RecipeRec: A Heterogeneous Graph Learning Model for Recipe Recommendation

Yijun Tian[1], **Chuxu Zhang**[2], **Zhichun Guo**[1], **Chao Huang**[3], **Ronald Metoyer**[1], **Nitesh V. Chawla**[1]
[1]Department of Computer Science, University of Notre Dame, USA
[2]Department of Computer Science, Brandeis University, USA
[3]Department of Computer Science, University of Hong Kong, Hong Kong
[1]{yijun.tian, zguo5, rmetoyer, nchawla}@nd.edu, [2]chuxuzhang@brandeis.edu,
[3]chaohuang75@gmail.com

Abstract

Recipe recommendation systems play an essential role in helping people decide what to eat. Existing recipe recommendation systems typically focused on content-based or collaborative filtering approaches, ignoring the higher-order collaborative signal such as relational structure information among users, recipes and food items. In this paper, we formalize the problem of *recipe recommendation with graphs* to incorporate the collaborative signal into recipe recommendation through graph modeling. In particular, we first present *URI-Graph*, a new and large-scale user-recipe-ingredient graph. We then propose *RecipeRec*, a novel heterogeneous graph learning model for recipe recommendation. The proposed model can capture recipe content and collaborative signal through a heterogeneous graph neural network with hierarchical attention and an ingredient set transformer. We also introduce a graph contrastive augmentation strategy to extract informative graph knowledge in a self-supervised manner. Finally, we design a joint objective function of recommendation and contrastive learning to optimize the model. Extensive experiments demonstrate that *RecipeRec* outperforms state-of-the-art methods for recipe recommendation. Dataset and codes are available at https://github.com/meettyj/RecipeRec.

1 Introduction

Large-scale food data offers rich knowledge about the food landscape and can help address many important issues of today's society. Recipe websites are a popular source of food inspiration and enable users to rate and comment on recipes as well as receive recipe recommendation [Tian *et al.*, 2022]. In particular, *www.food.com*, one of the largest recipe-sharing websites in the world, consists of over half a million recipes – reflecting the popularity of online recipe portals and the great demand for recipe recommendation services. However, digging into this overwhelming amount of online recipe resources to find a satisfying recipe is very difficult [Britto *et al.*, 2020]. Thus, recipe recommendation

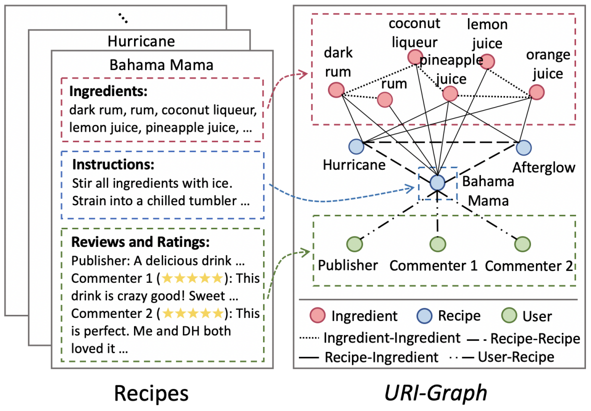

Figure 1: Illustration of *URI-Graph*. Recipe examples are shown on the left. The *URI-Graph* includes three types of nodes and four types of relations that connect these nodes.

systems are proposed to help users identify recipes that align with users' preferences and needs among numerous online resources [Khan *et al.*, 2019].

Existing recipe recommendation approaches typically rely on computing similarities between recipes [Chen *et al.*, 2020]. A few approaches attempt to take user information into account [Khan *et al.*, 2019; Gao *et al.*, 2020], but they only define similar users based on the overlapping rated recipes between users. This overlooks the higher-order collaborative signal (e.g., relational structure information) [Wang *et al.*, 2019a] present among ingredients, recipes, and users. However, individual preference for foods is complex. A user may decide to try a new recipe, for example, because of its ingredients, its cuisine type, or perhaps based on a friend's recommendation. Thus, a comprehensive recipe recommendation system should consider all of these factors. In particular, it is important to encode the higher-order collaborative signal into recipe recommendation and model the relationship among ingredients, recipes and users.

Therefore, we propose the problem of *recipe recommendation with graphs*, which leverages the advances of graph modeling to address recipe recommendation. With the higher-order collaborative signal encoded by a graph learning model, the recommendation system can successfully capture the hidden collaborative filtering effect. Due to the complex relationship among ingredients, recipes and users (e.g., complementary ingredients, similar recipes), solving recipe recommendation with graphs requires an extensive and comprehensive recipe graph. Hence, it should not be surprising that the lack of research on this topic could be due to the lack of such a dataset. With the aim of making better recipe recommendation, we create *URI-Graph*, a new and large-scale user-recipe-ingredient graph for recipe recommendation. As shown in Fig.1, we extract ingredients, instructions, and user interactions of each recipe to build *URI-Graph*, which consists of three types of nodes and four types of relations.

