredient relations.

II. RELATED WORK

A. LEARNING RECIPE REPRESENTATIONS

Cross-modal features, namely text and image features have been widely used for generating recipe representations [7], [11], [12]. These methods require image data to conduct recipe-related tasks. Recently, Li et al. has introduced Reciptor, a Set Transformer-based model [10] for learning recipe representations in a unsupervised fashion [9]. The authors pre-trained recipe representations from Recipe1M [8] using two loss functions which are the cosine similarity loss and the triplet loss. The authors of this work demonstrated the Set Transformer's effectiveness by using the pre-trained embeddings for food-related downstream tasks such as cuisine classification.

B. RECOMMENDATION IN FOOD DOMAIN

[1]The code for RecipeBowl is available in https://t.ly/rV8t

1) Recommending Ingredients and Food Pairings

Previous works related to food pairing discovery have been introduced where ingredient-ingredient relations are represented as edges in a network and its nodes denote the ingredients. Ahn et al. firstly proposed to define food pairings based on the number of flavor compounds shared between two ingredients [4]. Park et al. introduced Kitchenette, a Siamese Neural Networks based model trained on a large-scale dataset Recipe1M [8] to predict food pairing scores and discover novel ingredient pairings [6]. Haussman et al. incorporated semantic-driven knowledge graphs for food recommendation [13]. While the previously mentioned authors either utilized chemical information in ingredients or a large recipe corpus in food pairing related tasks, Park et al. further proposed to incorporate both aspects to construct a large scale ingredient-compound network called FlavorGraph using metapaths [14].

Prior works on recommending ingredients have also been proposed. Shino et al. used ingredient categories and co-occurrence relations to suggest suitable alternative ingredient for a given recipe [15]. Liu et al. extended this approach by considering the diversity of ingredient categories and novelty of ingredient combinations [16]. De Clercq et al. used non-negative matrix factorization and number of shared flavor compounds information to retrieve eliminated ingredients from recipes [3].

2) Recommending Recipes

Previous works have focused on personalized recommendation of recipes using various features and employing machine learning-based approaches [17]–[20]. While Ge et al. proposed to incorporate users' tags and ratings that indicate food preferences in recommendation [17], we employed a similar approach by utilizing recipe tag information such as *main dish*, *5-minute-cooking*. Other works have additionally taken nutrition-related factors into account to provide healthy food recommendations [21]–[24].

Perhaps one of the previous works that is closest to our task formulation is Cueto et al. [25]. The authors of this work employed memory-based collaborative filtering approaches to recommend ingredients for a given partial recipe. However, the dataset used in their work is small compared to our work as we trained our deep learning-based model on Recipe1M [8]. Moreover, while Cueto et al.'s model suggests only additional ingredients, our model is trained both on ingredient and recipe representations and provides each of their recommendations.

III. DATASET

A. PREPROCESSING ORIGINAL DATASET

We built an extended version of the Reciptor [9] dataset containing 507,834 recipes which is a subset of Recipe1M [7], [8]. Each recipe instance in our preprocessed dataset contains a list of ingredients, cooking instructions and cooking tags (630 unique tags) that were previously extracted from Recipe1M. Since the rich tag information (e.g., easy, healthy, seasonal [preference], main-dish, desserts, fruit [cuisine category],

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3120265, IEEE Access

Recipe	Information	Input Ingredients							Target Ingredients
Easy Garlic Chicken Breasts	Ingredients	chicken breasts	olive oil	garlic					**lime juice**
	Score	0.29	0.17	0.24					**0.30**
Garden Ranch Pizza	Ingredients	red bell peppers	ranch dressing	mozzarella cheese	parmesan cheese	broccoli	garlic clove		**pizza crust**
	Score	0.13	0.18	0.12	0.09	0.15	0.10		**0.22**
Creamy French Dressing	Ingredients	salt	garlic	sugar	dijon mustard	tomato paste	pepper	olive oil	**red wine vinegar**
	Score	0.06	0.14	0.09	0.16	0.16	0.11	0.10	**0.18**

TABLE 1: Examples of recipe completion data for RecipeBowl. Lime juice (0.30) in *Easy Garlic Chicken Breasts*, pizza crust (0.22) in *Garden Ranch Pizza*, and red wine vinegar (0.18) in *Creamy French Dressing* are selected as target ingredients.

meat, vegetarian, low-calorie [diet information], american, european, asian [regional category]) from Reciptor would be helpful in our task [17], we crafted a 630-dimensional tag information binary vector for each recipe instance. We prepared 3,729 unique ingredients and a 80%/10%/10% randomly partitioned dataset. Prior to dataset construction, we excluded recipes with few (4 or less) ingredients from each of the partitioned dataset. Therefore the dataset has 373,760 training recipes, 47,104 validation recipes and 47,104 test recipes.

B. SELECTING TARGET INGREDIENTS

We adopted De Clercq et al.'s recipe completion-based approach for training RecipeBowl [3]. The model is trained to predict a target ingredient x given a *leave-one-out* set X where x was previously eliminated from a *original* set $X \cup \{x\}$ of ingredients. Based on the above learning objective, we constructed a dataset for recipe completion where each instance includes an *leave-one-out* ingredient set, target ingredient and cooking tag information. Our main emphasis is to help the model learn cooking context based on the combinatory nature of various ingredients. In De Clercq et al.'s work, the target ingredients were selected randomly [3]. Among the randomly selected ingredients, commonly occurring ones such as salt and butter may act as trivial targets. These ingredients may render the model unable to differentiate the characteristics of ingredient combinations.

To prevent this, we selected target ingredients based on their Tf-Idf (Term Frequent-Inverse Document Frequency) score where terms and documents are ingredients and recipes respectively [26]. The Tf-Idf score indicates the relative importance of an ingredient within the recipe based on its occurrence in the whole corpus. We first calculated the Tf-Idf scores based on all ingredients, and then normalized them within each recipe where term frequency for each ingredient is always 1 in each recipe. We selected an ingredient x with the highest Tf-Idf score and eliminated it from each recipe.

Conclusively, the inputs for training RecipeBowl on recipe completion is the *leave-one-out* set X while the target is x for each recipe instance $X \cup \{x\}$. Table 1. shows the examples

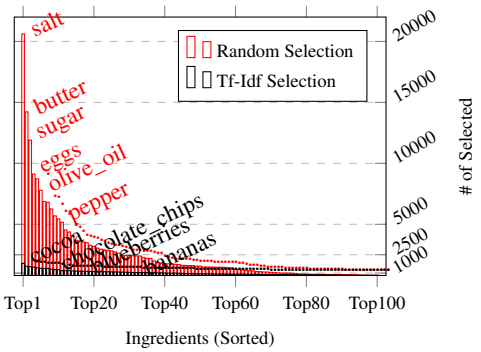

FIGURE 2: Analysis on Target Ingredient Selection

of our recipe completion dataset. In Table 1., the normalized Tf-Idf scores in *Creamy French Dressing* are low (e.g. salt (0.06), sugar (0.09), olive oil (0.10)). On the other hand, lime juice (0.30) in *Easy Garlic Chicken Breasts*, pizza crust (0.22) in *Garden Ranch Pizza* and red wine vinegar (0.18) in *Creamy French Dressing* have the highest normalized Tf-Idf scores.

