Recipe Representation Learning with Networks

Yijun Tian
University of Notre Dame, USA
yijun.tian@nd.edu

Chuxu Zhang
Brandeis University, USA
chuxuzhang@brandeis.edu

Ronald Metoyer
University of Notre Dame, USA
rmetoyer@nd.edu

Nitesh V. Chawla
University of Notre Dame, USA
nchawla@nd.edu

ABSTRACT

Learning effective representations for recipes is essential in food studies for recommendation, classification, and other applications. Unlike what has been developed for learning textual or cross-modal embeddings for recipes, the structural relationship among recipes and food items are less explored. In this paper, we formalize the problem *recipe representation learning with networks* to involve both the textual feature and the structural relational feature into recipe representations. Specifically, we first present *RecipeNet*, a new and large-scale corpus of recipe data to facilitate network based food studies and recipe representation learning research. We then propose a novel heterogeneous recipe network embedding model, *rn2vec*, to learn recipe representations. The proposed model is able to capture textual, structural, and nutritional information through several neural network modules, including textual CNN, inner-ingredients transformer, and a graph neural network with hierarchical attention. We further design a combined objective function of node classification and link prediction to jointly optimize the model. The extensive experiments show that our model outperforms state-of-the-art baselines on two classic food study tasks. Dataset and codes are available at https://github.com/meettyj/rn2vec.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; • **Information systems** → **Information systems applications**.

KEYWORDS

Recipe representation learning; Food network; Graph neural network

ACM Reference Format:
Yijun Tian, Chuxu Zhang, Ronald Metoyer, and Nitesh V. Chawla. 2021. Recipe Representation Learning with Networks. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management (CIKM '21), November 1–5, 2021, Virtual Event, Australia.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3459637.3482468

CIKM '21, November 1–5, 2021, Virtual Event, Australia

ACM ISBN 978-1-4503-8446-9/21/11…$15.00
https://doi.org/10.1145/3459637.3482468

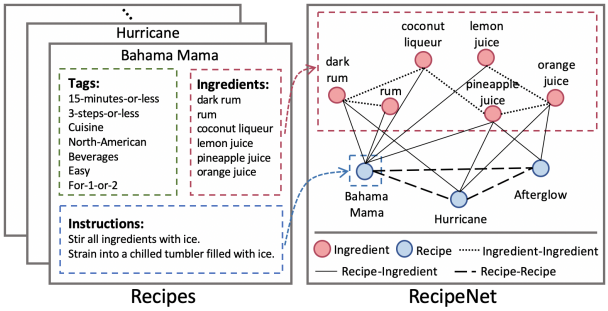

Figure 1: The illustration of *RecipeNet*. Recipe examples are shown on the left. The *RecipeNet* includes two types of nodes (ingredient node, recipe node with instruction) and three types of relations which connect these nodes.

1 INTRODUCTION

Large-scale food data offers rich knowledge about food and can help tackle many central issues of human society. Food computing applies computational approaches for acquiring and analyzing heterogeneous food data to address food-related issues in health, biology, gastronomy, and agronomy [26, 28]. In particular, recipe representation learning embeds the recipes in a latent space, which helps capture the hidden information from massive food datasets, and further supports various food studies, such as analyzing culinary habits [25, 34], recipe healthiness estimation [33], recipe question answering [1, 16], recipe recommendation [6, 37], and recipe classification [15, 22].

Most previous work addresses recipe representation learning in a specific image-recipe retrieval task where the goal is to map the recipe text with food images to obtain the cross-modal embeddings [4, 10, 23, 35, 42]. Solutions to this task take advantage of food images but images are not always available for recipe tasks and it is often difficult to find an accurate image for any given recipe. Unlike leveraging food images, Reciptor [22] proposed to learn recipe representations using only the textual information by aligning the ingredients with step-wise instructions. However, the affinity between ingredients as well as the underlying similarity between recipes are ignored. Even though they used an external knowledge base, FoodKG [12], to sample similar recipes, each recipe only appeared once in the sampled triplet, and the generated similarity score is used as the strict ground truth label. This approach overlooked the case where a recipe might have multiple similar ones, and the similarity could be different for each downstream task, which should be learned through jointly training.

Therefore, we propose the problem *recipe representation learning with networks*, which leverages the advances in network learning to address recipe representation learning. With the incorporation of relationship among recipes and ingredients through a network, as well as training a neural network over them jointly, the learned recipe embeddings are more comprehensive and informational. Due to the textual and the structural complexity, understanding recipes in a network demands a large and comprehensive recipe network. Hence, it should not be surprising that the lack of research work on this topic could be the result of missing such a dataset. With the aim of learning expressive recipe representations, we create and release *RecipeNet*, a new and large-scale corpus of recipe data structured in a network. As shown in the Fig. 1, *RecipeNet* contains two types of nodes and three types of relations among them.

Based on the created *RecipeNet*, we propose a novel heterogeneous recipe network embedding model, *rn2vec*, to learn recipe representations. Three major components of recipes (*i.e.,* instructions, ingredients, and nutrients) are all considered in *rn2vec*. Specifically, we first extract the nutrients for each ingredient from USDA nutritional database [38] and use TextCNN [17] to encode the instructions. Then we leverage a set transformer [21] based neural network to encode the ingredients and the pairwise- and higher-order interactions between them. After that, we design a graph neural network (GNN) to capture the structural information in *RecipeNet*. To fully encode this information, we propose a node-level attention module to distinguish the subtle difference of neighbor nodes under a specific relation. We also introduce a relation-level attention module to select the most meaningful relations surrounding a node and automatically fuse them with proper weights. Based on these two attention modules, *rn2vec* can determine the significance of each node and relation for learning recipe representations. We further design a combined objective function of node classification and link prediction to jointly optimize the model. Two classic evaluation tasks, namely, cuisine category classification and region prediction [22], are used to verify the quality of the learned embeddings. To summarize, our main contributions are as follows:

- To the best of our knowledge, this is the first attempt to study recipe representation learning with networks. We create and release *RecipeNet*, a new and large-scale corpus of recipe data, that enables superior recipe representation learning and further facilitates network based food studies.
- We propose a novel heterogeneous recipe network embedding model, *rn2vec*, for recipe representation learning. *rn2vec* is able to capture both textual and structural information and learn effective representations through several neural network modules, including textual CNN, inner-ingredients transformer, and a graph neural network with hierarchical attention. We further design a novel combined objective function of node classification and link prediction to jointly optimize the model.
- We conduct extensive experiments to evaluate the performance of our model. The results show the superiority of *rn2vec* by comparing with state-of-the-art baselines on two classic food study tasks: cuisine category classification and region prediction.

2 PROBLEM DEFINITION

In this section, we first introduce the concepts of recipe representation learning and content-associated heterogeneous recipe network. We then formally define the problem of recipe representation learning with networks.

