RECIPTOR: An Effective Pretrained Model for Recipe Representation Learning

Diya Li
Rensselaer Polytechnic Institute
Troy, NY, USA 12180
lid18@rpi.edu

Mohammed J. Zaki
Rensselaer Polytechnic Institute
Troy, NY, USA 12180
zaki@rpi.edu

ABSTRACT

Recipe representation plays an important role in food computing for perception, recognition, recommendation and other applications. Learning pretrained recipe embeddings is a challenging task, as there is a lack of high quality annotated food datasets. In this paper, we provide a joint approach for learning effective pretrained recipe embeddings using both the ingredients and cooking instructions. We present RECIPTOR, a novel set transformer-based joint model to learn recipe representations, that preserves permutation-invariance for the ingredient set and uses a novel knowledge graph (KG) derived triplet sampling approach to optimize the learned embeddings so that related recipes are closer in the latent semantic space. The embeddings are further jointly optimized by combining similarity among cooking instructions with a KG based triplet loss. We experimentally show that RECIPTOR's recipe embeddings outperform state-of-the-art baselines on two newly designed downstream classification tasks by a wide margin.

CCS CONCEPTS

• **Computing methodologies** → **Natural language processing**; • **Information systems** → **Data mining**.

KEYWORDS

food computing; recipe embedding; representation learning; set transformer; food knowledge graph

ACM Reference Format:
Diya Li and Mohammed J. Zaki. 2020. RECIPTOR: An Effective Pretrained Model for Recipe Representation Learning. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '20), August 23–27, 2020, Virtual Event, CA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3394486.3403223

1 INTRODUCTION

Food is fundamental to human beings as a necessity and for entertainment. Large-scale food data offers rich knowledge about food, and food data analysis can support a lot of human-centric applications in medicine, biology, gastronomy, and agronomy [20]. In particular, representing recipe data by embedding vectors can capture hidden information gleaned from massive food datasets and help further studies on food, such as eating habit advising [10] and health-aware food recommendation [35].

As illustrated in Fig. 1, a recipe usually contains a title, along with a set of ingredients (with their quantities and units), and cooking instructions in natural language sentences. Learning comprehensive (pre-trained) recipe representations which effectively encode all the content and can be broadly used for downstream applications still remains a challenge. Most previous work is task-oriented, and mainly aims to map the recipe text to corresponding food images in semantic space to obtain cross-modal embeddings that encode both text and visual features [4, 9, 17, 27, 32]. The cross-modal embeddings are often learnt from massive recipe datasets with abundant labels on recipe-image relations, where the labels are often used as the gold standard for objective function design. For instance, the objective of cross-modal embedding learning is to minimize the distance of recipe and image embeddings with the same label. These embeddings are then evaluated on a recipe-image retrieval task.

Unlike cross-modal embeddings, in this paper, we investigate the problem of recipe representation learning from only the textual content (i.e., using the ingredient set, and the step-wise instructions). Our goal is to find an effective latent space to represent recipes and demonstrate the effectiveness of the pretrained embeddings on various downstream tasks. One difficulty in recipe representation learning from text is the lack of (high-quality) labels in most existing large-scale recipe datasets [17, 25–27, 34]. Furthermore, the scant label information that exists is not very informative or useful, and therefore cannot be regarded as a gold standard for training. Those recipe datasets that do provide some labels are relatively small (56K [1], 46K [30], 28K [21], 66K [19]) and not suitable as a basis for learning large-scale pretrained embeddings for recipes. For effective recipe representation learning, we adopt the Recipe1M [17, 27] dataset which is the largest publicly available collection of recipes in English, containing over one million recipes.

While Recipe1M is a large dataset, its drawback is the lack of adequate (or curated) label information, and therefore it is not suitable for use in developing an effective objective function for learning. Instead, we adopt the triplet loss [5] for training, where the objective is to predict relative distances between inputs. The triplet loss demands triplet samples consisting of anchor, positive, and negative samples. For generating effective and feasible triplet samples, we further utilize external knowledge sources on recipes [3, 8, 10]. Among these, the FoodKG [10] is particularly relevant, since it is a semantics-driven knowledge graph built from the Recipe1M dataset, and it also integrates nutrient information, food ontology

KDD '20, August 23–27, 2020, Virtual Event, CA, USA

ACM ISBN 978-1-4503-7998-4/20/08…$15.00
https://doi.org/10.1145/3394486.3403223

and provenance information. We perform triplet sample mining from the knowledge graph through a similarity model.

The two major components of recipes, namely, ingredients and cooking instructions, are both considered in our representation learning. Intuitively, since ingredients are the building blocks of a recipe, it is important to consider them in effective representation learning. Whereas ingredients are a set and invariant to permutation, many existing works encode ingredients using sequential models like recurrent neural networks (RNNs) and long short-term memory networks (LSTMs) [4, 9, 17, 27, 32]. The other alternative to use a word2vec [18] based approach for ingredient embeddings is also not very effective, since the ingredient set size is typically small (e.g., on average 10 ingredients per recipe) and sparse. However, recent work has shown the effectiveness of preserving the permutation-invariant properties in modeling set data [36]. Therefore, unlike previous approaches, we develop permutation-invariant embeddings for the ingredient set. In addition, the interactions among the ingredients should also be emphasized. In order to quantify the interdependence among the ingredients, we employ a self-attention mechanism is applied over the ingredient set. Our approach therefore utilizes an approach based on the set transformer [15] – an attention-based permutation-invariant module – to encode the ingredients.