We further justify our target selection approach by the following analysis. Figure 2. shows two distributions of target ingredients based on different selection options (Random and Tf-Idf). The distribution based on random selection is skewed where the highly frequent target ingredients based on random selection are commonly used ingredients (e.g. salt, butter and sugar) in most recipes. On the other hand, the distribution based on Td-Idf selection is relatively uniform which provides a better learning setting for RecipeBowl.

Along with recommending ingredients, RecipeBowl aims to simultaneously suggest recipe candidates. We utilized the pretrained recipe embeddings from Reciptor [9] as ground truths for training the recipe inference task of our model. Since the pretrained embedding vectors include sequential recipe context, we expect RecipeBowl to suggest acceptable recipe candidates and benefit ingredient recommendation.

IV. MODEL

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3120265, IEEE Access

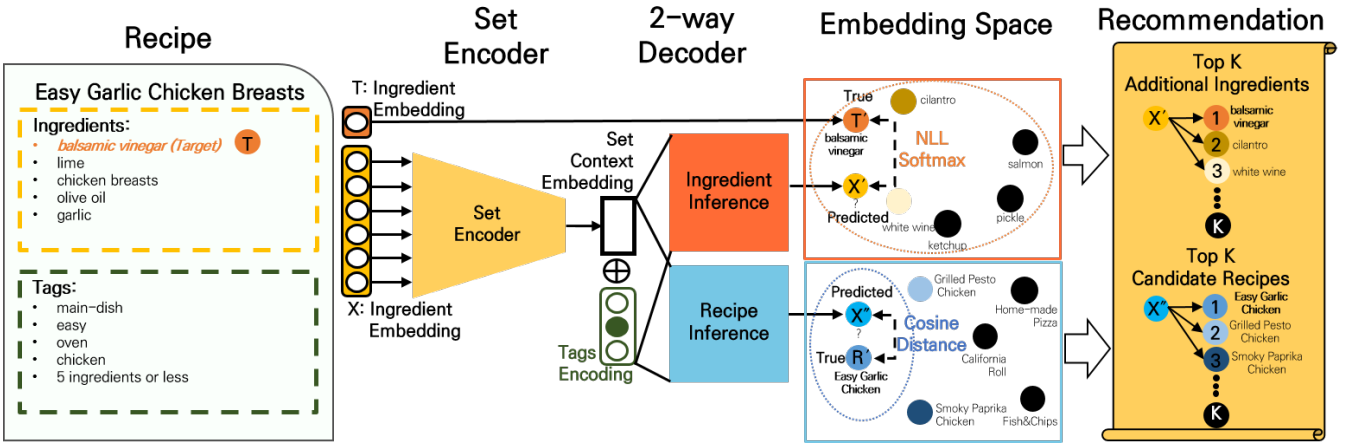

FIGURE 3: Overall Model Description for RecipeBowl: An example recipe *Easy Garlic Chicken Breasts* is shown on the left side. Top K additional ingredient and candidate recipe recommendations are shown on the right side.

A. OVERVIEW

RecipeBowl takes a set of ingredients as input and predicts a corresponding target ingredient and recipe as output (Figure 3.). The ingredients including the target are represented as continuous vectors retrieved from an embedding lookup table initialized by the ingredient node embeddings from FlavorGraph [14]. The target recipe vectors are pretrained embeddings retrieved from Reciptor [9]. The RecipeBowl consists of the Set Encoder and the 2-way Decoder. The Set Encoder encodes a set of ingredient vectors into a set context embedding space. The 2-way Decoder maps the set context vector into two different embedding spaces of different modality. The model is trained to approximate the predicted vector to its target ingredient and recipe vector in its corresponding embedding space and is trained in a multi-task learning fashion.

B. SET ENCODER - LEARNING SET REPRESENTATIONS

We adapt the Set Transformer framework in our model as the Set Encoder module to build latent representations for incomplete sets of ingredients using attention mechanism [10]. In this work, we constructed the Set Transformer as a stack of components including Induced Set Attention Blocks (ISAB) and a Multihead Attention based pooling (PMA) layer. The ISAB is fed with a input set of vectors to calculate self-attention weights between the elements where the final output is also a set of equal size. The PMA layer aggregates the element-wise features by calculating their attention weights on a set of parameterized seed vectors. Both ISAB and PMA layer use Multihead Attention Blocks (MAB) which are the components of the Transformer model originally proposed by Vaswani et al. [27]. The MAB computes the attention function with multiple projections of the input queries and key-value pairs. Different from the Set Transformer in Li et al.'s Reciptor [9], we constructed our version of Set Transformer with one ISAB followed by one PMA layer.

1) Multihead Attention

Given a set of n d_q-dimensional query vectors $\boldsymbol{Q} \in \mathbb{R}^{n \times d}$ and its corresponding key-value pairs $\boldsymbol{K} \in \mathbb{R}^{n \times d_k}$, $\boldsymbol{V} \in \mathbb{R}^{n \times d_v}$, an attention function takes $\mathbb{Q}$ as input and produces outputs using $\mathbb{K}$, $\mathbb{V}$. In our model, $d = d_q = d_k = d_v$ for simplicity.

$$\text{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \phi(\boldsymbol{Q}\boldsymbol{K}^\top)\boldsymbol{V} \tag{1}$$

where ϕ is scaled softmax $\phi(\cdot) = \text{softmax}(\cdot/\sqrt{d})$. The outputs of the above function are expressed as a weighted sum of $\boldsymbol{V}$ where each value's weight is determined by a dot product scalar of its corresponding key and the query.

An extended version of this mechanism called Multihead Attention was introduced by Vaswani et al. where multiple projections are applied to the query and key-value vectors to produce different attention-based outputs [27]. The k-head attention function has k triplets of linear transformations $W_i^{\boldsymbol{Q}}, W_i^{\boldsymbol{K}}, W_i^{\boldsymbol{V}} (i \in \{1, 2, ..., k\})$ each applied to $\boldsymbol{Q}$, $\boldsymbol{K}$ and $\boldsymbol{V}$ respectively. The k projections are each then fed into the attention function to produced k different outputs which are concatenated k-wise and finally projected into a h-dimensional space. The Multihead Attention is mathematically expressed as follows,

$$\text{Multihead}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = (O_1 \oplus ... \oplus O_i \oplus ... \oplus O_k)W^O \tag{2}$$

$$O_i = \text{Attention}(\boldsymbol{Q}W_i^{\boldsymbol{Q}}, \boldsymbol{K}W_i^{\boldsymbol{K}}, \boldsymbol{V}W_i^{\boldsymbol{V}}) \tag{3}$$

where $W_i^{\boldsymbol{Q}}, W_i^{\boldsymbol{K}}, W_i^{\boldsymbol{V}} \in \mathbb{R}^{d \times h}, W_i^O \in \mathbb{R}^{h \cdot k \times d}$.