Definition 2.1. **Recipe Representation Learning.** *Recipe representation learning aims at learning effective representations of recipes. In particular, suppose each recipe consists of various contents (e.g., cooking instructions, ingredients, nutrients, images). Given a set of N recipes* $\{R_1, R_2, ..., R_N\}$*, the task is to transform each recipe* R_i $(i < N)$ *into d-dimensional embeddings* $\mathcal{E} \in \mathbb{R}^d$ *with associated content information encoded.*

Definition 2.2. **Content-associated Heterogeneous Recipe Network.** *A content-associated heterogeneous recipe network is defined as a network* $G = (V, E, C)$ *with multiple types of nodes V (i.e., recipe node and ingredient node) and edges E (i.e., the relations among these nodes). In addition, nodes are associated with unstructured contents C, e.g., instructions, nutrients, or images.*

Problem 1. **Recipe Representation Learning with Networks.** *The task is to design a learning model* $\mathcal{F}_\Theta$ *with parameters* Θ *to learn recipe embeddings* $\mathcal{E} \in \mathbb{R}^d$ *on a recipe network* $G = (V, E, C)$*, while encoding both the structural relationship among nodes and the associated unstructured contents. The learned recipe representations can be utilized in various downstream tasks such as cuisine category classification and region prediction.*

To solve this problem, we first build a content-associated heterogeneous recipe network *RecipeNet*, and then develop a recipe network embedding model, *rn2vec*, to learn recipe representations.

3 RECIPENET

We create *RecipeNet* (Fig. 2 (a)) in two steps. First, we adopt a large set of recipes from Recipe1M [23, 35], extract the weight of each ingredient, and calculate the nutritional information of each recipe. Second, we transform the recipe dataset into *RecipeNet* with recipe nodes, ingredient nodes, and edges between the nodes.

3.1 Weights and Nutrients Calculation

Recipe1M [23, 35] contains 1,029,720 recipes collected from multiple cooking websites. However, the category information associated with these recipes is imbalanced and scarce. To assign each recipe a more nuanced and precised category, we crawl recipe tags from food.com[1], and further use 507,834 recipes in Recipe1M that are from the same website as our base recipe dataset [22].

Weights Calculation. The ingredient information from Recipe1M contains the ingredient, quantity, and unit information altogether in a single sentence, *e.g.,* 1 tablespoon olive oil (quantity-unit-ingredient). In order to extract the weight information, we leverage the natural language toolkit *nltk*[2] to tag the part-of-speech (pos) of each word in the ingredient sentence (*e.g.,* [('1', 'CD'), ('tablespoon', 'NN'), ('olive', 'JJ'), ('oil', 'NN')]). We then use the pos to identify the ingredients that follow the sequence pattern 'quantity-unit-ingredient', and extract the patterns correspondingly. After

[1]https://www.food.com
[2]https://www.nltk.org

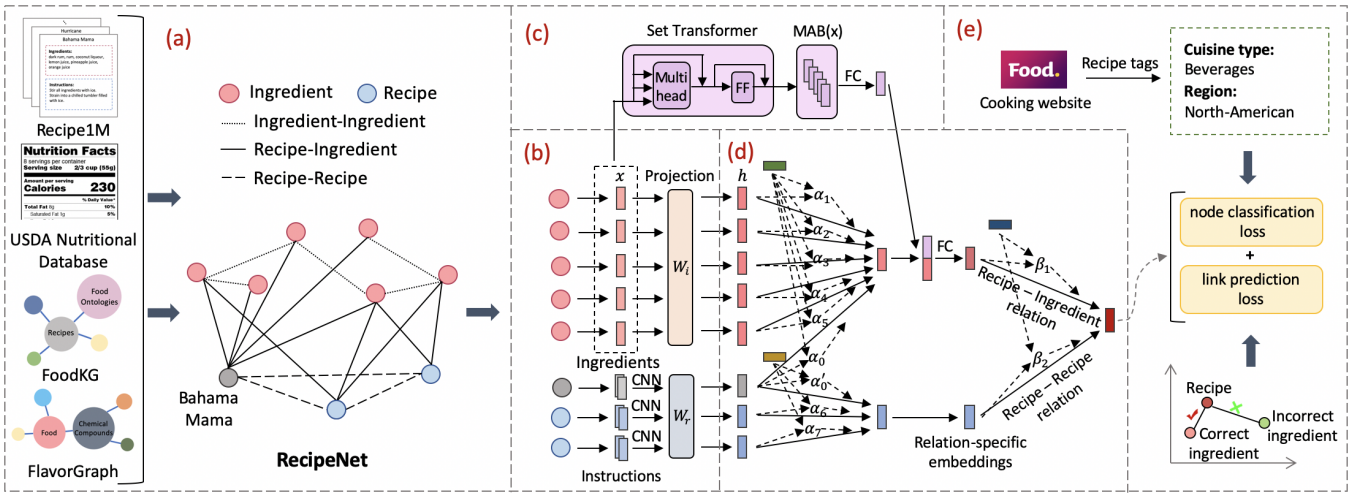

Figure 2: The overall framework of our model. (a) The construction of *RecipeNet*; (b) Encoding instructions through textual CNN and ingredients with nutrient vectors; (c) Inner-ingredients transformer: encoding the interactions among ingredients with permutation invariant preserved using set transformer; (d) GNN with hierarchical attention: encoding the nodes with same type through node-level attention, and then fusing the embeddings over different relations into the final recipe representations through relation-level attention; (e) A combined objective function: applying the recipe tags from food.com to generate ground-truth labels and formulate objective as the joint loss of the node classification task (cuisine category classification or region prediction) and the link prediction task (recipe-ingredient relation prediction).

Table 1: The statistics of *RecipeNet*.

Component	Name	Number	Data Source
Nodes	recipe	77,333	Recipe1M
	ingredient	9,271	Recipe1M
Edges	recipe-ingredient	524,671	Recipe1M
	recipe-recipe	850,354	FoodKG
	ingredient-ingredient	148,168	FlavorGraph

extracting the sequence patterns, we focus on those recipes that have complete sequence pattern for each ingredient to avoid noise. In total, 95,192 unique recipes are filtered out in this stage and used for further processing. In order to accurately calculate the weight of each ingredient, we go through each ingredient and only choose those that have measurable units [23]. We select 88,136 recipes that contain measurable units for all the ingredients and calculate the weight for each ingredient (*gram* is used as a standard unit).

Nutritional Information. Once we finish the previous stage, we match the ingredients with instances in the USDA nutritional database [38] to get the nutritional information. Given an ingredient in our dataset, we go through each unigram in the ingredient name to see if the word appears in any food instance of the USDA nutritional database. We take the intersection of multiple returned unigram search results as our potential matching set, if any, or simply combine all of the search results as our potential matching set. In order to select one food instance from this potential matching set to match with the ingredient, we first filter out three results shortest in length, then choose the one with the lowest sum index of the appeared unigrams. In other words, we prefer the one with the shortest length where the keywords appear earlier than other words. For example, given a search ingredient *olive oil* and the returned potential matching set with multiple food instances, we filter out three results with shortest length (*i.e.,* 'oil, olive, salad or cooking', 'mayonnaise, reduced fat, olive oil', and 'oil, corn, peanut, and olive'). This step filters the redundant results such us 'keebler, town house, flatbread crisps, sea salt and olive oil crackers'. Next, we calculate the sum index of the unigrams appearing in these results and choose the one has lowest sum index as our mapping. This step helps us find the correct matching 'oil, olive, salad or cooking' given the sum index for olive oil is 1 (1+0), while for other two is 7 (3+4) and 4 (4+0), respectively. This approach is valid because the primary ingredient usually appears in the front of the food instance sentence in the USDA nutritional database. In the end, we obtain 77,333 recipes with nutritional information and 9,271 unique ingredients after ignoring those recipes with unsuccessful matching ingredients.

3.2 Network Construction

Next, we construct the *RecipeNet* based on these recipes and ingredients. *RecipeNet* is comprised of two types of nodes (*i.e.,* recipe and ingredient) and three types of edges among them. More details of our constructed network are provided in Table 1.