Based on the created *URI-Graph*, we propose *RecipeRec*, a novel heterogeneous graph learning model for recipe recommendation. In particular, we first design a heterogeneous GNN to capture both recipe content and the relational structure information in *URI-Graph*. To fully encode this information, we propose an adaptive node-level attention module to distinguish the subtle difference of neighboring 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. Then, we apply a set transformer [Lee *et al.*, 2019] to encode the ingredient information and model the pairwise interactions and higher-order interactions among ingredients. Next, we introduce a graph contrastive augmentation strategy to extract informative graph knowledge in a self-supervised manner. Finally, we design a novel combined objective function of recommendation and graph contrastive learning to optimize the model. To summarize, our major contributions in this paper are as follows:

- To the best of our knowledge, this is the first attempt to study *recipe recommendation with graphs*. We create *URI-Graph*, a new and large-scale recipe graph dataset, that can facilitate recipe recommendation research and graph-based food studies.
- We propose *RecipeRec*, a novel heterogeneous graph learning model for recipe recommendation. *RecipeRec* can capture recipe content and relational structure information through various modules, including GNN with hierarchical attention, ingredient set transformer and graph contrastive augmentation. We also design a novel objective function to optimize the model.
- We conduct extensive experiments to evaluate the performance of our model. The results demonstrate the superiority of *RecipeRec* by comparing it with state-of-the-art baselines for recipe recommendation.

2 Preliminary

In this section, we describe the concept of user-recipe-ingredient graph, and formally define the problem of *recipe recommendation with graphs*. We further introduce the *URI-Graph* data that we create and use in this paper.

Definition 2.1. User-Recipe-Ingredient Graph. *A user-recipe-ingredient graph is defined as a heterogeneous graph $\mathcal{G} = (V, E, C)$ with three types of nodes V (i.e., user, recipe, ingredient) and four types of edges E (i.e., ingredient-ingredient, recipe-ingredient, recipe-recipe, and user-recipe relations). In addition, nodes are associated with contents C, (e.g., instructions, nutrients, or ratings).*

Problem 1. Recipe Recommendation with Graphs. *Given a user-recipe-ingredient graph $\mathcal{G} = (V, E, C)$, the task is to design a machine learning model $\mathcal{F}_\Theta$ with parameters Θ to recommend recipes to users. In particular, the task is to return a ranked list of recipes for each user, such that its top ranked recipes are true favorite recipes of users. The recommender is supposed to take advantage of the associated recipe contents C and relational structure information in $\mathcal{G}$.*

URI-Graph Data. To solve the problem, we first build *URI-Graph*, as shown in Fig.1. We start by collecting recipes from Reciptor [Li and Zaki, 2020] and crawl the user ratings for each recipe from *food.com* as the user-recipe interactions. We then match each ingredient to the USDA nutritional database [USDA, 2019] to get the nutritional information. Next, we build *URI-Graph* by representing the users, recipes and ingredients as nodes, and constructing four types of relations to connect them. Specifically, we first connect each recipe and its ingredients with an edge, denoting the recipe-ingredient relation, while the weight of each ingredient is used as the edge weight. We then connect recipe nodes by the similarity determined by FoodKG [Haussmann *et al.*, 2019] and the score is used as the edge weight, as shown in Reciptor [Li and Zaki, 2020]. We further connect ingredient nodes by the co-occurring probabilities using Normalized Pointwise Mutual Information [Tian *et al.*, 2021]. We also connect users and recipes based on the interactions, with the rating scores as the edge weight. In the end, we build the graph with 7,958 user nodes, 68,794 recipe nodes and 8,847 ingredient nodes. Moreover, the graph contains four types of edges, including 135,353 user-recipe, 647,146 recipe-recipe, 463,485 recipe-ingredient and 146,188 ingredient-ingredient relations.

3 RecipeRec

To address recipe recommendation over the graph, we develop *RecipeRec*, a heterogeneous graph learning model for recipe recommendation. As illustrated in Fig.2, *RecipeRec* contains a heterogeneous GNN with hierarchical attention and an ingredient set transformer, with a graph contrastive augmentation strategy and a joint objective function.

3.1 GNN with Hierarchical Attention

We first develop a heterogeneous GNN with hierarchical attention to obtain effective user and recipe embeddings. Specifically, for a node $v_i \in \mathcal{G}$, we use a type-specific input projection to transform the input feature, and leverage the adaptive node-level attention module to obtain a relation-specific embedding $h_{i,r}$ for each relation r that connects to v_i. Then, we propose the relation-level attention module to fuse all relation-specific embeddings and obtain the final GNN-based node embedding h_i^G.