Besides ingredients, cooking instructions are also essential for discriminative recipe representation, as they provide details of the cooking process and provide extra contextual information which is not captured by the ingredients alone. For example, we can tell the difference between *pot stickers* and *boiled dumplings* (the most common type in northern Chinese cuisine) from their ingredients, as the former contains oil. However, there is little difference between *boiled dumplings* and *steamed dumplings* in terms of their ingredients. They are both composed of filling and wrapper, whereas the former is boiled and the latter needs to be steamed, and thus the cooking instructions will be different.

For effective pretrained recipe embeddings, we therefore jointly learn ingredient and cooking instruction embeddings in order to obtain discriminative recipe representations. We name our representation learning model the Reciptor, which is derived from the bold letters in the phrase *joint ingredient- and step-based m**o**del for p**re**trained **recip**e representation learning*, and our framework is illustrated in Fig. 1.

As we pointed out above, there is an additional hurdle in evaluating recipe embedding models, namely the lack of benchmark datasets or meaningfully labeled recipes. For example, the category labels in Recipe1M are highly skewed, with over 80% of the recipes belonging to the *background* class. Furthermore, the remaining recipes are labeled based on their main ingredients or words in their title, with useful classes accounting for only a small proportion of the recipes (see Fig. 2). To create new benchmark datasets, we expand the recipe data by adding new informative tags (which were scraped from the web for a large subset of the recipes in Recipe1M). The new tags reveal useful information including healthiness, cooking time, difficulty of cooking, cuisine category, region information, and so on (Fig. 1 shows some example tags for the given recipe).

To evaluate our pretrained recipe embeddings, we design two new downstream tasks utilizing the recipe tags. A cuisine category classification task is designed to verify the effectiveness of recipe representations and the adaptation of our Reciptor model. Furthermore, to explore the quality of recipe representations, a recipe network is constructed based on the cosine similarities between embeddings, and then a graph convolutional network [14] is used to predict the region tags (e.g., *American*, *Asian*, *Mexican*, etc.). Experimental results on the two downstream tasks show the applicability and strength of our joint Reciptor model. To summarize, our main contributions in this paper are as follows:

- We propose a joint model for recipe representation learning to align ingredients and cooking instructions in a latent semantic space, along with keeping the ingredients permutation-invariant and discriminative. The generalized model can be easily adapted to downstream tasks.
- We propose a novel knowledge graph based approach (using an external food knowledge graph) to extract relevant positive and negative triplet samples to enable the use of a triplet loss for recipe representation learning.
- We design two classification tasks to verify the effectiveness of our pretrained recipe embeddings. Reciptor achieves superior results compared to state-of-the-art baselines. We also provide benchmarks for the cuisine category classification and region prediction tasks based on user defined tags (not used for training).

2 RELATED WORK

Food computing mainly utilizes methods from computer science to address food-related issues in medicine, biology, gastronomy, and agronomy [20]. The fast development of online networks (e.g., social networks, recipe-sharing websites, cell-phones, and cameras) has led to large-scale food datasets with rich knowledge (e.g., recipes, food images and food logs) and can thus provide great opportunities for food-related studies, such as discovering principles of food perception [23], analyzing culinary habits [26] and recommending recipes from ingredients [30]. These works focus on specific tasks by analyzing the content of recipes [23], adopting word vectors for category classification [26], or building ingredient networks [30], while we focus on the general representation of recipes which encode the recipe textual content including ingredients and cooking instructions.

Recent breakthroughs in deep learning have further increased the interest in large-scale food-oriented studies, due to their superiority in learning representations from various types of signals. However, most of the existing work focus on computer vision/image based approaches, such as recognizing food from images [12, 24]. A few works have focused on recipe healthiness estimation [25] and recipe question answering [34]. In our work, we try to enable machines to interpret structural recipe text data containing ingredients and cooking steps. Such types of text data can be embedded in the latent space to further support various downstream applications, such as recipe recommendation.

There are also a handful of studies that have explored creating embeddings on the food domain but most of them target cross-modal embeddings. To learn cross-modal embeddings for cooking recipes and food images, Salvador et al. [27] have proposed a joint training framework through semantic regularization for