We consider each recipe as a node with type *'recipe'* and each ingredient as a node with type *'ingredient'*. To associate each node with its content, we use the pretrained skip-instruction embeddings [23] to represent the recipe node, while formulating the nutrients of each ingredient into a vector and then using this nutrient vector to indicate the ingredient node. The nutrients are extracted through

the USDA nutritional database, spanning proximate component (*e.g.,* protein, fat), minerals (*e.g.,* calcium, iron), vitamins (*e.g.,* vitamin C, vitamin D), lipid components (*e.g.,* fatty acids), and amino acids (*e.g.,* tryptophan). We connect each recipe and its ingredients with an edge, denoting as recipe-ingredient relation, while the weight of each ingredient is used as the edge weight.

To build the recipe-recipe relation, we perform a similar recipe mining task from FoodKG [12], which is one of the largest recipe knowledge graphs and integrates recipe data from Recipe1M [23, 35] as well as food ontology from FoodOn [9]. We follow previous work [22] to import the FoodKg into Stardog[3], and then use SPARQL queries to mine similar recipes based on the textual features, including tags information, ingredient names, and the matched USDA food instances. After that, Stardog leverages machine learning and information retrieval based approaches to find the most similar recipes in FoodKG. Given a specific recipe as the input, the output is a set of similar recipes sorted by similarity score. We take those recipes with similarity score greater than 0.6[4] as similar pairs and build an edge between them as recipe-recipe relation, while the score serving as the edge weight between them.

In addition, we extract the ingredient-ingredient relation from FlavorGraph [30], which is a large-scale graph of food and chemical compound nodes for recommending food pairings. The pairing scores are calculated based on the co-occurring probabilities of ingredients using Normalized Pointwise Mutual Information (NPMI) [3], as shown in the KitcheNette [31]. The edges between every two ingredient nodes are constructed when satisfying one of the following conditions: (1) Each ingredient appears in more than 20 recipes in Recipe1M and both of them appear together in more than 5 recipes. (2) The calculated NPMI score for the ingredient pair is greater than 0.25, or the score is in the top 20 below the 0.25. After we building the *RecipeNet*, we propose our model *rn2vec* for recipe representation learning with networks.

4 RN2VEC

In this section, we describe our novel recipe network embedding model, *rn2vec*. As illustrated in Fig. 2, our model contains several major components. First, we use a textual CNN to encode instructions and use the nutrient vectors to represent each ingredient (Fig. 2 (b)). We then develop a transformer to learn the inner-ingredients set embedding (Fig. 2 (c)). After that, we introduce a graph neural network with node- and relation-level attentions to obtain recipe embeddings (Fig. 2 (d)). Finally, we design a novel combined objective function of node classification and link prediction to jointly optimize the model (Fig. 2 (e)).

4.1 Encoding Instructions and Ingredients

As shown in Fig. 2 (b), we use a textual CNN [17], a convolutional neural network for text, to encode the instruction sentence embeddings. Specifically, given a recipe with M sentence embeddings $\{x_{ins,1}, x_{ins,2}, ..., x_{ins,M}\}$ and a filter window size s, the encoding process can be formulated as:

$$x_{ins} = max[tan(W_{ins} \Big\|_{m=t}^{t+s-1} x_{ins,m} + b_{ins})], \tag{1}$$

where x_{ins} denotes the encoded instruction embedding, max is the max pooling operation, tan is the tangent function, W_{ins} and b_{ins} are learnable weights, t is the index of each instruction, and $\|$ is the concatenation operator. In addition, we use the nutrient vector of each ingredient to represent the ingredient embedding x_{ing}. We further use x_{ins} and x_{ing} as the input feature for recipe nodes and ingredient nodes, respectively. Due to the heterogeneity of input feature for different type of nodes, given a node v_i with type ϕ_i, we introduce a type-specific transformation matrix W_{ϕ_i} with output dimension d to project the input features of nodes with different types into the same embedding space. The projection process is formulated as follows:

$$\begin{aligned} x_i &= \begin{cases} x_{ins}, & \text{if } \phi_i = \text{recipe} \\ x_{ing}, & \text{if } \phi_i = \text{ingredient} \end{cases} \\ h_i &= W_{\phi_i} \cdot x_i, \end{aligned} \tag{2}$$

where $h_i \in \mathbb{R}^d$ and x_i are the projected and input feature of v_i, respectively. With this type-specific projection operation, the instruction and ingredient embeddings would be in the same embedding space, and the model can handle arbitrary types of input features.

4.2 Inner-ingredients Transformer

In order to better encode the ingredients in a recipe, the interaction between ingredients need to be incorporated. We utilize self-attention [40] as it has shown powerful performance on many tasks by addressing and learning the interdependence among the inputs. Specifically, given a recipe with n ingredients, we have a matrix Q of n query vectors with dimension d for each ingredient, denoted as $Q \in \mathbb{R}^{n \times d}$. An attention function maps Q to outputs using p key-value pairs with $K \in \mathbb{R}^{p \times d}, V \in \mathbb{R}^{p \times d}$:

$$Att(Q, K, V) = \omega\left(QK^\top\right)V, \tag{3}$$

where ω is the scaled activation function (here we use $softmax(\cdot/\sqrt{d})$ in our model). *Att(Q, K, V)* is the weighted sum of V, where a larger weight is generated when its corresponding key has a larger dot product with the query. Instead of computing a single attention function, we incorporate multi-head attention by projecting Q, k, V into different subspaces. Then the final multi-head output is computed by a linear transformation over the concatenation of all attention outputs. Considering two ingredients set $X_1, X_2 \in \mathbb{R}^{n \times d}$, the multi-head attention for X_1 on X_2 is defined as:

$$Attention(X_1, X_2) = \omega[(X_1 W_Q)(X_2 W_K)^\top](X_2 W_V), \tag{4}$$

where $W_Q, W_K, W_V \in \mathbb{R}^{d \times d}$ denote the trainable parameters. The multi-head self-attention on ingredient set X can be regarded as Attention(X, X). After that, this self-attention is fed into a Multi-head Attention Block (MAB) and Set Attention Block (SAB) [21] to get the transformer based inner-ingredients set embedding h_{trans}:

$$\begin{aligned} MAB(X_1, X_2) &= FFN[W_{mab} X_1 + Attention(X_1, X_2)], \\ SAB(X) &= MAB(X, X), \\ h_{trans} &= FC[SAB(X)], \end{aligned} \tag{5}$$

where MAB is an adaptation of the encoder block of the Transformer [40] without positional encoding and dropout, $W_{mab} \in \mathbb{R}^{d \times d}$ is a parameter matrix, and *FFN* is a feed forward neural network. In other words, SAB takes the ingredient set X as input and performs

[3] https://www.stardog.com/
[4] This threshold is empirically determined.

self-attention to generate a set of equal size. After encoding the ingredients through SAB, a fully connected layer (FC) is applied to obtain $h_{trans} \in \mathbb{R}^d$. That is, for each *recipe* node v_i, we compute the embedding $h_{i,trans}$ using all of its ingredients.

4.3 GNN with Hierarchical Attention

We introduce a graph neural network with hierarchical attention to model the interaction among nodes and learn node embeddings. For a node v_i, we first use the node-level attention to calculate each relation-specific embedding $h_{i,r}$ through relation r for all the relations connected to it, then use the relation-level attention module to combine all relation-specific embeddings and generate the final node embedding h_i.