2) Multihead Attention Block

While the query, key and value vectors involved in Multihead Attention may be different, the key and value vectors in the Multihead Attention Block are the same. Given two sets of vectors $\boldsymbol{X}, \boldsymbol{Y} \in \mathbb{R}^{n \times d}$, the MAB is mathematically expressed as follows,

$$\text{MAB}(\boldsymbol{X}, \boldsymbol{Y}) = \text{LayerNorm}(H + \text{RFF}(H))) \tag{4}$$

$$H = \text{LayerNorm}(\boldsymbol{X} + \text{Multihead}(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Y})) \tag{5}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3120265, IEEE Access

where RFF is a row-wise feedforward layer and LayerNorm is layer normalization ([28]).

3) Set Attention Block

The Set Attention Block was proposed by Lee et al. as an extension of the Multihead Attention Block to calculate self-attention weights between the vectors in a set [10]. The output from the SAB contains element-to-element interactions of the set. Higher order relations between the elements can be modeled through a stack of SABs. Our approach focuses on learning the combinatory nature of ingredients which provides a rationale for using SABs in the model architecture. Given a set of vectors $\boldsymbol{X} \in \mathbb{R}^{n \times d}$, the SAB is expressed as follows,

$$\text{SAB} = \text{MAB}(\boldsymbol{X}, \boldsymbol{X}) \tag{6}$$

4) Induced Set Attention Block

Another extension variant of the Multihead Attention Block proposed by Lee et al. is the Induced Set Attention Block. The ISAB contains trainable inducing vectors that are fed with the element vectors into the MAB to compute the outputs which are again fed into another MAB with the same element vectors [10]. Given a set of input vectors $\boldsymbol{X} \in \mathbb{R}^{n \times d}$ and a set of inducing vectors $\boldsymbol{K} \in \mathbb{R}^{k \times d}$, the ISAB is expressed as follows,

$$\text{ISAB} = \text{MAB}(\boldsymbol{X}, \boldsymbol{H}) \tag{7}$$

where $\boldsymbol{H} = \text{MAB}(\boldsymbol{K}, \boldsymbol{X})$.

5) Multihead Attention based Pooling Layer

One of the common permutation-invariant methods to aggregate the element-wise representations is element-wise summation [29], [30]. However, Lee et al. proposed aggregating the representations by applying multihead attention on another set of m parameterized seed vectors $\boldsymbol{S} \in \mathbb{R}^{m \times d}$ [10]. Given a set of n ingredient vectors refined by the previous SAB or ISAB, $\boldsymbol{Z} \in \mathbb{R}^{n \times d}$, pooling by Multihead Attention (PMA) is expressed as follows,

$$\text{PMA} = \text{MAB}(\boldsymbol{Z}, \text{RFF}(\boldsymbol{S})) \tag{8}$$

6) Set Transformer

Conclusively, given an input set of ingredient vectors $\boldsymbol{I} \in \mathbb{R}^{n \times d}$ the Set Transformer we employed in our work is mathematically expressed as follows,

$$\boldsymbol{S} = \text{LayerNorm}(\text{ReLU}(\text{PMA}(I')W_s + b_s))I' = \text{ISAB}(\boldsymbol{I}) \tag{9}$$

where $W_s \in \mathbb{R}^{d \times h}, b_s \in \mathbb{R}^h$ are the weights and biases for the final nonlinear transformation in the Set Transformer and $\boldsymbol{S} \in \mathbb{R}^h$ is the final latent representation for the set of ingredients. We denote this as the Set Encoder in our whole model architecture as it encodes a set of ingredients into a latent embedding space.

C. 2-WAY DECODER - PREDICTING INGREDIENTS AND RECIPES

The 2-way Decoder takes the set context vector concatenated with a 630-dimensional tag vector as input to generate the d-dimensional target ingredient vector and r-dimensional target recipe vector. The tag vectors are constraints to guide the model's predictive space. Given the encoded set representation $\mathbf{S} \in \mathbb{R}^d$ and the tag binary vector $\mathbf{T} \in \{0,1\}^{630}$, the predicted vectors for both the target ingredient $\hat{y_p} \in \mathbb{R}^d$ and recipe $\hat{y_q} \in \mathbb{R}^r$ are mathematically expressed as follows,

$$\hat{y_p} = \text{LayerNorm}(\text{ReLU}((\mathbf{S} \oplus \mathbf{T})\mathrm{W}_1 + b_1))\mathrm{W}_2 + \mathrm{b}_2 \tag{10}$$

$$\hat{y_q} = \text{LayerNorm}(\text{ReLU}((\mathbf{S} \oplus \mathbf{T})\mathrm{W}_3 + \mathrm{b}_3))\mathrm{W}_4 + \mathrm{b}_4 \tag{11}$$

where $\mathrm{W}_1, \mathrm{W}_3 \in \mathbb{R}^{h \times d}, \mathrm{W}_2, \mathrm{W}_4 \in \mathbb{R}^{d \times d}$ are trainable weights and $\mathrm{b}_1, \mathrm{b}_2, \mathrm{b}_3, \mathrm{b}_4 \in \mathbb{R}^d$ are trainable biases.

D. LOSS OBJECTIVE FUNCTION AND OPTIMIZATION

Given a pair of predicted and its ground truth target vectors $(\hat{y_p}, y_p)$, we employed a negative likelihood loss function based on a softmax over negative Euclidean distances in the ingredient embedding space [31], [32]. As we trained our model using batch sampling, the softmax for the Euclidean distance between the ith pair $(\hat{y_p}(i), y_p(i))$ is calculated over the batch of target ingredient vectors including $y_p(i)$. Given a batch B and model parameters Θ, the loss objective for RecipeBowl is mathematically expressed as follows,

$$\mathrm{f}(x, y) = -\sqrt{|x - y|^2} \tag{12}$$

$$\mathrm{L}_p\left(\hat{y_p}(i), y_p(i), \Theta\right) = -\log \frac{\mathrm{e}^{\frac{\mathrm{f}(\hat{y_p}(i), y_p(i))}{\tau}}}{\sum_{k=0}^{B-1} \mathrm{e}^{\frac{\mathrm{f}(\hat{y_p}(i), y_p(k))}{\tau}}} \tag{13}$$

where τ is a temperature scalar for controlling model optimization [33]. The model is therefore is trained on a distance metric learning setting since the Euclidean distance between the predicted ingredient and target ingredient is minimized [31]. Given the ith target ingredient as thet positive sample, we adopted the idea of using all other B target ingredients in a batch as negative samples for better optimization [34]. We will denote this scheme as using in-batch negatives.