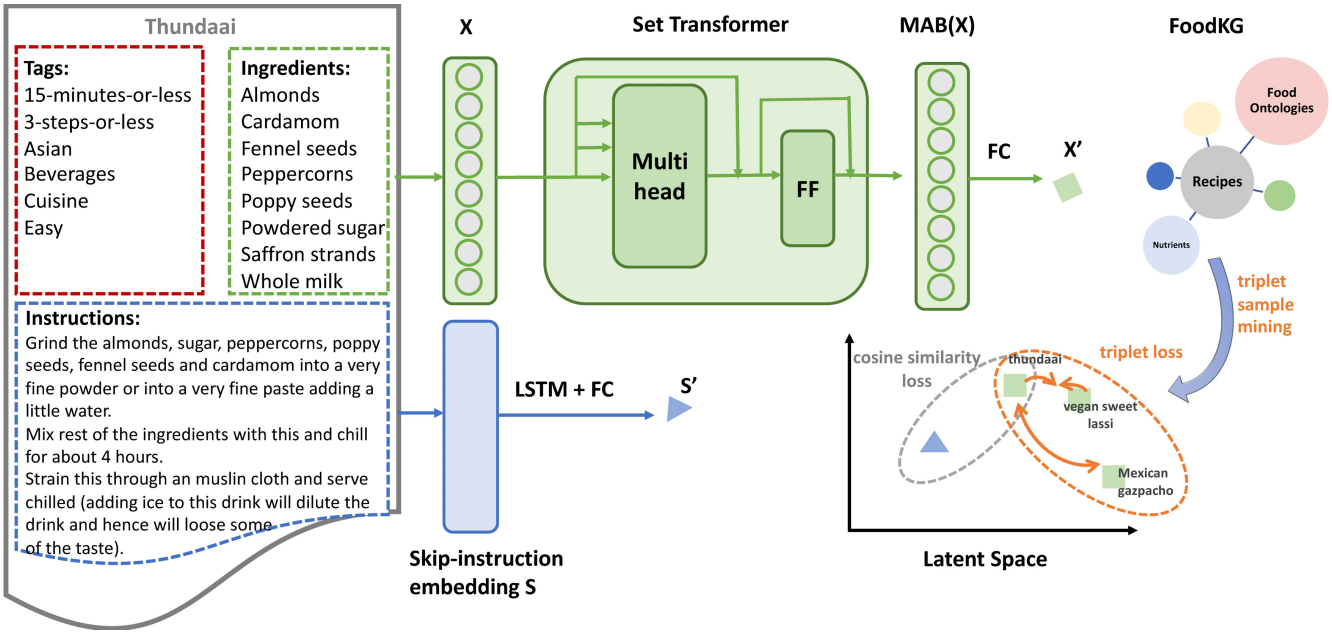

Figure 1: The Reciptor Model: A Set Transformer Joint Model over Recipe Ingredients and Steps. An example recipe is shown on the left for Thundaai – a popular milk-based drink in Northern India. The ingredients are shown in the green (dashed) box, whereas the steps appear in the blue (dashed) box. The tags are in red (dashed) box, and not used during training; they are used for benchmark dataset creation and downstream task evaluation. The deep learning architecture is shown on the right, and illustrates the set transformer based ingredient embeddings and the LSTM-based step embeddings, that are jointly trained via a novel knowledge graph based triplet loss, as well as cosine similarity loss.

the image-recipe retrieval task. Several related studies tackled this problem by adding other attributes (nutritional information, cuisine style) [11, 22], applying attention and self-attention mechanisms [6, 9], using adversarial networks [32], and introducing a double-triplet learning scheme for semantic category classification and retrieval tasks [4]. These models aim to align heterogeneous embedding spaces of images and text with minimum embedding distance as their objectives. The evaluation is also on the specific image-recipe retrieval task. Instead, we focus on the representation of recipe data at the textual level, and learn pretrained embeddings for various downstream tasks like label prediction and food recommendation.

3 THE RECIPTOR METHOD

We now introduce our Reciptor approach, illustrated in Fig. 1, which uses an attention-based permutation-invariant joint model for recipe representation learning (Section 3.1). To evaluate the quality of the pretrained recipe representations, two downstream tasks are also designed for extrinsic evaluation (Section 3.2).

3.1 Set Transformer Joint Model Construction

Assume each recipe $r = (X, S)$ is composed of 1) a set $X = \{x_1, x_2, \cdots, x_n\}$ of n ingredients, which can be single words (e.g., *almonds*) or short phrases (e.g., *whole milk*), and 2) a set $S = \{s_1, s_2, \cdots, s_l\}$ of cooking instructions containing l sentences (e.g., *"Mix rest of the ingredients with this and chill for about 4 hours."*). Following Marin et. al.'s work [17], we initialize each ingredient x_i with its word2vec embedding with d_w dimensions, and encode the cooking instructions with skip-instructions contextualized embeddings with d_s dimensions (see Section 3.1.3). Thus, we have $r = (X, S)$ with $X \in \mathbb{R}^{n \times d_w}$, $S \in \mathbb{R}^{l \times d_s}$ as the inputs to our Reciptor model, as illustrated in Fig. 1.

3.1.1 Encoding Ingredients with Set Transformer. Since the ingredients in a recipe comprise an (unordered) set, we need to capture this property in the recipe representation. Deep Sets [36] provide a simple way to construct permutation invariant set-input neural networks by defining objective functions on sets that are invariant to permutations. Self-attention [31] has also shown its power on many tasks by addressing and quantifying the interdependence among the inputs. To further effectively model interactions between ingredients within one recipe, we adopt the Set Transformer [15], an attention-based permutation-equivariant module to encode ingredients.

Given the ingredient set X containing n ingredients, assume we have n query vectors with dimension d_k for each corresponding ingredient, so that the matrix of query vector is $Q \in \mathbb{R}^{n \times d_k}$. To map queries Q to outputs using n_v key-value pairs with $K \in \mathbb{R}^{n_v \times d_k}$, $V \in \mathbb{R}^{n_v \times d_v}$, the attention function $\text{Att}(Q, K, V)$ is defined as:

$$\text{Att}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{1}$$

where $\frac{1}{\sqrt{d_k}}$ is a scaling factor. The function output $\text{Att}(Q, K, V)$ is a weighted sum of V where a value gets more weight if its corresponding key has a larger dot product with the query (computed $QK^T \in \mathbb{R}^{n \times n_v}$). The multi-head attention [31] idea was introduced to give the attention multiple "representation subspaces". By projecting Q, K, V onto h different d_k^m, d_k^m, d_v^m-dimensional vectors, respectively, the multi-head attention is then defined as:

$$\text{MultiHead}(Q, K, V) = \text{concat}(O_1, \cdots, O_j, \cdots, O_h)W^O \quad (2)$$

$$\text{where } O_j = \text{Att}(QW_j^Q, KW_j^K, VW_j^V) \quad (3)$$

where $W_j^Q, W_j^K \in \mathbb{R}^{d_k \times d_k^m}$, $W_j^V \in \mathbb{R}^{d_v \times d_v^m}$ and $W^O \in \mathbb{R}^{hd_v^m \times d}$ are learnable parameters. We set $d_k^m = d_k/h$, $d_v^m = d_v/h$, and $d = d_k = d_v = d_w$. The output $\text{MultiHead}(Q, K, V)$ is a linear projection on multiple attention subspaces.