Node-level Attention. To explain how we use node-level attention to get the relation-specific embeddings $h_{i,r}$, we start by describing a single node-level attention layer, as the sole layer to calculate node-level attention for each node. To compute the node embedding $h_{i,r}^{l+1}$ in layer $l+1$, the input of the node-level attention layer is a set of node embeddings from layer l: $\{h_1^l, h_2^l, \ldots, h_{N_{i,r}}^l\}$, $h_j^l \in \mathbb{R}^{d_l}$, where $N_{i,r}$ denotes the number of neighbor nodes that connect to v_i through relation r, and d_l is the dimension of embeddings in layer l. In order to acquire sufficient expressive power to transform the input feature into higher-level feature $z_i^l \in \mathbb{R}^{d_{l+1}}$, a shared linear transformation weight matrix $W_r^l \in \mathbb{R}^{d_l \times d_{l+1}}$ for relation r is applied:

$$z_i^l = W_r^l h_i^l. \tag{6}$$

Then we calculate the unnormalized attention score e_{ij}^l using the embeddings of v_i and the neighbor node v_j. We further normalize the e_{ij}^l using *softmax* function to make the coefficients easily comparable across different nodes. The e_{ij}^l and the normalized node-level attention vector α_{ij}^l is computed as:

$$\begin{aligned} \alpha_{ij}^l &= \frac{\exp(e_{ij}^l)}{\sum_{k \in N_{i,r}} \exp(e_{ik}^l)}, \\ e_{ij}^l &= \text{LeakyReLU}[W_{ij}^l (z_i^l \| z_j^l)], \end{aligned} \tag{7}$$

where $\|$ indicates concatenation operator, and $W_{ij}^l \in \mathbb{R}^{2d_{l+1}}$ is a weight vector that represents the attention between v_i and v_j. After that, we calculate the node embedding $h_{i,r}^{l+1}$ using α_{ij}^l as coefficients to linearly combine the neighbor nodes features. Because the inner-ingredients transformer captures the interaction among ingredients while the node-level attention considers different impacts of neighbor nodes, we also include the inner-ingredients set embedding $h_{i,trans}$ (Equation 5) for relation 'recipe-ingredient' *(r-i)* to compute the node embeddings at layer $l+1$:

$$h_{i,r}^{l+1} = \begin{cases} FC\left\{cat\left[h_{i,trans}, \sigma\left(\sum_{j \in N_{i,r}} \alpha_{ij}^l z_j^l\right)\right]\right\}, & \text{if r = } \textit{(r-i)} \\ \sigma\left(\sum_{j \in N_{i,r}} \alpha_{ij}^l z_j^l\right), & \text{if r} \neq \textit{(r-i)} \end{cases} \tag{8}$$

where $h_{i,r}^{l+1}$ is the learned embedding of layer $l+1$ for v_i through relation r, *cat* indicates the concatenate operation, and FC denotes a fully connected layer. Instead of performing a single attention function with a hidden layer, we extend the node-level attention to multi-head attention so that the training process is more stable. Specifically, we calculate the node-level attention K times in parallel, and then we concatenate and project it into the final learned relation-specific embedding $h_{i,r}^{l+1}$:

$$h_{i,r}^{l+1} = \begin{cases} FC\left\{cat\left[h_{i,trans}, \Big\|_{k=1}^{K} \sigma\left(\sum_{j \in N_{i,r}} \alpha_{ij}^l z_j^l\right) W_O\right]\right\}, & \text{if r = } \textit{(r-i)} \\ \Big\|_{k=1}^{K} \sigma\left(\sum_{j \in N_{i,r}} \alpha_{ij}^l z_j^l\right) W_O, & \text{if r} \neq \textit{(r-i)} \end{cases} \tag{9}$$

where $W_O \in \mathbb{R}^{Kd_k \times d_{l+1}}$ is a learnable parameter matrix with $d_k = d_{l+1}/K$. Due to the reduced dimension d_k of each head, the total computational cost is similar to that of single-head attention with dimension d_{l+1}.

Relation-level Attention. To learn the importance of each relation, we propose the relation-level attention. Specifically, we first transform the relation-specific node embeddings through a shared non-linear transformation matrix W_R. Then we use a relation-level intermediary vector q to calculate the similarity of transformed embeddings as the importance of each relation-specific node embedding. After that, we average the importance of each relation-specific node embedding under relation r to generate the un-normalized importance of each relation $w_{i,r}$ for node v_i. The process is formulated as follow:

$$w_{i,r} = \frac{1}{|V_r|} \sum_{i \in V_r} q^T \cdot \tanh(W_R \cdot h_{i,r}^{l+1} + b), \tag{10}$$

where $W_R \in \mathbb{R}^{d_{l+1} \times d_{l+1}}$ is the weight matrix, $b \in \mathbb{R}^{d_{l+1}}$ is the bias vector, $q \in \mathbb{R}^{d_{l+1}}$ is the relation-level intermediary vector, and V_r denotes the set of nodes under relation r. To make the coefficients comparable across different relations, we normalize the $w_{i,r}$ to get the relation-level attention vector $\beta_{i,r}$ for each relation r using the *softmax* function:

$$\beta_{i,r} = \frac{\exp(w_{i,r})}{\sum_{r \in R_i} \exp(w_{i,r})}, \tag{11}$$

where R_i denotes the associated relations of v_i. Here the relation-level attention vector $\beta_{i,r}$ can be interpreted as the contribution of relation r to node v_i. Apparently, the higher the $\beta_{i,r}$, the more important the relation r is. Since different tasks have different training objectives, the importance of each relation may contribute differently to the task, and correspondingly the relation-level attention vector for a specific relation r could have different weights. After that, we fuse the relation-level attentions with relation-specific node embeddings to obtain the final node embedding h_i^{l+1} for v_i at layer $l+1$ as follow:

$$h_i^{l+1} = \sum_{r=1}^{R_i} \beta_{i,r} \cdot h_{i,r}^{l+1}. \tag{12}$$

4.4 Objective Function

To optimize the model, we design a novel objective function by combining both supervised node classification loss and self-supervised link prediction loss (Fig. 2 (e)). In particular, we extract recipe tags from food.com as ground-truth labels, and design a cuisine category classification and a region prediction task (Further details about these two tasks can be found in Section 5.1). To be more specific, we

introduce a node classification loss (Cross-Entropy) for the cuisine category classification (or recipe region prediction):

$$L_{node} = -\sum_{i \in \mathcal{Y}_T} Y_i \log(FC(h_i)), \tag{13}$$

where $\mathcal{Y}_T$ is the set of training data, FC denotes the fully connected layer, Y_i is the one-hot label of v_i, and h_i is the learned embedding of v_i (Eq. 12). We also design a self-supervised link prediction loss to improve the optimization process. The link prediction task involves comparing the similarity scores (inner-products of node embeddings) between nodes connected by a *recipe-ingredient* relation against the similarity scores between an arbitrary pair of recipe and ingredient nodes. For example, given an edge connecting a recipe node v_i and an ingredient node v_j, we encourage the score between v_i and v_j to be higher than the score between v_i and a randomly sampled negative ingredient node v'_j. We use a margin loss to formulate the link prediction loss:

$$L_{link} = \sum_{i \in \mathcal{Y}_T, j \in N_{i,ing}} max(0, 1 - h_i h_j + h_i h'_j), \tag{14}$$

where h_i, h_j, h'_j are the embeddings of nodes v_i, v_j, v'_j respectively, and $N_{i,ing}$ indicates the ingredient neighbors of v_i. The final objective function of our model L is defined as the weighted combination of L_{node} and L_{link}:

$$L = L_{node} + \lambda L_{link}. \tag{15}$$

where λ is a trade-off weight for balancing two losses.