Type-specific Input Projection. For the input features of each node type, we use nutrient vectors to represent the ingredients (x_{ing}), the average of the pretrained skip-instruction embeddings [Marin *et al.*, 2019] to represent the

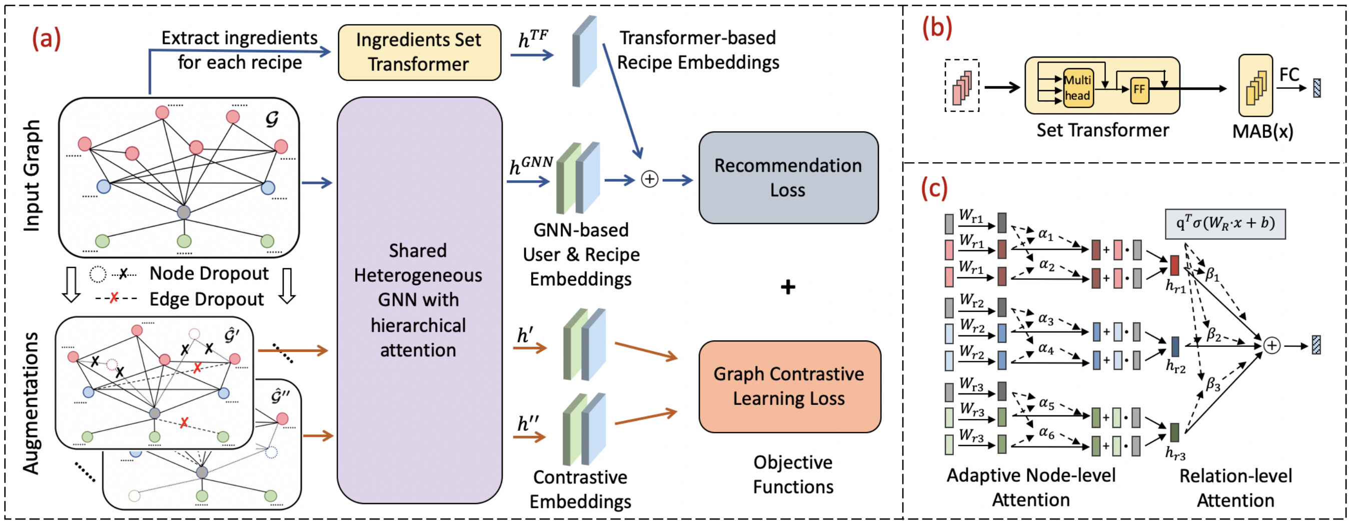
(a)
Extract ingredients for each recipe
Ingredients Set Transformer
h^{TF}
Transformer-based Recipe Embeddings
Input Graph
$\mathcal{G}$
Shared Heterogeneous GNN with hierarchical attention
h^{GNN}
GNN-based User & Recipe Embeddings
Recommendation Loss
Node Dropout
Edge Dropout
Augmentations
$\hat{\mathcal{G}}'$
$\hat{\mathcal{G}}''$
h'
h''
Contrastive Embeddings
Graph Contrastive Learning Loss
Objective Functions
(b)
Multi head
FF
Set Transformer
MAB(x)
FC
(c)
$q^T\sigma(W_R \cdot x + b)$
Adaptive Node-level Attention
Relation-level Attention

Figure 2: (a) The overall framework of our model: we first propose a heterogeneous GNN with hierarchical attention and an ingredient set transformer to learn user and recipe embeddings, which are later fed into the recipe recommendation loss. We then introduce a graph contrastive augmentation strategy to extract informative graph knowledge in a self-supervised manner. Next, the joint loss of recipe recommendation and graph contrastive learning is used to optimize the model; (b) Ingredient set transformer: encoding the interactions among ingredients with permutation invariant preserved; (c) Heterogeneous GNN with hierarchical attention: encoding the nodes of the same type through adaptive node-level attention and then fusing the embeddings over different relations through relation-level attention.

recipes (x_{ins}) and random initialized vectors to represent the users (x_{user}). Due to the heterogeneity of the associated information, given a node v_i of type ϕ_i, we introduce a type-specific input projection $W_{\phi_i} \in \mathbb{R}^{d_{\phi_i} \times d}$ to project the input features into a shared embedding space:

$$x_i = \begin{cases} x_{ing}, & \text{if } \phi_i = \text{ingredient} \\ x_{ins}, & \text{if } \phi_i = \text{recipe} \\ x_{user}, & \text{if } \phi_i = \text{user} \end{cases} \tag{1}$$
$$h_i = W_{\phi_i} \cdot x_i,$$

where x_i is the input feature of v_i with dimension d_{ϕ_i}, and h_i is the projected feature of v_i with dimension d.

Adaptive Node-level Attention. To compute the relation-specific embedding $h_{i,r}^{l+1}$ for v_i in layer $l+1$ of GNN, we first apply a shared weight matrix $W_r^l \in \mathbb{R}^{d_l \times d_{l+1}}$ to transform the input features from layer l, where d_l and d_{l+1} are the dimensions of embeddings in layer l, $l+1$ respectively:

$$z_i^l = W_r^l \cdot h_i^l, \tag{2}$$

where $z_i^l \in \mathbb{R}^{d_{l+1}}$ is the intermediary embedding of v_i. We then calculate the unnormalized attention score e_{ij}^l between v_i and v_j, and normalize the e_{ij}^l using $softmax$ function:

$$e_{ij}^l = \text{LeakyReLU}\left[W_{ij}^l \cdot (z_i^l \| z_j^l)\right],$$
$$\alpha_{ij}^l = \frac{\exp(e_{ij}^l)}{\sum_{k \in N_{i,r}} \exp(e_{ik}^l)}, \tag{3}$$