For training the model on recipe prediction given the ith pair $(\hat{y_q}, y_q)$ in the training batch, we employed the cosine embedding loss defined as below,

$$\text{cosine}(x, y) = \frac{x \cdot y}{\|x\| \, \|y\|} \tag{14}$$

$$\mathrm{L}_q\left(\hat{y_q}(i), y_q(i), \Theta\right) = 1 - \text{cosine}(\hat{y_q(i)}, y_q(i)) \tag{15}$$

Finally, the multi-objective loss function for a batch of quadruplets $(\widehat{y_p}, y_p, \widehat{y_q}, y_q)$ is as below,

$$\mathrm{L}\left(\widehat{y_p}, y_p, \widehat{y_q}, y_q, \Theta\right) = \frac{1}{B}\sum_{i=0}^{B-1} L_p(i) + \frac{1}{B}\sum_{i=0}^{B-1} L_q(i) \tag{16}$$

where $L_p(i), L_q(i)$ are the simplified notations of the loss function for ith sample in batch.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3120265, IEEE Access

Types	Models	MRR	Recall@1	Recall@5	Recall@10
Simple Statistics	Popularity Choice	0.0080	0.0019	0.0019	0.019
Traditional Machine Learning Methods	Random Forest	0.0077 (0.0002)	0.0019 (0.0002)	0.0083 (0.0003)	0.0155 (0.0005)
	Logistic Regression	0.1354 (0.0011)	0.0754 (0.0010)	0.1888 (0.0013)	0.2535 (0.0015)
	Simple MLP	0.1685 (0.0031)	0.0955 (0.0006)	0.2349 (0.0028)	0.3141 (0.0053)
Deep Learning Methods	Vanilla Sum	0.2178 (0.0040)	0.1300 (0.0035)	0.3044 (0.0046)	0.3965 (0.0058)
	Bi-directional LSTM	0.2024 (0.0071)	0.1199 (0.0057)	0.2831 (0.0094)	0.3681 (0.0104)
	Deep Sets	0.2126 (0.0082)	0.1258 (0.0065)	0.2975 (0.0109)	0.3872 (0.0114)
	Reciptor	0.2103 (0.0011)	0.1249 (0.0011)	0.2936 (0.0015)	0.3807 (0.0024)
	RecipeBowl	**0.2261 (0.0020)**	**0.1358 (0.0020)**	**0.3166 (0.0021)**	**0.4072 (0.0023)**

TABLE 2: Evaluation results. Best results are in bold. All experiments were repeated 10 times with different splits. All results except Popularity Choice have mean and standard deviation for each metric. All results compared to RecipeBowl have a p-value below 0.05 as result of significance test

Purposes	Ablations	MRR	Recall@1	Recall@5	Recall@10
Our Model	**RecipeBowl**	**0.2281**	**0.1379**	**0.3182**	**0.4086**
Recipe Context	RecipeBowl -Cooking Tags	0.1463	0.0797	0.2035	0.2790
	RecipeBowl -Recipe Prediction Layer	0.2153	0.1286	0.3015	0.3899
Pre-trained Ingredient Embeddings	RecipeBowl -FlavorGraph +word2vec	0.2044	0.1218	0.2857	0.3704
	RecipeBowl -FlavorGraph +random	0.2153	0.1275	0.3021	0.3902
Decoder	RecipeBowl -2-way Decoder	0.1343	0.0736	0.1863	0.2559
Loss Function	RecipeBowl -Euclidean +Dot-product	0.0511	0.0200	0.0697	0.1088
	RecipeBowl -In-batch Negatives	0.0579	0.0269	0.0792	0.1191

TABLE 3: Ablation test results. The best results are in bold. All results were obtained from experiments on the first random split of out dataset.

V. EXPERIMENTS

A. EXPERIMENTAL SETTING

We conducted experiments to evaluate and compare our proposed RecipeBowl's performance on recipe completion task with other model options. We firstly performed a simple preliminary experiment by giving each leave-one-out input set of ingredients the same list of ingredients sorted by their occurrence as target ingredient in the whole dataset. We denote this method as Popularity Choice. We selected traditional machine learning approaches for our baseline experiments to evaluate our proposed model architecture. We imported the pre-trained FlavorGraph embeddings and summed each of the input ingredients into a single 300-dimensional continuous vector [14]. We then concatenated it with its corresponding 630-dimensional cooking tag vector. As a result, the dimension of each input vector is 930. The baseline models that were used in this setting are Random Forest Classifier, Logistic Regression and MLP Classifier and were all imported from the Scikit-learn Python package [35]. They are multi-class classification models where the class labels are the 3,729 unique ingredients.

We additionally conducted baseline experiments on various types of Set Encoders to assess the use of our custom Set Transformer while retaining other model features in RecipeBowl such as the Decoder and use of cooking tag vectors. The baseline modules for the Set Encoders are the following,

- **Vanilla Sum:** The ingredient vectors from the FlavorGraph embedding lookup table are summed into a single set context vector for each recipe input. This resembles the continuous bag-of-words model [36].
- **Bidirectional LSTM:** Previously used in recipe embedding experiments by Li et al. [9], this module encodes a sequence of ingredients in both directions into a set context vector.
- **Deep Sets:** Introduced by Zaheer et al. and used in Lee et al.'s baseline experiments [10], the Deep Sets model is a permutation-invariant deep learning model that builds deeper element-wise and set-wise representations through a stack of layers [30].
- **Reciptor:** Adopted from Li et al.'s [9], the Reciptor model is a Set Transformer containing 2 ISABS, 1 PMA and 1 SAB. All inherent MABs have 4 attention heads while each ISAB has 16 trainable inducing vectors and the PMA has 2 trainable seed vectors.

As our version of Set Transformer (1 ISAB, 1 PMA) is used in the RecipeBowl architecture as the Set Encoder, we denote other deep learning model variants by their corresponding Set Encoder since the other components in the model architecture are fixed.

1) Model Training and Evaluation Metrics

We fit the traditional machine learning models into our large training dataset and evaluated their performance based on the predicted probabilities for each class (3,729 ingredients). The predicted list of probabilities were sorted for evaluative purposes. The deep learning architectures using various Set Encoder modules including RecipeBowl and its ablated

versions were trained to the maximum of 60 epochs with early stopping using the AdaBound optimizer [37]. All models were trained on the same training dataset and evaluated on the same test dataset as well. The hyperparameters for RecipeBowl that were estimated using the validation dataset and are available in the anonymous code repository.

We retrieved the predicted ingredient vectors of test dataset from the deep learning models including RecipeBowl, to generate ranking-based recommendation results. We then calculated a pairwise matrix of cosine similarity scores between the vector predictions for the *incomplete* ingredient set in test dataset and 3,729 actual ingredient vectors. We sorted the similarity scores to obtain a ranked list of recommended ingredients. Both lists are used for evaluation based on multi-item recommendation. We used Mean Reciprocal Rank (MRR) and Recall@K (K=1,5,10) to evaluate the recommendation results derived from the scores.

B. EXPERIMENTAL RESULTS

1) Model Performance

We made 10 different 80%/10%/10% random splits of our dataset to perform the main experiments on the recipe completion task. In addition, the random initialization of trainable parameters in deep learning models is different according to each of the random split. For each model configuration including the traditional machine learning models, we calculated the mean and standard deviation of each evaluation metric MRR, Recall@1, Recall@5 and Recall@10. We also conducted statistical tests to obtain p-values to prove RecipeBowl's statistical significance.

Table 2 shows the evaluation results of RecipeBowl and other baseline models. Results show that RecipeBowl achieved the highest performance in all metrics (MRR: 0.2261 (0.0020), Recall@1: 0.1358 (0.0020), Recall@5: 0.3166 (0.0021), Recall@10: 0.4072 (0.0023)). According to the results, utilizing several model-related components and additional features such as tag vectors helped RecipeBowl outperform other model options. It is notable that our version of the Set Transformer used in RecipeBowl has less model complexity than Li et al.'s version used in Reciptor [9] which led to better generalization results (MRR: 0.2103 (0.0011)).