5 EXPERIMENTS

In this section, we conduct extensive experiments for two food study tasks (*i.e.*, cuisine category classification and region prediction) to compare performance of different models. We also introduce ablation studies, embedding visualizations, and case studies to demonstrate how *rn2vec* can learn superior recipe representations.

5.1 Experimental Setup

In this work, we randomly choose 70% of the recipes as the training set, 15% as the validation set, and the remaining 15% as the test set. Since the tags crawled from food.com indicate the characteristics of each recipe, we build two new categories (*i.e.,* cuisine type category, and region category) to evaluate our model. The cuisine type category contains nine classes in total, *i.e.*, Appetizers, Beverages, Breads, Soups, Salads, Desserts, Vegetables, Main-dish, and Others. We assign each recipe to only one out of these nine classes (by one-hot encoding). With these labeled data, we design a cuisine category classification task to predict the cuisine category of each recipe. Moreover, we further evaluate our model on recipe region prediction task. The region categories span four classes, namely, American, European, Asian, and Mexican. We also create an additional class, Miscellaneous, for those recipes that do not have the region information. For fair evaluation, we ignore the Miscellaneous class when predicting the region, since the recipes in Miscellaneous could belong to any of these regions. The statistics of each category are shown in Table 2.

Table 2: The statistics of cuisine and region categories, and samples in training, validation, and test sets.

Task	Category	#Recipes	Partition	#Recipes
Cuisine	Others	16,820		
	Appetizer	6,722	Training	54,358
	Beverage	6,010		
	Bread	6,114		
	Soups	1,749	Validation	11,616
	Salads	3,957		
	Desserts	20,209		
	Vegetables	4,435	Test	11,759
	Main-dish	11,717		
	Total			77,733
Region	Miscellaneous	61,714	Training	11,237
	American	8,408		
	European	4,194	Validation	2,340
	Asian	2,237		
	Mexican	1,180	Test	2,442
	Total	77,733		16,019

5.2 Baselines

We compare our model with eight baseline methods, including various recipe representation learning and network embedding models: 1) **Word2Vec** [24], which applies the word2vec model to encode ingredients as input and uses a neural network as the multi-label classifier. 2) **TextCNN** [17], a sentence-level classifier by convolutional neural network with pretrained skip-instruction embeddings [23] as input. 3) **DeepWalk** [32]: A random walk based homogeneous network embedding method. 4) **matapath2vec** [8]: A heterogeneous network embedding method based on meta-path guided random walk. Here we use meta-path *recipe-ingredient-recipe*. 5) **GraphSAGE** [11]: A graph neural network model which generates embeddings by sampling and aggregating features of local neighbors. 6) **GAT** [41]: A graph attention network model which aggregates neighbors information through attention mechanism. 7) **RGCN** [36]: A relational graph convolutional network model which distinguishes neighbors connected by different relations. 8) **Reciptor** [22]: A set transformer-based model optimized by instructions-ingredients similarity and a knowledge graph based triplet loss.

5.3 Implementation Details

For the proposed model *rn2vec*, we set the learning rate to 0.005, the number of node-level attention heads to 8, the number of set transformer attention heads to 2, the hidden size to 128, the input dimension of skip-instruction embeddings to 1024, the input dimension of ingredient embeddings to 46, batch size to 64, training epochs to 100, and the trade-off factor λ to 0.1. We optimize the model with Adam [18] and decay the learning rate exponentially by $\gamma = 0.95$ every epoch. In TextCNN, we follow the default setup using 3 filters with window size (3, 4, 5) and dropout rate 0.5. For random walk based network embedding methods including DeepWalk and metapath2vec, we set window size to 5, walk length to 30, walks rooted at each node to 5, and the number of negative samples

Table 3: Cuisine category classification results.

Metric	Method	Appetizer	Beverage	Bread	Soups	Salads	Desserts	Vegetables	Main-dish	Others	Total
Precision	word2vec [24]	53.2	66.6	65.9	36.2	46.6	65.7	42.5	51.1	38.2	59.9
	TextCNN [17]	40.4	62.7	71.1	28.6	42.2	67.7	57.1	54.0	41.6	58.6
	DeepWalk [32]	39.2	62.8	49.6	18.8	39.0	56.7	34.2	41.0	34.1	50.1
	metapath2vec [8]	46.8	72.8	56.9	34.2	46.5	67.3	30.1	45.6	38.6	57.5
	GraphSAGE [11]	49.6	82.8	67.3	41.9	49.6	77.3	56.9	60.3	42.7	66.5
	GAT [41]	64.7	85.5	72.8	52.8	52.9	78.0	55.7	62.8	51.9	70.2
	RGCN [36]	65.8	87.2	68.7	51.4	62.0	75.6	58.0	63.9	53.4	70.3
	Reciptor [22]	56.1	73.1	67.2	21.2	54.7	75.9	49.3	56.1	52.2	65.4
	rn2vec	**66.9**	**87.3**	**75.3**	**56.4**	**65.0**	**81.0**	**66.4**	**71.4**	**64.4**	**75.0**
Recall	word2vec [24]	33.5	53.8	70.0	7.42	38.8	81.1	17.8	51.6	45.7	56.7
	TextCNN [17]	33.3	78.7	34.9	24.9	54.3	86.2	2.85	77.1	30.3	63.0
	DeepWalk [32]	23.4	48.3	41.4	1.31	22.8	75.2	5.71	59.9	34.3	50.0
	metapath2vec [8]	30.6	64.2	54.2	6.11	32.0	79.6	9.84	62.7	40.5	56.7
	GraphSAGE [11]	40.5	69.1	37.6	47.6	63.4	83.3	36.4	62.9	50.5	63.2
	GAT [41]	43.1	78.4	73.9	16.6	**73.9**	86.1	37.5	71.5	51.2	70.5
	RGCN [36]	40.0	72.9	81.1	33.2	56.7	87.3	48.2	73.5	51.1	71.1
	Reciptor [22]	40.8	70.5	73.3	3.06	56.7	84.6	31.1	79.6	43.4	68.4
	rn2vec	**53.5**	**81.4**	**85.0**	**59.8**	72.8	**87.5**	**60.2**	**82.6**	**54.1**	**78.2**
F1	word2vec [24]	41.1	59.5	66.0	12.3	42.3	72.6	25.1	51.4	41.6	58.2
	TextCNN [17]	36.5	69.8	46.9	26.6	47.5	75.8	5.43	63.5	35.1	60.7
	DeepWalk [32]	29.3	54.6	45.2	2.45	28.7	64.7	9.78	48.7	34.2	50.1
	metapath2vec [8]	37.1	68.2	55.5	10.4	37.9	72.9	14.8	52.8	39.5	57.1
	GraphSAGE [11]	44.6	75.3	48.3	44.6	55.7	80.2	44.4	61.6	46.3	64.8
	GAT [41]	51.7	81.8	73.3	25.3	61.7	81.8	44.8	66.9	51.5	70.3
	RGCN [36]	49.8	79.4	74.4	40.3	59.2	81.1	52.7	68.3	52.2	70.7
	Reciptor [22]	47.2	71.8	70.2	5.34	55.7	80.0	38.1	65.8	47.4	66.9
	rn2vec	**59.4**	**84.2**	**79.9**	**58.1**	**68.7**	**84.1**	**63.1**	**76.6**	**58.8**	**76.6**

Table 4: Region prediction results.