where $\|$ is concatenation, $W_{ij}^l \in \mathbb{R}^{2d_{l+1} \times d_{l+1}}$ is a weight vector, $N_{i,r}$ denotes the index set of neighboring nodes that connect to v_i through relation r and α_{ij}^l is the attention weight between v_i and v_j. We use α_{ij}^l as coefficients to linearly combine the neighboring nodes features. We further make it adaptive to recommendation task by incorporating the interaction between v_i and v_j so that the interaction between users and recipes can be formulated attentively [Wang *et al.*, 2019a]. We also extend it to multi-head attention to make the training process stable. The process is formulated as follows:

$$h_{i,r}^{l+1} = \Big\|_{m=1}^{M} \text{ReLU}\left(\sum_{j \in N_{i,r}} \alpha_{ij}^l \cdot z_j^l + W_h^l \cdot (h_i^l \odot h_j^l)\right) \cdot W_a, \tag{4}$$

where $h_{i,r}^{l+1}$ is the encoded embedding of layer $l+1$ for v_i through relation r, $\odot$ denotes the element-wise product, $W_h^l \in \mathbb{R}^{d_l \times d_{l+1}}$ and $W_a \in \mathbb{R}^{Md_m \times d_{l+1}}$ are trainable weights, M is the number of attention heads, and d_m is the dimension of attention heads which satisfies $d_m = d_{l+1}/M$.

Relation-level Attention. We further introduce the relation-level attention to learn the importance of each relation and fuse all relation-specific node embeddings. In particular, we first apply a shared non-linear weight matrix $W_R \in \mathbb{R}^{d_{l+1} \times d_{l+1}}$ to transform the relation-specific node embeddings. Then we use a vector $q \in \mathbb{R}^{d_{l+1}}$ and calculate the similarities and average it for all node embeddings of relation r to obtain the importance score $w_{i,r}$ for node v_i:

$$w_{i,r} = \frac{1}{|V_r|} \sum_{i \in V_r} q^{\mathrm{T}} \cdot \tanh(W_R \cdot h_{i,r}^{l+1} + b), \tag{5}$$

where V_r denotes the set of nodes that are correlated with relation r and $b \in \mathbb{R}^{d_{l+1}}$ is the bias vector. We then normalize the $w_{i,r}$ to get the final relation-level attention weight $\beta_{i,r}$:

$$\beta_{i,r} = \frac{\exp(w_{i,r})}{\sum_{r \in R_i} \exp(w_{i,r})}, \tag{6}$$

where R_i indicates the associated relations of v_i and $\beta_{i,r}$ can be explained as the contribution of relation r to v_i. After that, we fuse the relation-specific node embeddings $h_{i,r}^{l+1}$ with $\beta_{i,r}$ to obtain the node embedding h_i^{l+1}:

$$h_i^{l+1} = \sum_{r=1}^{R_i} \beta_{i,r} \cdot h_{i,r}^{l+1}. \tag{7}$$

We denote the embedding obtained in the last propagation layer as the final GNN-based node embedding h_i^G such that $h_i^G = h_i^L$, where L is the number of layers.

3.2 Ingredient Set Transformer

To better encode the ingredients in a recipe, we propose an ingredient set transformer to incorporate the ingredient interactions. Specifically, given a recipe with *n* ingredients, we have a matrix $Q \in \mathbb{R}^{n\times d}$ of *n* query vectors with dimension d. We define an attention function Att maps Q to outputs using p key-value pairs with $K \in \mathbb{R}^{p\times d}, V \in \mathbb{R}^{p\times d}$:

$$Att(Q, K, V) = \omega\left(QK^\top\right) V, \tag{8}$$

where ω is the scaled activation function (we use $softmax(\cdot/\sqrt{d})$ in our model) and $Att(Q, K, V)$ is the weighted sum of V. We further incorporate the multi-head attention by projecting Q, K, V into different subspaces. Considering two ingredient sets $X_1, X_2 \in \mathbb{R}^{n\times d}$, the multi-head attention for X_1 on X_2 is defined as:

$$Att(X_1, X_2) = \omega[(X_1 \cdot W_Q)(X_2 \cdot W_K)^\top](X_2 \cdot W_V), \tag{9}$$

where $W_Q, W_K, W_V \in \mathbb{R}^{d\times d_o}$ are the trainable weights and d_o is the output dimension. Then we feed this self-attention into a Multi-head Attention Block (MAB) and a Set Attention Block (SAB) [Lee *et al.*, 2019] to get the transformer-based recipe embeddings h_i^{TF} of recipe v_i:

$$\begin{aligned} MAB(X_1, X_2) &= FFN[W_m \cdot X_1 + Att(X_1, X_2)], \\ SAB(X) &= MAB(X, X), \\ h_i^{TF} &= FFN[SAB(X)], \end{aligned} \tag{10}$$

where FFN is a feed-forward neural network and $W_m \in \mathbb{R}^{d\times d_o}$ is a trainable weight. We then combine the generated h_i^{TF} with the GNN-based learned embedding h_i^G to obtain the final embedding for each recipe node, denote as $h_i = W_O \cdot (h_i^G + h_i^{TF})$, where $W_O \in \mathbb{R}^{d_o\times d_o}$ is a weight for output projection. For each user node, we have $h_i = h_i^G$.