Given two ingredient sets $X_1, X_2 \in \mathbb{R}^{n \times d_w}$, according to equation Eqs. (1) to (3), define X_1's multi-head attention on X_2 as follows:

$$\text{Attention}(X_1, X_2; \theta) = \text{softmax}\left(\frac{(X_1W_Q)(X_2W_K)^T}{\sqrt{d}}\right)(X_2W_V) \quad (4)$$

where $\theta = \{W_Q, W_K, W_V, \cdots\} = \{\{W_j^Q, W_j^K, W_j^V\}_{j=1}^h, \cdots\}$ denotes the trainable parameter set. Then, the multi-head self-attention $\text{SelfAtt}(X)$ on ingredient set X can be expressed as $\text{Attention}(X, X)$.

Let S_n be the set of all permutations of indices $\{1, \cdots, n\}$. Define a function $f : X^n \to Y^n$ to be permutation equivariant if and only if for any permutation $\pi \in S_n$, $f(\pi x) = \pi f(x)$. It's easy to show that the self-attention function is permutation equivariant due to Eq. (4):

$$\text{SelfAtt}(\pi \cdot X) = \pi \cdot \text{SelfAtt}(X) \quad (5)$$

The self-attention on the ingredient set ($\text{Attention}(X, X) \in \mathbb{R}^{n \times d_w}$), is next fed into the Set Attention Block (SAB) [15] defined as:

$$SAB(X) = MAB(X, X) \quad (6)$$

$$\text{MAB}(X_1, X_2) = \text{FFN}(WX_1 + \text{Attention}(X_1, X_2)) \quad (7)$$

where the Multihead Attention Block (MAB) is an adaptation of the encoder block of the Transformer [31] without positional encoding and dropout, FFN is a feed-forward network and W is a trainable parameter matrix. The set attention block $\text{SAB} \in \mathbb{R}^{n \times d_w}$ takes the ingredient set X as input, and after computing the multi-head self-attention between the ingredients, it results in a set of equal size.

After getting the encoded ingredients through SAB, which preserve the permutation-invariant property and are attentive to the inner interactions among ingredient set, a fully connected layer (FC) is added to obtain the final ingredient representation $X' \in \mathbb{R}^{d_r}$ as outlined in Fig. 1. The final ingredient representation X' is fixed at 600 dimensions ($d_r = 600$).

3.1.2 Triplet Loss and Triplet Sample Mining from an External Knowledge Graph. The biggest problem for recipe representation learning on text is that there is no well-defined goal for the model to train. To tackle this problem, we adopt a ranking loss where the objective is to predict relative distances between inputs. As a typical ranking loss, triplet loss has shown effectiveness on many metric learning tasks [29, 33]. The triplets are formed by an anchor recipe x_a, a positive sample x_p, and a negative sample x_n. The objective of triplet loss is to minimize the distance between the anchor recipe and the positive sample, and to maximize the distance between the anchor and the negative sample, in latent space, formalized as follows:

$$\mathcal{L}_{\text{triplet}}(x_a, x_p, x_n) = \max\Big(0, d(x_a, x_p) + \alpha - d(x_a, x_n)\Big) \quad (8)$$

where $d(x_i, x_j)$ expresses the cosine distance between recipe representations x_i and x_j, and the constant α ($\alpha > 0$) is a margin parameter to prevent pushing the representations of similar recipes to collapse into very small clusters.

To generate feasible triplets, we perform triplet sample mining from an external knowledge graph. In particular, we use FoodKG [10], which is one of the largest recipe knowledge graphs that integrates the recipe data derived from Recipe1M [27] with nutrient information from the USDA (which can be mapped to external ontologies), and it also incorporates the FoodOn ontology [8] to provide detailed information about the origin and preparation of foods, thus preserving the provenance information.

To extract informative (x_a, x_p, x_n) triplets for the triplet loss, we import the knowledge graph into an RDF (subject-predicate-object) datastore (in particular, we use Stardog; www.stardog.com), resulting in approximately 41 million RDF triples containing various tuple relations. We then use SPARQL queries to mine the triplet samples. Note that we have to use the recipe text directly, since during training we do not have access to good recipe embeddings. In fact, the whole point is to generate positive and negative samples to enable better representation learning. Therefore, the goal is to use textual features to extract the triplet samples via SPARQL queries on the RDF store.

Let r_a refer to the actual recipe text corresponding to the embedding x_a (similarly for p and n). For triplet mining, each recipe text r is therefore converted into a set of textual features comprising the recipe tags, ingredient names, and corresponding USDA food items. The stardog RDF store uses information retrieval and machine learning based approaches (textual features are vectorized via feature hashing and vectors are saved in a search index created using cluster pruning) to find the most similar nodes in the KG for a given SPARQL query. Given an anchor recipe, the output is a set of candidate recipes sorted by similarity score.