Metric	Method	American	European	Asian	Mexican	Total
Precision	word2vec [24]	66.4	52.0	59.9	53.0	61.3
	TextCNN [17]	59.8	52.3	43.2	17.4	56.3
	DeepWalk [32]	56.4	41.7	36.7	22.2	52.3
	metapath2vec [8]	63.9	54.8	70.1	62.5	62.9
	GraphSAGE [11]	65.7	64.1	51.0	56.1	62.5
	GAT [41]	70.8	56.9	74.3	60.8	66.9
	RGCN [36]	69.3	60.2	78.6	58.7	67.7
	Reciptor [22]	65.4	58.9	66.0	62.2	64.1
	rn2vec	**73.2**	**67.9**	**78.8**	**67.9**	**72.4**
Recall	word2vec [24]	71.7	47.9	61.0	37.2	61.3
	TextCNN [17]	81.4	31.1	39.0	2.09	56.3
	DeepWalk [32]	**82.3**	18.6	30.3	2.09	52.3
	metapath2vec [8]	81.0	38.2	61.0	28.8	62.9
	GraphSAGE [11]	81.4	35.1	72.5	12.0	62.5
	GAT [41]	74.7	58.1	68.7	41.4	66.9
	RGCN [36]	79.9	53.9	67.2	33.5	67.7
	Reciptor [22]	82.2	41.6	58.2	29.3	64.1
	rn2vec	82.1	**58.4**	**77.1**	**47.6**	**72.4**
F1	word2vec [24]	68.9	49.9	60.4	43.7	61.3
	TextCNN [17]	69.0	39.0	41.0	3.74	56.3
	DeepWalk [32]	66.9	25.7	33.2	3.83	52.3
	metapath2vec [8]	71.4	45.0	65.2	39.4	62.9
	GraphSAGE [11]	72.7	45.4	59.9	19.8	62.5
	GAT [41]	72.7	57.5	71.4	49.2	66.9
	RGCN [36]	74.2	56.9	72.5	42.7	67.7
	Reciptor [22]	72.9	48.7	61.8	39.9	64.1
	rn2vec	**77.4**	**62.8**	**77.9**	**56.0**	**72.4**

to 5. For a fair comparison, we set the embedding dimension to 128 for all of the above models except for Reciptor as we follow the original setup and use 600 as the embedding dimension.

5.4 Performance Comparison

We use the Precision, Recall, and F1 score as the evaluation metrics. Micro-average is used for F1 score to avoid the class imbalance issues. The performances of all models on two tasks (cuisine category classification and region prediction) are reported on Table 3 and Table 4 respectively. The best results are highlighted in bold. According to these tables, we can find that our model *rn2vec* outperforms all the baselines for both tasks in most cases. As for the recall value for category Salads, *rn2vec* only has a small difference compared to the best baseline GAT, but still remains the second best among all models. Even GAT performs well in predicting Salads, it fails in predicting Appetizers, Soups, Vegetables, and Main-dish. Considering these five categories are close to each other in the embedding space, GAT is biased to Salads while sacrifices the performance on other categories. However, *rn2vec* is more stable when predicting each category and can generate decent results on all of them. In the region prediction task, *rn2vec* exceeds all baselines for the recall score except a 0.2% difference for category American compared to the best baseline DeepWalk. However, DeepWalk has the lowest recall score among all baselines in predicting other categories, especially when the category has small quantity. In other words, DeepWalk performs poorly when the labels are imbalanced. However, our model *rn2vec* leverages several neural network modules and performs well in this imbalanced situation. In general, *rn2vec* improves the precision score by **+4.7%**, recall score by **+7.1%**, and F1 score by **+5.9%**, compared with the best baseline in cuisine category classification task, and improves all the scores by **+4.7%** in region

Table 5: Model variant comparison: (1) NA - node-level attention? (2) RA - relation-level attention? (3) IT - inner-ingredients transformer? (4) NC - node classification loss? (5) LP - link prediction loss?

Component	$rn2vec_{NA}$	$rn2vec_{RA}$	$rn2vec_{IT}$	$rn2vec$
NA	✔	✔	✔	✔
RA		✔	✔	✔
IT			✔	✔
NC	✔	✔	✔	✔
LP				✔

Table 6: F1 scores of different model variants.

Task	Category	$rn2vec_{NA}$	$rn2vec_{RA}$	$rn2vec_{IT}$	$rn2vec$
Cuisine category classification	Appetizer	54.2	58.1	58.5	**59.4**
	Beverage	79.2	83.0	83.8	**84.2**
	Bread	75.1	78.2	79.6	**79.9**
	Soups	39.0	52.3	55.9	**58.1**
	Salads	61.4	64.9	67.0	**68.7**
	Desserts	82.8	83.9	83.8	**84.1**
	Vegetables	60.7	61.5	**64.0**	63.1
	Main-dish	74.0	75.3	75.9	**76.6**
	Others	57.0	56.5	58.0	**58.8**
	Total	73.5	75.5	76.1	**76.6**
Region prediction	American	75.7	74.9	76.3	**77.4**
	European	56.4	62.0	62.6	**62.8**
	Asian	74.9	**78.1**	76.6	77.9
	Mexican	55.8	**58.8**	58.7	56.0
	Total	69.7	70.8	71.6	**72.4**

prediction task. This demonstrates that *rn2vec* can obtain better recipe embeddings compared to the other models.

5.5 Ablation Studies

rn2vec is a joint learning framework composed of several neural network modules and optimized by two losses. How do different components impact the model performance? To answer this question, we conduct ablation studies to evaluate the performances of several model variants including: (a) $\boldsymbol{rn2vec_{NA}}$ that only uses node-level attention and node classification loss, while the importance of each relation is assigned equally; (b) $\boldsymbol{rn2vec_{RA}}$ that uses node-level attention, relation-level attention, and node classification loss; (c) $\boldsymbol{rn2vec_{IT}}$ that uses all modules but only optimized through node classification loss. Further details about the difference between these variants can be found in Table 5. We report the performances (in terms of F1 score) of these model variants for both tasks in Table 6. From this table, we can find that the performance increases when we incorporate more modules and optimize the model with both losses. This demonstrates the efficiency of node-level attention, relation-level attention, inner-ingredients transformer, and both loss functions. Among the variants, *rn2vec* achieves best results in most cases, despite the imbalance of each category. For those categories where *rn2vec* does not obtain the best result, the performance drop is small. This indicates the strong capability of different components in our model.

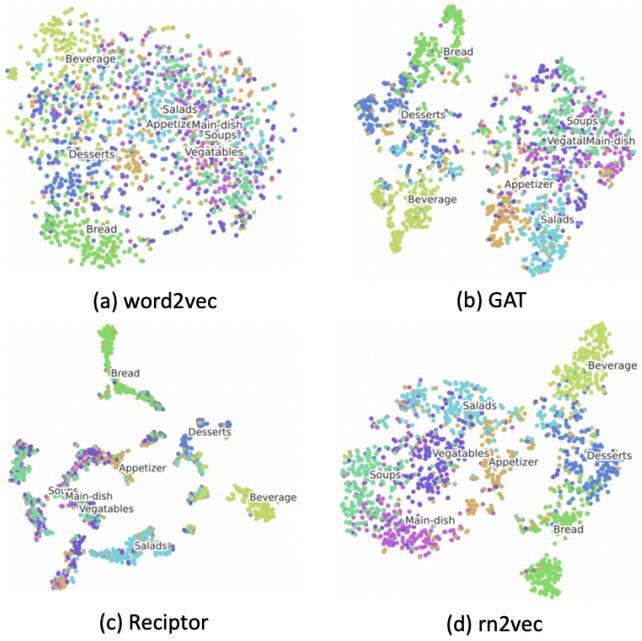

Figure 3: Visualization of recipe embeddings generated by different models. Each point indicates one recipe and its color indicates the cuisine category.