3.3 Objective Function for Recommendation

We employ a ranking-based objective function to train the model. In particular, the objective involves comparing the preference scores between nodes connected by a *user-recipe* relation against the scores between an arbitrary pair of user and recipe nodes. For example, given a connected user node v_i and a recipe node v_j, we encourage the score $s_{i,j}$ between v_i and v_j to be higher than the score $s_{i,j'}$ between v_i and a random negative recipe node v_j'. Accordingly, we use a predictor (e.g., inner product) to calculate the score and formulate the recommendation loss L_{rec} as follows:

$$\begin{aligned} s_{i,j} &= score\_predictor(h_i, h_j), \\ L_{rec} &= \sum_{i\in\mathcal{U}, j\in N_i} max(0, 1 - s_{i,j} + s_{i,j'}), \end{aligned} \tag{11}$$

where h_i, h_j, h_j' are the embeddings of nodes v_i, v_j, v_j' respectively, $\mathcal{U}$ is the set of users, and N_i indicates the recipe neighbors of v_i.

3.4 Graph Contrastive Augmentation

Since the above objective function only considers explicit and supervised information in the *URI-Graph*, we further introduce a graph contrastive augmentation strategy to capture graph knowledge in a self-supervised manner. In particular, the input graph $\mathcal{G}$ will be augmented with two correlated views $\hat{\mathcal{G}}', \hat{\mathcal{G}}''$, where $\hat{\mathcal{G}}' \sim q'(\cdot|\mathcal{G}), \hat{\mathcal{G}}'' \sim q''(\cdot|\mathcal{G})$, and $q'(\cdot|\mathcal{G})$, $q''(\cdot|\mathcal{G})$ are two augmentation functions composed of node dropout and edge dropout [Wang *et al.*, 2019a]. We apply these augmentations each training epoch and define a contrastive loss to maximize the agreement between different views of the same node, compared to that of other nodes. Specifically, we treat the views of the same node as positive pairs (i.e., $\{(h_i^{'}, h_i^{''})|i \in \mathcal{Y}_T\}$) and those of different nodes that appear in the same batch as negative pairs (i.e., $\{(h_i^{'}, h_j^{''})|i \in \mathcal{Y}_T, j \in \mathcal{B}_T, i \neq j, \phi_i = \phi_j\}$). Here $\mathcal{Y}_T$ is the set of users and recipes that appear in the training set, and $\mathcal{B}_T \subseteq \mathcal{Y}_T$ is a random batch of users and recipes in the training set. We use InfoNCE [Gutmann and Hyvärinen, 2010] to model the graph contrastive learning loss L_{con}:

$$L_{con} = \sum_{i\in\mathcal{Y}_T} -\log \frac{\exp(h_i^{'}, h_i^{''}/\tau)}{\sum_{j\in\mathcal{B}_T} \exp(h_i^{'}, h_j^{''}/\tau)}, \tag{12}$$

where τ is the temperature. The final objective function L is defined as the weighted combination of L_{rec} and L_{con}:

$$L = L_{rec} + \lambda L_{con}, \tag{13}$$

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

4 Experiments

In this section, we conduct extensive experiments to compare the performances of different models. We also show ablation studies, parameter sensitivity, case studies and embedding visualization to demonstrate the superiority of *RecipeRec*.

4.1 Experimental Setup

We employ the classic leave-one-out method to evaluate the model performance [He *et al.*, 2017]. During the testing, we randomly sample 100 negative recipes for each user and evaluate the performance using *Precision* (Pre), *Hit Rate* (HR), *Normalized Discounted Cumulative Gain* (NDCG), and *Mean Average Precision* (MAP). We report the performance under top@*K*, where *K* ranges from 1 to 10.

4.2 Baseline Methods

We compare our model with 14 baselines, including traditional recommendation approaches **BPR** [Rendle *et al.*, 2009] and **NeuMF** [He *et al.*, 2017], recipe representation learning and recommendation methods **Nutrition** [Teng *et al.*, 2012], **PopIngre** [Teng *et al.*, 2012], **IngreNet** [Teng *et al.*, 2012] and **Reciptor** [Li and Zaki, 2020], homogeneous GNNs **GraphSAGE** [Hamilton *et al.*, 2017] and **GAT** [Veličković *et al.*, 2018], heterogeneous GNNs **RGCN**