2) Ablation Study on General Model Architecture

We performed ablation tests to find whether 1) utilizing recipe context information, 2) employing a negative likelihood loss function based on a softmax over euclidean distances with in-batch negatives, 3) using the pre-trained FlavorGraph vectors as initial embeddings for RecipeBowl and 4) adding a Decoder before projecting the set context vectors into another embedding space were effective or detrimental to RecipeBowl's training.

Table 3 shows the ablation results on RecipeBowl. All ablation experiments were performed using the first random split of our dataset. The ablation results illustrate the importance of selecting the right loss criteria for training RecipeBowl. Combining the effects of distance metric learning and in-batch negatives randomly containing both easy and hard (highly related to targets) ingredient negatives seemingly benefit RecipeBowl's performance.

In terms of model architecture, results show RecipeBowl's dependency on both the 2-way Decoder (MRR: 0.1343) and tag vectors (MRR: 0.1463). Considering the risks of multi-task learning, our ablation results show that recipe prediction task does not negatively affect RecipeBowl but rather boosts by a small amount (MRR: 0.2153). Though we imported the pre-trained FlavorGraph embeddings from Park et al.'s work, our ablation results show less difference in performance (MRR: 0.2153) leaving room for further investigation.

VI. ANALYSIS

A. RECIPEBOWL RECOMMENDATIONS

The RecipeBowl accepts any ingredient sets and recommends additional ingredients and candidate recipes which is illustrated in Figure 3. In Table 4, we show six different user input examples with different cooking tags. Here, we recommend top 10 ingredients and top 5 recipes. Our model made accurate ingredient predictions (bold-faced) for the first four examples. In addition, RecipeBowl provided relevant and plausible alternatives other than the actual target ingredient in those examples. Moreover, RecipeBowl served its purpose as a 2-way recommender given the recommended recipe titles that are relevant to both the user input and cooking tags.

For the last two examples in Table 4, although RecipeBowl did not predict the correct target ingredients (bold-faced, torillas, cooked white rice), there were still meaningful suggestions. For the Mexican dish, our model recommended tortilla chips at top 1 while tortillas are ranked third. For the Rice dish, while our model did not predict perfectly (cooked white rice, out of top 10), most of the recommendations are still aligned with the target ingredient (e.g. wild rice, yellow rice). We expect RecipeBowl's flexibility and understanding in cooking to be helpful in making cooking choices.

B. ANALYSIS ON PREDICTIONS IN EMBEDDING SPACE

Figure 4. shows the distribution of both target and predicted embeddings vectors. While the predicted ingredients are close to their corresponding targets, the embedding seemed to be clustered into eight categories overall. This shows that the RecipeBowl model learned not only the optimal ingredient for the given set but also recipe categorical features.

Figure 5. shows the distribution of sixteen target embeddings and their corresponding predictions which is illustrated in the *Embedding Space* of Figure 3. In this analysis, 16 target ingredients were randomly selected according to their ingredient categories along with their predictions in the test dataset. Most of the predicted ingredients tended to form clusters corresponding to the selected targets. Moreover, some target ingredients are centered in the prediction clusters (e.g. mashed bananas, bread, chicken breasts). Interestingly, clusters that belong to the same ingredient category (e.g. pork chops, chicken wings, chicken breasts) tend to be relatively close to each other. We also found target pairs

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3120265, IEEE Access

Cooking Tags	User Input	Top 10 Recommendations (Ingredients) / Top 5 Recommendations (Recipes)
Main Dish Chicken Oven	chicken breasts, fresh cilantro, red onions, barbecue sauce, mozzarella cheese, cornmeal, olive oil	**pizza dough (target, top1 recommended)**, pizza crust, flat bread, pimento cheese, crisp bread, taco seasoning, grape tomatoes, olives, plum tomatoes, pizza sauce
		BBQ Chicken Pizza - California Pizza Kitchen Style (original), *Ranch Chicken Burgers, Crispy Chicken Strips, Town House Chicken, Spiced Burgers, Weight Watchers Chicken Cordon Bleu*
Breads Muffins Baking	butter, sugar, vanilla, cream cheese evaporated milk, boiling water	**dark chocolate (target, top1 recommended)**, banana chips, plain flour, vanilla essence, banana extract, unsweetened cocoa powder, dark chocolate chips, peanuts, oats, mini chocolate chips
		Banana Muffins With Chocolate Peanut Frosting (original), *Martha Stewart's Peanut-Butter Surprises, Stout Gingerbread Cupcakes With Cream, Better Than Toll House Cookies!, Lunchbox Peanut Butter Brownies, Great Chocolate Chip Cookies*
Tex-Mex Dinner Party Cocktails	grapefruit juice, simple syrup, tequila, lime juice, ice	**grapefruits (target, top1 recommended)**, guava nectar, guava juice, margarita mix, pomegranate juice, lime wedge, Licor 43, Coke, maraschino cherry juice, cola
		Siesta - Grapefruit Margarita (original), *Citrus Cranberry Delight, Aida's Curse Cocktail, Pineapple, Watermelon & Strawberry Slushes, Cobalt Colada, Frozen Blender Mojito*
Japanese Appetizers Lunch	nori, hot sauce, tuna, lettuce, mayonnaise	**sushi rice (target, top1 recommended)**, tobiko, white sesame seeds, imitation crab sticks, tuna fish, rice cakes, tuna steak, crabsticks, wasabi, soybean paste
		Spicy Tuna Salad Roll (original), *Oven Hot Ham & Cheese Sandwiches, General Tso's Chicken Wraps, Bourbon Street Deli Special Sandwich, Hg's Southwest Burritofest!*
Mexican Ground beef Spicy	lean ground beef, sharp cheddar cheese, ground cumin, garlic clove, bell pepper, chili powder, vegetable oil, eggs, milk, onion, salt	tortilla chips, diced green chilies, jalapeno, **tortillas (target, top4 recommended)**, corn tortillas, taco sauce, corn tortilla chips, poblano chiles, refried beans, jalapeno pepper
		Casserole Quiche With Crisp-Fried Tortilla Pieces (original), *Italian Shepherd's Pie, Pastitsio Pie, Meatloaf Pot Roast, Bacon Cheeseburger Upside Down Pizza, Sweet and Sour Cocktail Meatballs*
Main-dish Rice Vegetables	chicken breasts, olive oil, garlic	balsamic vinegar, oregano, ground ginger, green peppers, cumin, ground black pepper, thyme, Italian seasoning, yellow onion, green bell pepper, **lime (target, out of top10)**
		Roasted Garlic Chicken (original), *Sesame Chicken, Italian Wrap Chicken Breast, Teriyaki Chicken, Apricot Rosemary Chicken*

TABLE 4: RecipeBowl Recommendation Results. Examples of these six cases are all from the test set. For the first four examples, RecipeBowl model accurately predicted the target ingredient (bold-faced), but the last two examples, it did not. However, the recommendations still seem reasonable.

bread flour&yeast and cocoa&chocolate being close to each other along with their prediction clusters. Bread flour and yeast are known to be used together in most recipes while cocoa is one of the materials for making chocolate chips. These observations show that the RecipeBowl model learned ingredient relationships during training.