5.6 Embedding Visualization

For a more intuitive understanding and comparison, we generate visualizations of recipe embeddings using t-SNE [39]. As shown in Fig. 3, we can find that Word2vec does not perform well. Only Beverage, Desserts, and Bread are separated apart while other categories are mixed together. GAT can successfully distinguish Bread, Desserts, Beverage, Salads, and some part of the Appetizers, while the difference is not clear and the remaining categories are mixed with each other. Reciptor can separate Beverage, Desserts, Bread, and part of Salads well, but fails to distinguish others. However, our model *rn2vec* can clearly identify each category. In particular, for categories Soups, Main-dish, and Vegetables, *rn2vec* can clearly separate their embeddings, while the other models have them mixed together. This again demonstrates that *rn2vec* can learn discriminative recipe embeddings.

5.7 Case Study

To show different model performance with concrete examples, we analyze some misclassified cases in the cuisine category classification task, as shown in Table 7. Specifically, Word2Vec misclassifies all of these four recipes. One potential reason is that ingredients always appear in numerous cuisines, and Word2Vec fails to encode the recipe embeddings comprehensively by only using the ingredients. GAT successfully classifies the recipes *Molasses Raisin Bread* and *Yogurt Smoothie*, but fail to classifies *Watercress Canapes* and *Savory Scones*. This is because the category Appetizer is close to Salads, and category Bread is close to Desserts in the latent space, while GAT is not able to capture the difference between them if not considering the heterogeneity of nodes. Reciptor successfully

Table 7: Error cases for cuisine category classification.

Recipe Title	Ingredients	Ground Truth	word2vec [24]	GAT [41]	Reciptor [22]	*rn2vec*
Molasses Raisin Bread	active dry yeast, molasses, vanilla margarine, cinnamon, flour, raisins	Bread	Desserts	Bread	Bread	Bread
Yogurt Smoothie	low-fat yogurt, cool whip free, fresh strawberries, ice cubes	Beverage	Desserts	Beverage	Desserts	Beverage
Watercress Canapes	low fat cottage cheese, tarragon, dill fresh chives, watercress, crackers	Appetizer	Main-dish	Salads	Salads	Appetizer
Savory Scones	pork sausage, onions, buttermilk baking mix, milk, instant onion, dried oregano	Bread	Main-dish	Desserts	Main-dish	Bread

classifies the recipe *Molasses Raisin Bread*, but fails to classifies the other three recipes. This may be because the triplet loss of Reciptor only considers each recipe once in all triplets sampled from FoodKG, which significantly prevents the model from capturing precise structural information and various relationship among recipes. However, our model *rn2vec* leverages both textual and structural information with several neural network modules, which can clearly distinguish the difference between categories and thus correctly classify all of these recipes.

6 RELATED WORK

In this section, we review existing work including recipe datasets and food graphs, recipe representation learning, and network embedding approaches.

Recipe Datasets and Food Graphs. Existing recipe datasets only focus on the recipe text or recipe images [2, 20, 23, 29, 35], while ignoring the structural relationship between recipes. For example, Food-101 [2] is a dataset of 101 food categories with 101'000 images. Recipe1M [23, 35] is a dataset containing over one million cooking recipes and 13 million food images. Several others tried to formulate the relationship between foods, while recipe information is not included. For example, the ingredient network [37] contained the co-occurrence frequency or substitution information between ingredients. FlavorGraph [30] showed the relationship between foods and chemical compounds. On the other hand, FoodKG [12] contained recipes and a food ontology as a knowledge graph but specifically designed for recipe recommendation and information retrieval. Therefore, although these datasets established a large step towards learning rich recipe representations, they are still limited in either generality, integrity, or feasibility.

Recipe Representation Learning. Recent breakthroughs in deep learning [14] have further fueled the interest in food studies for their superiority in learning representations. However, most existing work [4, 10, 23, 35, 42] focused on the cross-modal embeddings through image-recipe retrieval task. For example, Salvador *et. al.*. [35] proposed a joint training framework using semantic regularization, while several related studies applied adversarial networks [42], used attention and self-attention mechanisms [5, 10], leveraged cuisine attributes [13, 27], and introduced a double-triplet learning scheme [4]. These models aim to align food images and recipe text in a joint embedding space with closest embedding distance as the objective function. On the other hand, Reciptor [22] proposed a set transformer-based joint model to learn recipe representations from only the textual level, by aligning the ingredients with the instructions in a latent space and using a knowledge graph based triplet loss.

Network Embedding. Network embedding [7] has became one of the most popular data mining topics in the past few years. Many network embedding models [8, 11, 19, 32, 36, 41, 43, 44] were proposed to learn vectorized node embeddings that can be used in various network mining tasks. For example, DeepWalk [32] obtained node embeddings by feeding a set of random walks over network to SkipGram model [24]. Nodes in the network were trained on each walk where the neighbor nodes served as the contextual information. matapath2vec [8] performed meta-path based random walks and utilized SkipGram method to embed the heterogeneous networks. GAT [41] incorporated attention mechanism to learn node embeddings by sampling and aggregating features from neighbors with different weights. RGCN [36] embeded nodes by aggregation and keeps a distinct linear projection weight for each edge type. In our work, we apply the essence in network embedding and solve the recipe representation learning with networks.

7 CONCLUSIONS

In this paper, we propose and formalize the problem of *recipe representation learning with networks*. To solve this problem, we create *RecipeNet*, a new and large-scale corpus of recipe data to facilitate network based food studies and recipe representation learning research. Furthermore, we develop *rn2vec*, a novel recipe network embedding model, to learn recipe representations. *rn2vec* is able to capture textual, structural, and nutritional information through several neural network modules. We also design a novel objective function combined of node classification and link prediction to jointly optimize the model. The extensive experiments show that *rn2vec* outperforms state-of-the-art baselines for both cuisine category classification and region prediction tasks. In the future, we plan to incorporate more information into *RecipeNet* and to improve *rn2vec*. We observe that there are still plenty of useful information components that we can use such us individual and food flavor information. One promising direction is to integrate human's ratings and comments on recipes into *RecipeNet* and *rn2vec*. In addition, more interesting tasks could be investigated such us recipe recommendation and food suggestion.

ACKNOWLEDGMENTS

This work is supported by the Agriculture and Food Research Initiative grant no. 2021-67022-33447/project accession no.1024822 from the USDA National Institute of Food and Agriculture.

REFERENCES

[1] Semih agcioglu, Aykut Erdem, Erkut Erdem, and Nazli Ikizler-Cinbis. 2018. RecipeQA: A Challenge Dataset for Multimodal Comprehension of Cooking Recipes. In *EMNLP*.

[2] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. 2014. Food-101 – Mining Discriminative Components with Random Forests. In *ECCV*.

[3] G. Bouma. 2009. Normalized (pointwise) mutual information in collocation extraction. In *GSCL*.

[4] Micael Carvalho, Rémi Cadène, David Picard, Laure Soulier, Nicolas Thome, and Matthieu Cord. 2018. Cross-Modal Retrieval in the Cooking Context: Learning Semantic Text-Image Embeddings. In *SIGIR*.