For triplet sample generation, given the recipe dataset R, we randomly choose a recipe $r_a \in R$ as the anchor sample, and we generate triplet samples by extracting the most relevant recipe as the positive sample r_p, and randomly choosing a negative sample r_n from the set of candidates with similarity score in the range $[0, 0.2]$. We run the triplet extraction process $|R|/3$ times, and the final set of triplets (x_a, x_p, x_n) are then used to optimize the triplet loss.

As an example, given an anchor recipe *French Beef Stew*, we get the positive sample *Crock Pot Beef Stew* with similarity score 0.742 and a negative sample *Clam Chowder* with similarity score 0.048. With the mined triplet samples from the knowledge graph, the triplet loss is computed over the recipe triplet's corresponding ingredient representations to get better alignment in semantic space. The triplet loss objective essentially enforces a semantic structure on the latent space by making sure that related recipes are closer to each other than to non-related ones. Fig. 1 illustrates an example of the triplet loss in action, where the embedding for *thundaai*

moves closer to *vegan sweet lassi*, and further away from *Mexican gazpacho*.

3.1.3 Jointly Training with Cooking Instructions. Since the triplet samples mined from the knowledge graph use only textual features of the recipes, solely utilizing triplet loss for recipe representation learning may simply learn to recreate the split, and may not necessarily capture the semantics of the recipes themselves. To tackle this problem, we jointly train the ingredients with cooking instructions. Previous work [17, 27] has shown that cooking instructions are necessary for recipe representation for a discriminative understanding of the cooking process. We follow Marin et. al.'s work [17] to use the pertained skip-instruction embeddings, denoted as S, as the initial input to our joint model. Skip-instruction embeddings are the product of a sequence-to-sequence model which encodes a sentence and uses that encoding as context when decoding/predicting the previous and next sentences, thus making the skip-instructions context-aware. A forward LSTM followed by a fully connected layer (FC) is used to get the final instruction representation $S' \in \mathbb{R}^{d_r}$. Since both the ingredient and instruction representations have d_r dimensions, we use the cosine similarity loss to align the ingredient representations with instruction representations in the latent space. To optimize the cosine loss, we randomly pick negative ingredient-instruction pairs with 80% probability during training.

In our RECIPTOR model, recipes are batched into triplets $(X_{\{a,p,n\}}, S_{\{a,p,n\}})$. As illustrated in Fig. 1, the final objective function of our model is a weighted combination of the cosine similarity loss and the triplet loss. Thus, we learn recipe representations by minimizing the following objective function:

$$\mathcal{L}(X'_{\{a,p,n\}}, S'_{\{a,p,n\}}; \theta) = (1-\lambda)\ \mathcal{L}_{\text{triplet}}(X'_a, X'_p, X'_n; \theta) + \lambda \sum_{i \in \{a,p,n\}} \mathcal{L}_{\text{cosine}}(X'_i, S'_i; \theta) \quad (9)$$

where θ is the set of all parameters, λ is a weight factor, which we set as 0.1, to focus more on the triplet loss $\mathcal{L}_{\text{triplet}}$ (over all the triplets, (X'_a, X'_p, X'_n), mined from the knowledge graph). $\mathcal{L}_{\text{cosine}}$ is the cosine similarity loss on the ingredient and instruction representations $(X'_{\{a,p,n\}}, S'_{\{a,p,n\}})$ over all recipe triplets.

Note that we mainly focus on learning informative ingredient representations, and the steps/instructions are treated as an auxiliary component for better learning. Thus, the jointly trained ingredient representations X' are used as the final recipe representation for downstream evaluation tasks, especially since they already incorporate the information from S'.

3.2 Recipe Representation Evaluation

Traditionally, there are two main approaches to evaluate representations: intrinsic evaluation and extrinsic evaluation. In the former, representations are tested directly for preserving syntactic or semantic relations. In the later approach, representations are used in downstream tasks [2, 28]. Methods for intrinsic evaluation are usually based on comparing representations with human judgment on tasks such as word relation and analogy, semantic similarity, concept categorization, synonym detection, and so on. Due to the lack of any gold standard or human judgment data for recipes, we focus on extrinsic evaluation on two downstream tasks.

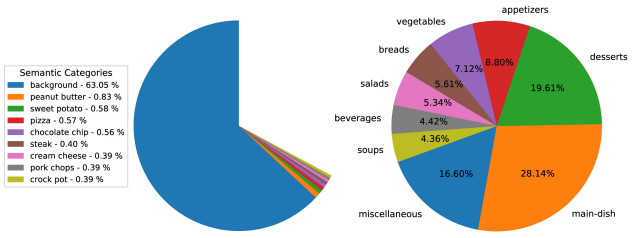

Figure 2: Tag statistics on Recipe1M subset (507,834 recipes). Left: distribution of the top 9 semantic categories on the subset; Right: new cuisine category distribution on the same subset.

3.2.1 Category Classification. The Recipe1M dataset [17] provides a total of 1,047 semantic categories which are parsed from the recipe titles. The semantic categories cover about half of the recipes (507,834 out of 1,029,720), with all those recipes without a semantic category assigned to a *background* class. They also assign *background* class to the remaining uncategorized dataset. However, as shown in Fig. 2, the *background* class composes a large portion (63.05%) of the subset, while the other top semantic categories only account for a small percentage, i.e., *peanut butter* accounts for 0.83%. Besides the biased distribution of semantic categories, we also observe that some categories are non-discriminative. For instance, there are 119 *chicken* related classes. Among them, some categories are indistinguishable like *grilled chicken* with *roast chicken*, *chicken breasts* with *chicken breast*, and *baked chicken* with *chicken bake*.