C. ANALYSIS ON SET REPRESENTATION VECTORS

Figure 6. shows clustermaps of 150 randomly sampled set context embedding vectors. The *Set Context Embeddings* according to Figure 3 are the set-wise vectors from the *Set Encoder*, prior to being propagated to the *2-way Decoder*. We selected blueberries, apples, buttermilk and chocolate chips from the previous list used in t-sne visualization and extracted incomplete ingredient lists with equal size of 150 containing each of them from the test dataset. We then used the Set Encoder of RecipeBowl to generate 4 groups of 150 set context vectors and visualized a clustermap for each group. We selected blueberries and apples since both of them are fruit ingredients used in a wide variety of dishes. On the contrary, we additionally selected buttermilk and chocolate chips that may be used in limited recipe categories such as bakery and desserts. The clustermaps shown in Figure 6. seemed to show distinctive clusters which brought interesting insight. For example, apples can be used in a wide range of recipes such as sweet desserts (*Caramel Apple*), bakery foods (*Apple Maple Muffin*) or as sauces in meat-based dishes (*Apple Pork Chops*) [38], [39]. Buttermilk is widely used in bakery products due to its nutritional value and taste enhancement features [40]. We can observe that among the sampled 150 set context vectors including buttermilk, most of them were used in bakery recipes (*Basic Chocolate Cake*). Overall, RecipeBowl can distinguish different types of recipe context according to the uses of a particular ingredient. The detailed clustermaps for these ingredients can be found in the code repository.

D. ANALYSIS ON ATTENTION WEIGHTS IN SET ENCODER

Figure 7. shows attention weights of the input ingredients. We extracted and aggregated the attention values computed in the first MAB of the ISAB in RecipeBowl's Set Encoder in Figure 3 and normalized them with min-max scaling. We studied the recommendation examples and observed which ingredient seems to have high influence towards building the set context vector. For *Spicy Tuna Salad Roll*, nori recieved the highest attention which helped RecipeBowl understand the set input is mainly Japanese cuisine. For *BBQ Chicken Pizza*, chicken breasts, fresh cilantro and red onions were majorly attentive interestingly compared to mozzarella cheese. Lastly, the input set for *Casserole Quiche* contained ingredients mainly used in Mexican cuisine such as bell peppers and

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3120265, IEEE Access

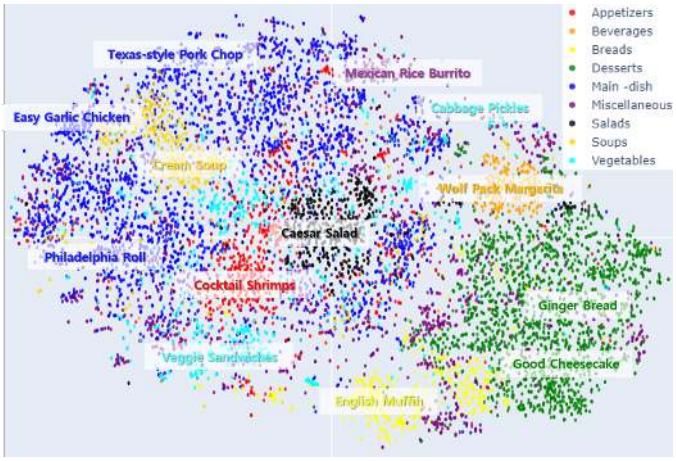

FIGURE 4: Predicted ingredient vector embeddings. (10,000 sampled in test).

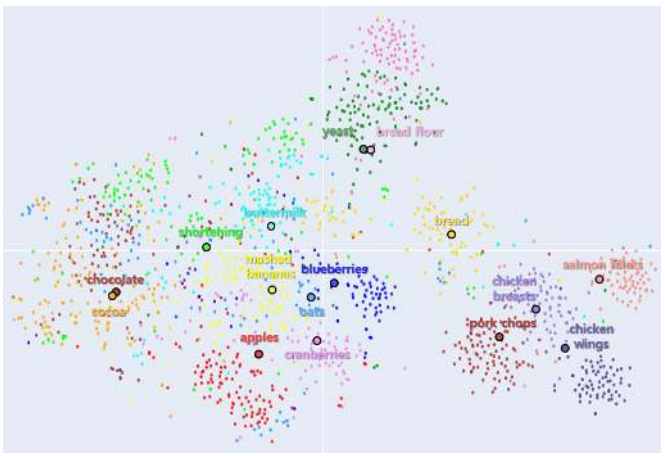

FIGURE 5: Predicted ingredient vector embeddings. (Sampled according to 16 target ingredients).

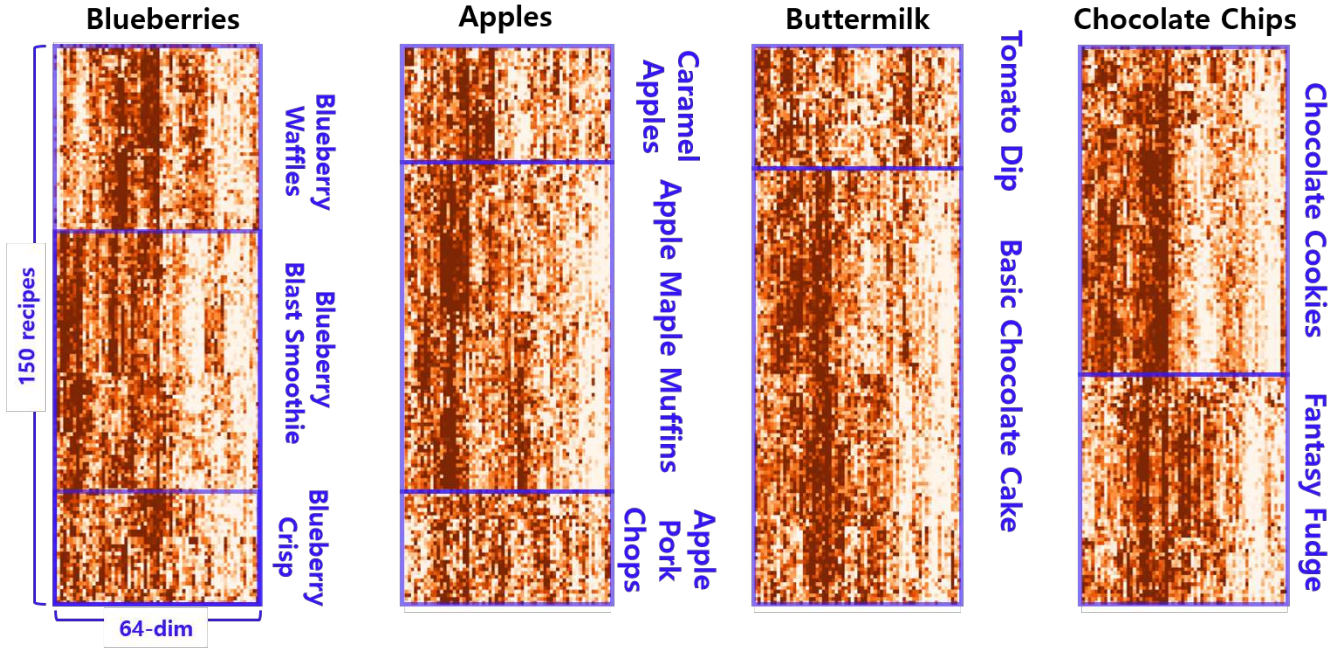

FIGURE 6: Clustermaps of 150 set context vector embeddings for each target ingredient. The dimension size of each set context vector is 64. The recipe names for each recipe cluster are representative examples.

chili peppers [41], [42]. In turn, we speculate that RecipeBowl was able to predict tortilla chips based on highly attentive values of the above ingredients as tortilla-related ingredients are also commonly used in Mexican dishes.