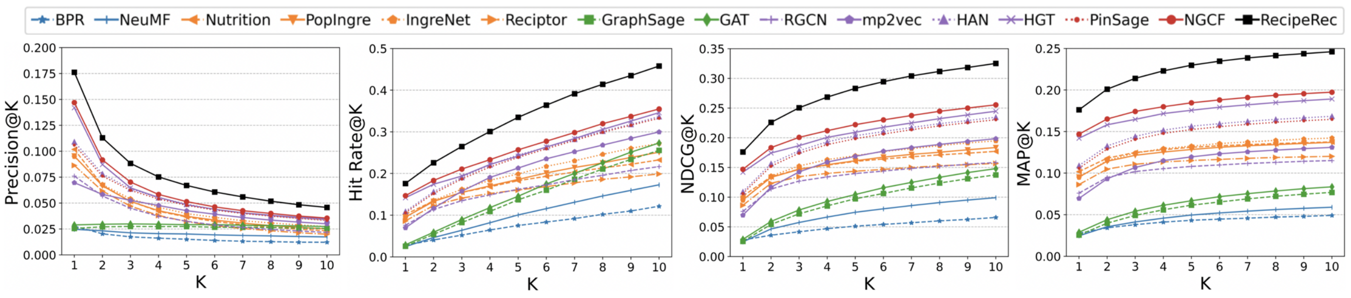

Figure 3: Performance of Top-K recipe recommendation where K ranges from 1 to 10.

[Schlichtkrull *et al.*, 2018], **mp2vec** [Dong *et al.*, 2017], **HAN** [Wang *et al.*, 2019b] and **HGT** [Hu *et al.*, 2020], and graph-based recommendation methods **PinSage** [Ying *et al.*, 2018] and **NGCF** [Wang *et al.*, 2019a].

4.3 Implementation Details

For the proposed *RecipeRec*, we set the learning rate to 0.005, the number of attention heads to 4, the hidden size to 128, the temperature τ to 0.07, λ to 0.1, node and edge dropout ratio to 0.1, batch size to 1024 and the training epochs to 100.

4.4 Performance Comparison

We report the performances of all models in Figure 3. As shown in the figure, *RecipeRec* outperforms all the baselines for all metrics over different Ks. Specifically, traditional recommendation approaches (i.e., BPR, NeuMF) perform poorly because they neither consider the higher-order collaborative signal nor the associated ingredients. Recipe recommendation approaches (e.g., PopIngre, IngreNet) obtain decent performance after incorporating the relational information, while Reciptor performs poorly since it is designed to learn recipe representations only, which ignores the user information. Homogeneous GNNs (i.e., GraphSage, GAT) behave poorly because of the neglect of node type information. Heterogeneous GNNs (e.g., HAN, HGT) and graph-based recommendation approaches (i.e., PinSage, NGCF) achieve satisfactory results, but fail to encode the influence of different nodes and relations as well as the complex interactions among nodes. Finally, *RecipeRec* achieves the best performance compared to all the baselines, demonstrating the effectiveness of the proposed model.

4.5 Ablation Study

RecipeRec is a joint learning framework composed of several neural network modules and optimized by a joint loss. We further conduct ablation studies to evaluate the performance of different model variants including: (a) ***RecipeRec-NA*** that only uses adaptive node-level attention and recommendation loss; (b) ***RecipeRec-RA*** that utilizes adaptive node-level attention, relation-level attention and recommendation loss; (c) ***RecipeRec-IT*** that combines all neural network modules but only uses recommendation loss. We report the performances of these model variants under different Ks in Figure 4. From the figure, we can find that the performance increases when we incorporate more modules

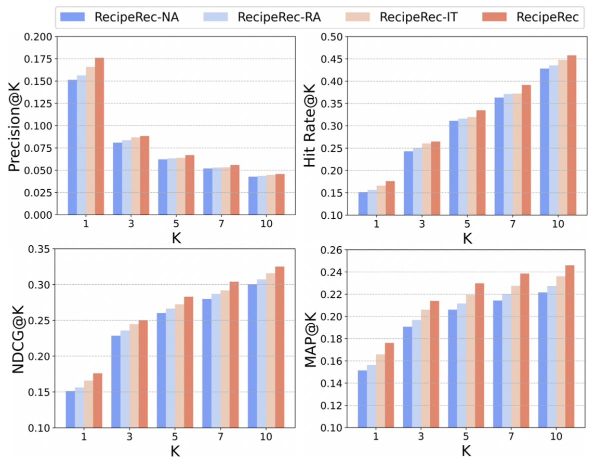

Figure 4: Results of different model variants.

and optimize the model with both losses. Among these variants, the full model *RecipeRec* achieves the best result in all cases. This demonstrates the effectiveness of adaptive node-level attention, relation-level attention, ingredient set transformer and graph contrastive augmentation in our model.

4.6 Parameter Sensitivity

We perform parameters sensitivity analyses, including the impact of dropout ratio, trade-off weight, and score predictor.

Impact of Dropout Ratio in Contrastive Learning. We conduct experiments to compare the influence of node dropout and edge dropout in Table 1. We can find that in most cases, the model performs the best when the ratio is 0.1. In general, the performance decreases when further increasing the dropout ratio, which indicates the suitable node/edge dropout ratio will lead to the best model performance.