For a more robust evaluation, we therefore expand the Recipe1M subset with food tags that were scraped from www.food.com, since the FoodKG [10] knowledge graph doesn't take food tags into account. Taking the recipe *Thundaai* in Fig. 1, for example, it has 6 tags indicating the cooking time (*15-minutes-or-less*), difficulty of cooking (*3-steps-or-less, easy*), cuisine category (*beverage*) and region information (*Asian*). Using this kind of tag information, we create a new cuisine category label for each recipe spanning 8 classes, namely, *appetizers, beverages, breads, soups, salads, desserts, vegetables*, and *main-dish*. We assign the remaining untagged recipes as *miscellaneous*. As illustrated in Fig. 2, the 8 new classes cover 83.40% of the dataset and the distribution is more balanced compared to the semantic categories from Recipe1M.

We perform category classification over the new cuisine categories using our pretrained recipe embeddings. To make the evaluation focused on the quality of recipe representation, in the category classification task, a simple feed forward neural network (FFNN) is directly added to the joint model which maps representations to the 9 cuisine classes (8 main classes and 1 miscellaneous class). We set cross entropy loss as the objective function for multi-label classification.

3.2.2 Region Tag Prediction over Recipe Network. Besides category classification, we also want to explore the relationships between recipe representations in graph perspective. Therefore, we built a recipe network based on the cosine similarity between pretrained recipe embeddings.

In the recipe network, each recipe represents a node, and we assign an edge between two nodes if their cosine score is larger than a threshold δ. We also create a food region category from the recipe tags, for each node in the graph, with the labels comprising 5 classes (*American, Asian, European, Mexican*, and *Other*). The task is to predict the region labels over the nodes of the recipe network. Note that our pretrained recipe embeddings are used solely to create the recipe network, and not for the classification. The aim is to show that this graph contains important relationships between recipes that are useful for region tag prediction.

For the region prediction task we employ a graph convolutional network (GCN) [14], which is a widely used approach for modeling graph/network data by sharing filter parameters over all nodes in the graph. Here, we utilize a GCN to do region tag prediction over the recipe network. We set the input feature vector for each recipe/node in the GCN as the binary valued vector over the ingredient vocabulary. Each element indicates whether the corresponding ingredient is either present (1) or absent (0) in the recipe. Thus, the model input $X \in \mathbb{R}^{N \times D}$ is a feature matrix in which N is the number of nodes and D is the length of feature vectors (the size of the ingredient vocabulary). The adjacency matrix $A \in \mathbb{R}^{N \times N}$ of the recipe graph is used for propagation in the GCN model, as a representative description of the network structure. The layer-wise propagation rule at layer l is given as:

$$f(H^{(l)}, A) = \sigma\left(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right) \tag{10}$$

where $H^{(l)} \in \mathbb{R}^{N \times D}$ is the matrix of activations in the l^{th} layer, and $H^{(0)} = X$; $W^{(l)}$ is a weight matrix; $\hat{A} = A + I$, where I is the identity matrix; $\hat{D}$ is the diagonal node degree matrix of $\hat{A}$; and $\sigma(\cdot)$ is a non-linear activation function – we use ReLU [7]. Cross-entropy loss is used for the multi-label prediction task.

We again emphasize that we do not use the pretrained recipe representations as input features, since the goal of the downstream task is to examine the quality of recipe network (as opposed to the GCN's ability to do prediction). Thus, in our evaluation, we keep the model setting the same, but we use different approaches to construct the recipe network via different baselines.

4 EXPERIMENTAL EVALUATION

4.1 Datasets and Experimental Setup

The original Recipe1M dataset [17, 27] contains 1,029,720 recipes collected from over two dozen popular cooking websites, and includes English cooking recipes (ingredients and instructions), images, and categories. However, the category information is scarce for most recipes. Over 80% of the recipes are tagged as *background*. Furthermore, the cuisine categories apply to only a subset of the recipes; we use this subset comprising 507,834 recipes from the food.com website, with additional recipe tags scraped from the same website. We randomly choose 90% of the recipe triplets (457,050 recipes in total) as the training set for recipe representation learning, and the remaining 10% (50,784 recipes) as the validation set for parameter tuning. Each recipe in the subset contains more than 3 tags covering various properties. With the additional tag information, we obtain triplet samples from the FoodKG for recipe representation learning, and generate two new categories of downstream evaluation datasets and tasks. The code and data for Reciptor are available at https://github.com/DiyaLI916/Reciptor.

4.1.1 Category Prediction: Data and Baselines. In the category classification task, we keep the same partition ratio for training, validation, and test set as in [27]. The statistics on cuisine categories and number of samples are listed in Table 1. To test the effectiveness of

Table 1: Cuisine category statistics and number of samples in training, validation, and test sets over Recipe1M subset for category classification.