VII. CONCLUSION & FUTURE WORK

We introduce RecipeBowl, a set-based cooking recommender for candidate ingredients and recipes. To train the model, we formulate a supervised learning recipe completion setting using an extended dataset from Reciptor [9]) and employing the Set Transformer [10] framework to encode ingredients into a set context representation. Based on the evaluation results from the formulated recipe completion task, our model showed best results among other set encoding variation baselines and traditional machine learning algorithms. Recommendation results demonstrate RecipeBowl's ability to generate both plausible and diverse recommendations for a given set of ingredient. We performed in-depth model analysis on RecipeBowl in a bottom-to-top fashion 3 starting from the predicted Embedding Space where the vector embeddings formed meaningful clusters. We also investigated the visualizations of the set context vectors which are the direct outputs from the Set Encoder and examined the attention weights extracted from the Set Encoder itself and found them

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3120265, IEEE Access

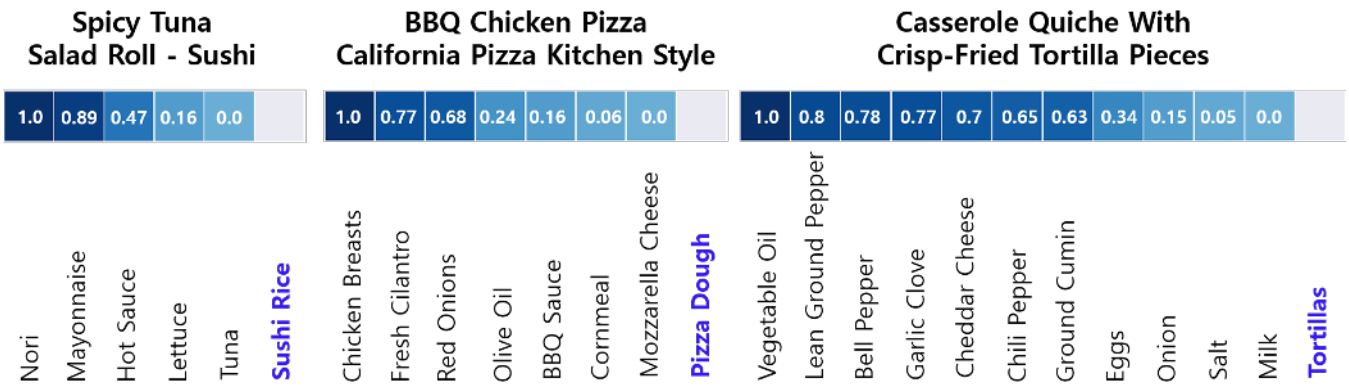

FIGURE 7: Attention weights of ingredients for each set input. Bold-faced are target ingredients. The rest are input ingredients with attention values transformed by min-max normalization for each recipe.

supportive to our model's performance.

While RecipeBowl was able to suggest both appropriate ingredients and recipe candidates for a given set of other ingredients, some recipe candidates seemed inconsistent with the suggested ingredients. We plan to improve RecipeBowl by encouraging it to recommend recipe candidates related to some of its suggested ingredients. Though our RecipeBowl exploited our custom-made Set Transformer to be trained successfully on recipe completion, we plan to improve the Set Encoder to extract richer cooking knowledge and provide better interpretability. In addition, we plan to incorporate nutritional features and consider dietary requirments during recommendation. Lastly, we plan to release an applicable version of RecipeBowl in the future.

REFERENCES

[1] Weiqing Min, Shuqiang Jiang, and Ramesh C Jain. Food recommendation: Framework, existing solutions and challenges. IEEE Transactions on Multimedia, 2019.

[2] Chun-Yuen Teng, Yu-Ru Lin, and Lada A Adamic. Recipe recommendation using ingredient networks. In Proceedings of the 4th Annual ACM Web Science Conference, pages 298–307, 2012.

[3] Marlies De Clercq, Michiel Stock, Bernard De Baets, and Willem Waegeman. Data-driven recipe completion using machine learning methods. Trends in Food Science & Technology, 49:1–13, 2016.

[4] Yong-Yeol Ahn, Sebastian E Ahnert, James P Bagrow, and Albert-László Barabási. Flavor network and the principles of food pairing. Scientific reports, 1:196, 2011.

[5] Neelansh Garg, Apuroop Sethupathy, Rudraksh Tuwani, Shubham Dokania, Arvind Iyer, Ayushi Gupta, Shubhra Agrawal, Navjot Singh, Shubham Shukla, Kriti Kathuria, et al. Flavordb: a database of flavor molecules. Nucleic acids research, 46(D1):D1210–D1216, 2017.

[6] Donghyeon Park, Keonwoo Kim, Yonggyu Park, Jungwoon Shin, and Jaewoo Kang. Kitchenette: Predicting and ranking food ingredient pairings using siamese neural networks. Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, 2019.

[7] Amaia Salvador, Nicholas Hynes, Yusuf Aytar, Javier Marin, Ferda Ofli, Ingmar Weber, and Antonio Torralba. Learning cross-modal embeddings for cooking recipes and food images. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3020–3028, 2017.

[8] Javier Marin, Aritro Biswas, Ferda Ofli, Nicholas Hynes, Amaia Salvador, Yusuf Aytar, Ingmar Weber, and Antonio Torralba. Recipe1m: A dataset for learning cross-modal embeddings for cooking recipes and food images. arXiv preprint arXiv:1810.06553, 2018.

[9] Diya Li and Mohammed J Zaki. Reciptor: An effective pretrained model for recipe representation learning. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pages 1719–1727, 2020.

[10] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In International Conference on Machine Learning, pages 3744–3753. PMLR, 2019.

[11] Jingjing Chen, Lei Pang, and Chong-Wah Ngo. Cross-modal recipe retrieval: How to cook this dish? In International Conference on Multimedia Modeling, pages 588–600. Springer, 2017.

[12] Bin Zhu, Chong-Wah Ngo, Jingjing Chen, and Yanbin Hao. R2gan: Cross-modal recipe retrieval with generative adversarial network. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 11477–11486, 2019.

[13] Steven Haussmann, Oshani Seneviratne, Yu Chen, Yarden Ne'eman, James Codella, Ching-Hua Chen, Deborah L McGuinness, and Mohammed J Zaki. Foodkg: a semantics-driven knowledge graph for food recommendation. In International Semantic Web Conference, pages 146–162. Springer, 2019.

[14] Donghyeon Park, Keonwoo Kim, Seoyoon Kim, Michael Spranger, and Jaewoo Kang. Flavorgraph: a large-scale food-chemical graph for generating food representations and recommending food pairings. Scientific Reports, 11, 2021.

[15] Naoki Shino, Ryosuke Yamanishi, and Junichi Fukumoto. Recommendation system for alternative-ingredients based on co-occurrence relation on recipe database and the ingredient category. In 2016 5th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI), pages 173–178. IEEE, 2016.