Impact of Trade-off Weight in Joint Loss. To investigate the influence of graph contrastive learning, we study the impact of trade-off weight λ in Table 2. We can find that the model achieves the highest performance when λ is 0.1. Further increment of the value will decrease the performance. This demonstrates the incorporation of suitable graph contrastive augmentation can lead to the best recommendation results.

Impact of Score Predictor. To show the effectiveness of the score predictor used in our model, we examine the effect of other predictors (i.e., cosine similarity, multilayer perception)

Dropout	Ratio	K=1		K=5		K=10	
		HR	NDCG	HR	NDCG	HR	NDCG
Node Drop	0.1	15.49	15.49	**32.34**	**27.29**	**45.13**	**31.67**
	0.2	**15.77**	**15.77**	32.11	27.08	44.55	31.33
	0.3	15.66	15.66	31.93	27.25	44.30	31.46
	0.4	14.99	14.99	31.69	26.51	44.79	31.01
	0.5	14.68	14.68	30.72	25.76	43.10	29.99
Edge Drop	0.1	**17.18**	**17.18**	**32.94**	**27.73**	**45.12**	**31.88**
	0.2	17.01	17.01	32.01	27.29	44.40	31.53
	0.3	16.10	16.10	32.54	27.49	44.43	31.55
	0.4	16.16	16.16	31.79	26.91	43.99	31.08
	0.5	16.31	16.31	31.31	26.44	43.74	30.68

Table 1: Performance *w.r.t.* dropout ratio in contrastive learning.

λ	K=1		K=5		K=10	
	HR	NDCG	HR	NDCG	HR	NDCG
0.1	**17.33**	**17.33**	**32.88**	**28.00**	**45.70**	**32.37**
0.2	15.79	15.79	31.64	26.74	44.87	31.27
0.3	15.96	15.96	31.95	26.88	45.11	31.38
0.4	15.73	15.73	32.41	26.19	45.33	31.60
0.5	15.39	15.39	32.50	27.10	45.06	31.38

Table 2: Performance *w.r.t.* trade-off weight λ.

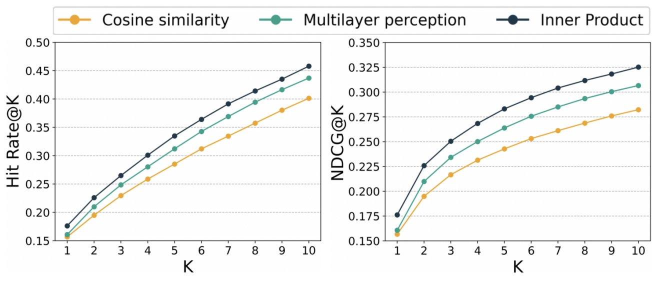

Figure 5: Performance *w.r.t.* the score predictor.

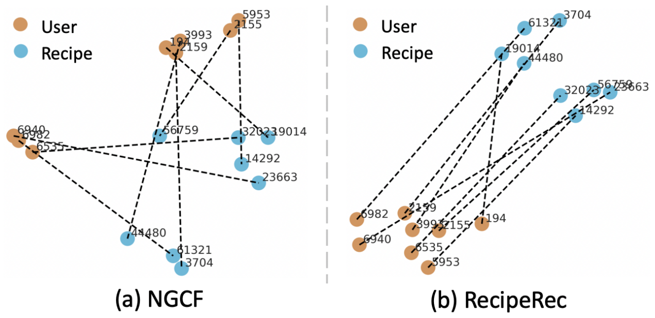

Figure 6: Embedding visualization of user and recipe. The edge indicates the interaction between user and recipe.

in Fig.5. From this figure, we conclude that inner product performs the best since it considers both the angle difference and the magnitude between two embeddings.

4.7 Case Study: Embedding Visualization

To show a more intuitive understanding and comparison, we randomly select 8 user-recipe interaction pairs and visualize their embeddings using t-SNE. As shown in Fig.6, NGCF can roughly separate the users and recipes into two clusters. There are gaps within each cluster and the lines between clusters are disordered. Ideally, there should be a clear mapping between each user-recipe pair (similar to the 'king-man=queen-woman' relationship in Mikolov *et al.* (2013)) and the lines should be parallel to each other. However, *RecipeRec* can easily divide the users and recipes into two condense clusters and obtain parallel lines for most pairs, which further demonstrates the superiority of our model.

5 Related Work

This work is closely related to the studies of recipe recommendation and graph neural networks (GNNs).

Recipe Recommendation. Existing recipe recommendation approaches are mostly content-based [Chen *et al.*, 2020] or collaborative filtering-based [Khan *et al.*, 2019] or hybrid [Gao *et al.*, 2020]. However, none of these approaches consider the higher-order collaborative signal such as relations. In our work, we propose a novel heterogeneous graph learning model to encode both recipe content and relational structure information. Additionally, existing recipe recommendation systems usually focus on the rating prediction task, which minimizes the errors between predictions and real ratings. This is concerned with only the observed recipes with explicit ratings, while our work uses a pairwise ranking strategy to examine the recommendation, which accounts for all the recipes in the collection.