Category	#Recipes	Partition	#Recipes
Appetizers	44,707		
Beverages	22,453	Training	355,077
Breads	28,492		
Soups	22,119		
Salads	27,097	Validation	76,463
Desserts	99,563		
Vegetables	36,180		
Main-dish	142,913	Test	76,294
Miscellaneous	84,310		
Total			507,834

our model in the category classification task on cuisine categories dataset, as described in Section 3.2.1, we employ the same FFNN architecture, but we evaluate our recipe representation model Reciptor with respect to the following baseline embedding models and ablated versions:

- Word2Vec: This baseline sets the model input as ingredients encoded with word2vec [18], and uses a feed forward neural network as the multi-label classifier.
- Cross-Modal Embedding Baseline: This baseline sets the model input as the cross-modal embeddings that integrate the information of recipes and their corresponding images learnt via a joint model [17].
- Shallow Joint Model (SJM): similar to the joint model proposed in [27], the shallow joint model encodes ingredients with a Bi-LSTM network, and instructions with a forward LSTM. The goal in the shallow joint model is to align the ingredients and instructions in semantic space, using the cosine similarity loss between ingredients and instructions. This can be considered as an ablated version of our Reciptor model, without the triplet loss or the set transformer.
- Joint Model (JM): In this baseline, we use a joint model with ingredients encoded with Bi-LSTM and instructions encoded with a forward LSTM. The objective function is a combination of cosine similarity loss and triplet loss. This is an ablated version of our model without the set transformer.

4.1.2 Region Prediction: Data and Baselines. We also extract new region categories from recipe tags, as described in Section 3.2.2. The statistics of the regions and the size of the training, validation, and test sets are listed in Table 2. Note that we use only a subset of the original dataset for recipe network construction, mainly due to the size of the full recipe network comprising over 500K nodes and over a million edges. Using the full network causes memory usage issues

for the GPU-based learning, as the GCN operates on the whole recipe network with huge learnable matrices. Recall that the intent is not to judge the GCN model, but rather to evaluate whether the Reciptor pretrained embeddings lead to a higher quality recipe network compared to other baseline approaches to construct the graph. Thus, we select a 10% stratified subset of the recipe data, with the further constraint that the final recipe network remain as one connected component. Each class is sampled at a 10% rate, except for the overabundant *Other* category that is sampled at a 5% rate. This recipe network maintains good network properties, and can be used for further studies on food networks.

Table 2: Region category statistics and number of samples in training, validation, and test sets in Recipe1M subset for region prediction.

Category	#Recipes	#Recipes in Exp.	Partition	#Recipes
American	62,638	6,264 (10%)	Training	13,223
Asian	26,883	2,688 (10%)		
European	49,695	4,969 (10%)	Validation	3,306
Mexican	14,115	1,411 (10%)		
Other	354,503	17,725 (5%)	Test	16,528
Tag Total	507,834	Exp. Total		33,057

The baselines listed below are used to construct the recipe network, followed by the GCN approach for the region prediction task, with the goal to evaluate the quality of the recipe graph:

- word2vec, cross-modal, SJM and JM based networks: We construct the recipe network with the similarity score between two nodes/recipes based on the corresponding set of embeddings, separately for each of the methods. For word2vec, we take the average of the ingredients' embeddings as the recipe representation.
- FoodKG [10] based network: We construct the recipe network with similarity scores derived directly from the FoodKG knowledge graph, using the feature-vector based similarity model described in Section 3.1.2.

The main hyperparameters for all our model and experiments are listed in Table 3. We set the input feature dimension as 1433 for the region prediction task, since that is the size of pruned ingredient vocabulary after removing ingredients with frequency lower than 20. We tune all hyperparameters on the validation sets. All the neural network models are implemented using PyTorch v1.3.0. All experiments are conducted on a machine which has an Intel i7-2700K CPU and an Nvidia Titan Xp GPU with 16GB of memory.

4.2 Experimental Results and Analysis

Our evaluation of Reciptor's pretrained recipe representations on the two downstream tasks is shown in Tables 4 and 6. We show the Precision (*P*), Recall (*R*) and F1 scores (*F1*), where we use the micro-averaged values for the reported results to avoid potential class imbalance issues.

4.2.1 Quantitative analysis on category classification. For the joint models (SJM, JM, and Reciptor) used for category classification,

Table 3: Hyperparameters for Reciptor, category classification, and region prediction tasks.

Parameter Name	Values
Reciptor Model	
dimension of skip-instruction embedding	1024
dimension of ingredient embedding	300
number of head	2
weight parameter λ	0.2
batch size	300
training epoch	300
hidden size	600
learning rate	0.0001
optimizer	Adam [13]
Category Classification Task	
training epoch	50
dropout rate	0.1
learning rate	0.01
optimizer	AdamW [16]
Region Prediction Task	
threshold δ for network construction	0.9
input feature dimension	1433
training epoch	200
hidden size	100
dropout rate	0.7
learning rate	0.05
optimizer	Adam [13]

a simple feed-forward network is used as the classifier, and the hidden vector of ingredients (X') with 600 dimensions is fed into the classifier to get the final output. We can see in Table 4 that the shallow joint model (SJM) slightly outperforms the word2vec baseline, as the latter deals only with ingredients. The shallow model tries to map the ingredients and instructions together, and improves the quality of the recipe representation to some extent. The shallow joint model also shows competitive results compared to the cross-modal embedding baseline (76.4% vs. 76.9%), which suggests that cross-modal (text plus image) embeddings have limited improvement for text-based downstream tasks. The joint model (JM) has a huge improvement on the *F1* score (5% points) compared to the shallow joint model, which demonstrates the effectiveness of utilizing the triplet loss. The SJM can be treated as an ablated version of JM model without using FoodKG, which also demonstrates that the triplet samples mined from the knowledge reveal useful similarity relationships among recipes. However, our Reciptor model achieves the best performance on the classification task with nearly a 10% absolute improvement on the *F1* score compared to the word2vec and cross-modal embeddings, and close to 5% improvement over JM and 10% over SJM, both of which can be considered as ablated versions of Reciptor. This shows the effectiveness of both the permutation invariant embeddings, and knowledge graph based triplet loss evaluation.