[16] Kuan-Hung Liu, Hung-Chih Chen, Kuan-Ting Lai, Yi-Ying Wu, and Chih-Ping Wei. Alternative ingredient recommendation: A co-occurrence and ingredient category importance based approach. In PACIS, page 298, 2018.

[17] Mouzhi Ge, Mehdi Elahi, Ignacio Fernaández-Tobías, Francesco Ricci, and David Massimo. Using tags and latent factors in a food recommender system. In Proceedings of the 5th International Conference on Digital Health 2015, pages 105–112, 2015.

[18] Ifeoma Adaji, Czarina Sharmaine, Simone Debrowney, Kiemute Oyibo, and Julita Vassileva. Personality based recipe recommendation using recipe network graphs. In International Conference on Social Computing and Social Media, pages 161–170. Springer, 2018.

[19] Tossawat Mokdara, Priyakorn Pusawiro, and Jaturon Harnsomburana. Personalized food recommendation using deep neural network. In 2018 Seventh ICT International Student Project Conference (ICT-ISPC), pages 1–4. IEEE, 2018.

[20] Angelos Nezis, Haris Papageorgiou, Pavlos Georgiadis, Petr Jiskra, Dimitris Pappas, and Maria Pontiki. Towards a fully personalized food recommendation tool. In Proceedings of the 2018 International Conference on Advanced Visual Interfaces, pages 1–3, 2018.

[21] Elizabeth Gorbonos, Yang Liu, and Chính T Hoàng. Nutrec: nutrition ori-

ented online recipe recommender. In 2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI), pages 25–32. IEEE, 2018.

[22] Wenjie Wang, Ling-yu Duan, Hao Jiang, Peiguang Jing, Xuemeng Song, and Liqiang Nie. Market2dish: Health-aware food recommendation. arXiv preprint arXiv:2012.06416, 2020.

[23] Meng Chen, Xiaoyi Jia, Elizabeth Gorbonos, Chinh T Hoang, Xiaohui Yu, and Yang Liu. Eating healthier: Exploring nutrition information for healthier recipe recommendation. Information Processing & Management, 57(6):102051, 2020.

[24] Jose M Ordovas, Lynnette R Ferguson, E Shyong Tai, and John C Mathers. Personalised nutrition and health. Bmj, 361, 2018.

[25] Paula Fermín Cueto, Meeke Roet, and Agnieszka Słowik. Completing partial recipes using item-based collaborative filtering to recommend ingredients. arXiv preprint arXiv:1907.12380, 2019.

[26] Juan Ramos et al. Using tf-idf to determine word relevance in document queries. In Proceedings of the first instructional conference on machine learning, volume 242, pages 29–48. Citeseer, 2003.

[27] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Advances in neural information processing systems, pages 5998–6008, 2017.

[28] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. arXiv preprint arXiv:1607.06450, 2016.

[29] Harrison Edwards and Amos Storkey. Towards a neural statistician. arXiv preprint arXiv:1606.02185, 2016.

[30] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. In Advances in neural information processing systems, pages 3391–3401, 2017.

[31] Jacob Goldberger, Geoffrey E Hinton, Sam Roweis, and Russ R Salakhutdinov. Neighbourhood components analysis. Advances in neural information processing systems, 17:513–520, 2004.

[32] Ruslan Salakhutdinov and Geoff Hinton. Learning a nonlinear embedding by preserving class neighbourhood structure. In Artificial Intelligence and Statistics, pages 412–419, 2007.

[33] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. arXiv preprint arXiv:2004.11362, 2020.

[34] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 4004–4012, 2016.

[35] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12:2825–2830, 2011.

[36] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781, 2013.

[37] Liangchen Luo, Yuanhao Xiong, Yan Liu, and Xu Sun. Adaptive gradient methods with dynamic bound of learning rate. arXiv preprint arXiv:1902.09843, 2019.

[38] Joan Morgan. The new book of apples. Random House, 2013.

[39] K.R Price, T Prosser, A.M.F Richetin, and M.J.C Rhodes. A comparison of the flavonol content and composition in dessert, cooking and cider-making apples; distribution within the fruit and effect of juicing. Food Chemistry, 66(4):489–494, 1999.

[40] Abdelmoneim H Ali. Current knowledge of buttermilk: Composition, applications in the food industry, nutritional and beneficial health characteristics. International Journal of Dairy Technology, 72(2):169–182, 2019.

[41] Jeffrey Pilcher. The globalization of mexican cuisine. History Compass, 6(2):529–551, 2008.

[42] Edgar Rojas-Rivas, Alicia Rendón-Domínguez, José Alberto Felipe-Salinas, and Facundo Cuffia. What is gastronomy? an exploratory study of social representation of gastronomy and mexican cuisine among experts and consumers using a qualitative approach. Food Quality and Preference, 83:103930, 2020.

KEONWOO KIM received the B.S. degree in computer science from Korea University, South Korea, in 2018, where he is currently pursuing the Ph.D. degree in computer science. His current research focuses on developing effective data representation methods and applying them to various research domains such as food science, material science and bio-informatics.

DONGHYEON PARK received the B.S. degree in computer science, in 2015, and the M.S. degree in the Interdisciplinary Graduate Program in Bioinformatics, in 2017, and Ph.D. degree in computer science, in 2020, from Korea University at Seoul, South Korea. His current research interests include food-informatics and natural language processing in general. He is specifically interested in personalized food-recipe recommendation and nutrition precision with Artificial Intelligence techniques.

MICHAEL SPRANGER received his Diploma from the Humboldt- Universitaet zu Berlin (Germany) in 2008 and a PhD from the Vrije Universiteit in Brussels (Belgium) in 2011 (both in Computer Science). Michael is the COO of Sony AI and a Senior Researcher at Sony CSL. His work focuses on fundamentals of AI, intelligent agents. His most recent work explores creativity in science and gastronomy.

KANA MARUYAMA is an AI Engineer at Sony AI. She received a master's degree in computer science from Hokkaido University, Japan. In the past, she worked on developing camera products, fusion of web technology and embedded technology, causal analysis, and Japanese NLP technologies. In Sony AI, she currently focuses on developing AI technologies for enhancing the creativity of chefs.

JAEWOO KANG received the B.S. degree in computer science from Korea University, Seoul, South Korea, in 1994, the M.S. degree in computer science from the University of Colorado at Boulder, CO, USA, in 1996, and the Ph.D. degree in computer science from the University of Wisconsin-Madison, WI, USA, in 2003. From 1996 to 1997, he was a Technical Staff Member with AT&T Labs Research, Florham Park, NJ, USA. From 1997 to 1998, he was a Technical Staff Member with Savera Systems Inc., Murray Hill, NJ, USA. From 2000 to 2001, he was the CTO and a co-founder of WISEngine Inc., Santa Clara, CA, USA, and Seoul. From 2003 to 2006, he was an Assistant Professor with the Department of Computer Science, North Carolina State University, Raleigh, NC, USA. Since 2006, he has been a Professor with the Department of Computer Science, Korea University. He also serves as the Department Head for the Interdisciplinary Graduate Program in Bioinformatics with Korea University.