[5] Jing-Jing Chen, Chong-Wah Ngo, Fu-Li Feng, and Tat-Seng Chua. 2018. Deep Understanding of Cooking Procedure for Cross-modal Recipe Retrieval. In *ACM-MM*.

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

[7] Peng Cui, Xiao Wang, Jian Pei, and Wenwu Zhu. 2019. A Survey on Network Embedding. *IEEE Transactions on Knowledge and Data Engineering* (2019).

[8] Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. 2017. metapath2vec: Scalable Representation Learning for Heterogeneous Networks. In *KDD*.

[9] Damion Dooley, Emma Griffiths, Gurinder Gosal, Pier Luigi Buttigieg, Robert Hoehndorf, Matthew Lange, Lynn Schriml, Fiona Brinkman, and William Hsiao. 2018. FoodOn: a harmonized food ontology to increase global food traceability, quality control and data integration. *NPJ Science of Food* (2018).

[10] Matthias Fontanellaz, Stergios Christodoulidis, and Stavroula G. Mougiakakou. 2019. Self-Attention and Ingredient-Attention Based Model for Recipe Retrieval from Image Queries. In *MADiMa*.

[11] William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *NeurIPS*.

[12] Steven Haussmann, Oshani Seneviratne, Yu Chen, Yarden Ne'eman, James Codella, Ching-Hua Chen, Deborah L. McGuinness, and Mohammed J. Zaki. 2019. FoodKG: A Semantics-Driven Knowledge Graph for Food Recommendation. In *ISWC*.

[13] Luis Herranz, Weiqing Min, and Shuqiang Jiang. 2018. Food recognition and recipe analysis: integrating visual content, context and external knowledge. *arXiv preprint arXiv: 1801.07239* (2018).

[14] M. I. Jordan and T. M. Mitchell. 2015. Machine learning: Trends, perspectives, and prospects. *American Association for the Advancement of Science* (2015).

[15] S. Kalajdziski, G. Radevski, I. Ivanoska, K. Trivodaliev, and B. Risteska Stojkoska. 2018. Cuisine classification using recipe's ingredients. In *MIPRO*.

[16] Abdullah Khilji, Riyanka Manna, Sahinur Laskar, Dr. Partha Pakray, Dipankar Das, Sivaji Bandyopadhyay, and Alexander Gelbukh. 2021. CookingQA: Answering Questions and Recommending Recipes Based on Ingredients. *Arabian Journal for Science and Engineering* (2021).

[17] Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *EMNLP*.

[18] Diederik Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. In *ICLR*.

[19] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.

[20] Tomasz Kusmierczyk, Christoph Trattner, and Kjetil Norvag. 2016. Understanding and Predicting Online Food Recipe Production Patterns. In *ACM-HT*.

[21] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam R. Kosiorek, Seungjin Choi, and Yee Whye Teh. 2019. Set Transformer: A Framework for Attention-based Permutation-Invariant Neural Networks. In *ICML*.

[22] Diya Li and Mohammed J. Zaki. 2020. RECIPTOR: An Effective Pretrained Model for Recipe Representation Learning. In *KDD*.

[23] Javier Marin, Aritro Biswas, Ferda Ofli, Nicholas Hynes, Amaia Salvador, Yusuf Aytar, Ingmar Weber, and Antonio Torralba. 2019. Recipe1M+: A Dataset for Learning Cross-Modal Embeddings for Cooking Recipes and Food Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2019).

[24] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. In *ICLR*.

[25] Weiqing Min, Bing-Kun Bao, Shuhuan Mei, Yaohui Zhu, Yong Rui, and Shuqiang Jiang. 2018. You Are What You Eat: Exploring Rich Recipe Information for Cross-Region Food Analysis. *IEEE Transactions on Multimedia* (2018).

[26] Weiqing Min, Shuqiang Jiang, Linhu Liu, Yong Rui, and Ramesh Jain. 2019. A Survey on Food Computing. *arXiv preprint arXiv:1808.07202* (2019).

[27] Weiqing Min, Shuqiang Jiang, Shuhui Wang, Jitao Sang, and Shuhuan Mei. 2017. A Delicious Recipe Analysis Framework for Exploring Multi-Modal Recipes with Various Attributes. In *ACM-MM*.

[28] Ole G. Mouritsen, Rachel Edwards-Stuart, Yong-Yeol Ahn, and Sebastian E. Ahnert. 2017. Data-driven Methods for the Study of Food Perception, Preparation, Consumption, and Culture. *Frontiers in ICT* (2017).

[29] A. Myers, N. Johnston, V. Rathod, A. Korattikara, A. Gorban, N. Silberman, S. Guadarrama, G. Papandreou, J. Huang, and K. Murphy. 2015. Im2Calories: Towards an Automated Mobile Vision Food Diary. In *ICCV*.

[30] Donghyeon Park, Keonwoo Kim, Seoyoon Kim, and Michael Spranger. 2021. FlavorGraph: a large-scale food-chemical graph for generating food representations and recommending food pairings. *Scientific Reports* (2021).

[31] Donghyeon Park, Keonwoo Kim, Yonggyu Park, Jungwoon Shin, and Jaewoo Kang. 2019. KitcheNette: Predicting and Ranking Food Ingredient Pairings using Siamese Neural Network. In *IJCAI*.

[32] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: Online Learning of Social Representations. In *KDD*.

[33] M. Rokicki, C. Trattner, and E. Herder. 2018. The Impact of Recipe Features, Social Cues and Demographics on Estimating the Healthiness of Online Recipes. In *ICWSM*.

[34] Sina Sajadmanesh, Sina Jafarzadeh, Seyed Ali Ossia, Hamid R. Rabiee, Hamed Haddadi, Yelena Mejova, Mirco Musolesi, Emiliano De Cristofaro, and Gianluca Stringhini. 2017. Kissing Cuisines: Exploring Worldwide Culinary Habits on the Web. In *WWW*.

[35] Amaia Salvador, Nicholas Hynes, Yusuf Aytar, Javier Marin, Ferda Ofli, Ingmar Weber, and Antonio Torralba. 2017. Learning Cross-modal Embeddings for Cooking Recipes and Food Images. In *CVPR*.

[36] Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling Relational Data with Graph Convolutional Networks. In *ESWC*.

[37] ChunYuen Teng, Yu-Ru Lin, and Lada A. Adamic. 2012. Recipe recommendation using ingredient networks. In *WebSci*.

[38] Agricultural Research Service. U.S. Department of Agriculture. 2019. USDA National Nutrient Database for Standard Reference, Release 27. Methods and Application of Food Composition Laboratory Home Page, http://www.ars.usda.gov/nea/bhnrc/mafcl. (August 2019).

[39] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing Data using t-SNE. *Journal of Machine Learning Research* (2008).

[40] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. In *NeurIPS*.

[41] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR*.

[42] Hao Wang, Doyen Sahoo, Chenghao Liu, Ee-Peng Lim, and Steven C. H. Hoi. 2019. Learning Cross-Modal Embeddings with Adversarial Networks for Cooking Recipes and Food Images. In *CVPR*.

[43] Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V Chawla. 2019. Heterogeneous graph neural network. In *KDD*. 793–803.

[44] Chuxu Zhang, Ananthram Swami, and Nitesh V Chawla. 2019. Shne: Representation learning for semantic-associated heterogeneous networks. In *WSDM*. 690–698.