Graph Neural Networks. Many graph neural networks [Veličković *et al.*, 2018; Wang *et al.*, 2019b; Zhang *et al.*, 2019] were proposed to encode the graph-structure data. They take advantage of content information and relation information in graphs to learn node embeddings for various downstream tasks such as recommendation. Recently, several graph-based models have been proposed for recommendation [Zhang *et al.*, 2017; Ying *et al.*, 2018; Wang *et al.*, 2019a]. However, most of them either fail to consider the collaborative signal. In this work, we propose a novel heterogeneous GNN to consider both collaborative signal and recipe information. In addition, contrastive learning has shown great performance in pre-training GNN [Qiu *et al.*, 2020]. We thus introduce a graph contrastive augmentation strategy to capture informative graph knowledge and improve the model.

6 Conclusion

In this paper, we propose and formalize the problem of *recipe recommendation with graphs*. To solve this problem, we create *URI-Graph*, a new and large-scale recipe graph data to facilitate graph-based food studies. Furthermore, we develop *RecipeRec*, a novel heterogeneous graph learning model for recipe recommendation. *RecipeRec* is able to capture both recipe content and the relational structure information through a graph neural network and a transformer. A novel combined objective function of recommendation loss and graph contrastive learning loss is employed to optimize the model. Extensive experiments demonstrate that *RecipeRec* outperforms state-of-the-art baselines.

Acknowledgements

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

[Britto *et al.*, 2020] Larissa Britto, Luciano Pacífico, Emilia Oliveira, and Teresa Ludermir. A cooking recipe multi-label classification approach for food restriction identification. In *ENIAC*, 2020.

[Chen *et al.*, 2020] Meng Chen, Xiaoyi Jia, Elizabeth Gorbonos, Chinh T. Hoang, Xiaohui Yu, and Yang Liu. Eating healthier: Exploring nutrition information for healthier recipe recommendation. *IP&M*, 2020.

[Dong *et al.*, 2017] Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. metapath2vec: Scalable representation learning for heterogeneous networks. In *KDD*, 2017.

[Gao *et al.*, 2020] Xiaoyan Gao, Fuli Feng, Xiangnan He, Heyan Huang, Xinyu Guan, Chong Feng, Zhaoyan Ming, and Tat-Seng Chua. Hierarchical attention network for visually-aware food recommendation. *IEEE Transactions on Multimedia*, 2020.

[Gutmann and Hyvärinen, 2010] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *AISTATS*, 2010.

[Hamilton *et al.*, 2017] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NeurIPS*, 2017.

[Haussmann *et al.*, 2019] 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 *ISWC*, 2019.

[He *et al.*, 2017] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *WWW*, 2017.

[Hu *et al.*, 2020] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. Heterogeneous graph transformer. In *WWW*, 2020.

[Khan *et al.*, 2019] Mansura A. Khan, Ellen Rushe, Barry Smyth, and David Coyle. Personalized, health-aware recipe recommendation: An ensemble topic modeling based approach. *arXiv preprint arXiv: 1908.00148*, 2019.

[Lee *et al.*, 2019] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam R. Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *ICML*, 2019.

[Li and Zaki, 2020] Diya Li and Mohammed J. Zaki. Reciptor: An effective pretrained model for recipe representation learning. In *KDD*, 2020.

[Marin *et al.*, 2019] 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. *TPAMI*, 2019.

[Mikolov *et al.*, 2013] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *ICLR*, 2013.

[Qiu *et al.*, 2020] Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. Gcc: Graph contrastive coding for graph neural network pre-training. In *KDD*, 2020.

[Rendle *et al.*, 2009] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *UAI*, 2009.

[Schlichtkrull *et al.*, 2018] Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *ESWC*, 2018.

[Teng *et al.*, 2012] ChunYuen Teng, Y. Lin, and Lada A. Adamic. Recipe recommendation using ingredient networks. In *WebSci*, 2012.

[Tian *et al.*, 2021] Yijun Tian, Chuxu Zhang, Ronald Metoyer, and Nitesh V. Chawla. Recipe representation learning with networks. In *CIKM*, 2021.

[Tian *et al.*, 2022] Yijun Tian, Chuxu Zhang, Ronald Metoyer, and Nitesh V. Chawla. Recipe recommendation with hierarchical graph attention network. *Frontiers in Big Data*, 2022.

[USDA, 2019] Agricultural Research Service. USDA. National nutrient database for standard reference, release 27. August 2019.

[Veličković *et al.*, 2018] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *ICLR*, 2018.

[Wang *et al.*, 2019a] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. Neural graph collaborative filtering. In *SIGIR*, 2019.

[Wang *et al.*, 2019b] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Peng Cui, Philip S. Yu, and Yanfang Ye. Heterogeneous graph attention network. In *WWW*, 2019.

[Ying *et al.*, 2018] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *KDD*, 2018.

[Zhang *et al.*, 2017] Chuxu Zhang, Lu Yu, Yan Wang, Chirag Shah, and Xiangliang Zhang. Collaborative user network embedding for social recommender systems. In *SDM*, 2017.

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