To gain more insight, we analyze some misclassified cases shown in Table 5. The shallow joint model (SJM) misclassifies *Quick Cucumber Kimchee* as *appetizer* while correctly predicting *Kimchi Salad aka Quick Kimchi* as *salad*. In recipe representation learning, *Quick Cucumber Kimchee* and *Kimchi Salad aka Quick Kimchi*

Table 4: Category Classification Results (best results in bold).

Category	word2vec			Cross-Modal			Shallow Joint Model			Joint Model			Reciptor		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
Appetizers	73.5	72.2	72.8	75.0	74.4	74.7	75.8	74.4	75.1	79.4	78.8	79.1	**84.2**	**83.9**	**84.0**
Beverages	76.3	75.1	75.7	78.3	77.0	77.7	78.9	78.0	78.5	82.4	81.1	81.8	**87.9**	**85.7**	**86.8**
Breads	75.8	74.1	75.0	78.5	77.2	77.9	78.0	77.2	77.6	82.4	80.5	81.5	**87.0**	**85.5**	**86.3**
Soups	71.9	70.0	71.0	73.9	72.5	73.2	74.8	73.6	74.2	79.9	78.8	79.4	**84.0**	**83.4**	**83.7**
Salads	69.9	69.3	69.6	74.7	73.2	74.0	73.5	73.6	73.5	80.8	79.0	79.9	**84.9**	**83.3**	**84.1**
Desserts	77.3	75.2	76.2	81.7	80.0	80.9	80.6	79.4	80.0	84.3	82.8	83.6	**89.5**	**88.3**	**88.9**
Vegetables	72.1	69.8	71.0	74.0	72.1	73.1	75.2	72.4	73.8	82.3	80.9	81.6	**85.0**	**84.3**	**84.7**
Main-dish	74.9	73.8	74.4	78.3	77.0	77.6	77.0	75.8	76.4	81.7	81.2	81.5	**86.8**	**86.0**	**86.4**
Miscellaneous	71.5	72.7	72.1	76.0	74.6	75.3	75.0	74.4	74.7	81.1	80.7	80.9	**86.0**	**85.1**	**85.6**
Total	74.2	73.1	73.7	77.6	76.2	76.9	77.0	75.9	76.4	81.9	81.0	81.4	**86.7**	**85.7**	**86.2**

Table 5: Error cases for category classification on test set. Recipe ingredients are not fully listed for conciseness.

Recipe Title	Ingredients	Ground Truth	SJM	JM	Reciptor
Quick Cucumber Kimchee	English cucumber, carrot, fish sauce, honey garlic cloves, scallions, kosher salt, ...	Salad	Appetizer	Salad	Salad
Kimchi Salad aka Quick Kimchi	napa cabbage, garlic cloves, carrot, sugar fresh ginger, salt, scallions, ...	Salad	Salad	Salad	Salad
Rich Chocolate Pudding	refined flour, baking powder, baking soda cocoa powder, butter, dark chocolate, eggs fresh cream, golden syrup, condensed milk	Dessert	Bread	Bread	Dessert
Tofu-Almond French Toast	Italian bread, almond extract, ground cardamom cinnamon, ground cloves, soft silken tofu, ...	Miscellaneous	Bread	Bread	Bread

Table 6: Region Prediction Results (best results in bold).

Category	word2vec			Cross-Modal			FoodKG			Shallow Joint Model			Joint Model			Reciptor		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
American	68.6	66.3	67.4	71.7	70.4	71.0	74.7	72.9	73.8	72.9	71.5	72.2	73.9	73.6	73.8	**78.4**	**77.5**	**77.9**
Asian	69.8	68.7	69.3	73.2	71.9	72.5	75.8	74.6	75.2	73.0	71.0	72.0	76.4	74.5	75.4	**79.3**	**78.8**	**79.1**
European	71.3	70.8	71.0	74.8	74.0	74.4	77.5	75.0	76.2	75.0	71.8	73.4	76.8	75.0	75.9	**80.8**	**79.1**	**80.0**
Mexican	62.7	60.8	61.8	64.0	63.7	63.9	73.7	69.5	71.6	70.3	67.2	68.7	73.8	71.0	72.4	**75.2**	**73.7**	**74.4**
Other	70.0	67.6	68.8	73.3	71.4	72.4	76.0	74.8	75.4	73.5	72.0	72.8	75.7	74.0	74.9	**79.2**	**78.1**	**78.7**
Total	69.6	67.6	68.6	72.8	71.3	72.1	75.9	74.2	75.1	73.4	71.6	72.5	75.5	74.0	74.8	**79.1**	**78.0**	**78.6**

are in the same triplet, and are closely mapped in semantic space. Thus, the *Quick Cucumber Kimchee* is correctly predicted as *salad* in both the joint model (JM) and Reciptor, which use the triplet loss for better mapping. The *Rich Chocolate Pudding* is misclassified as *bread* in SJM and JM. One potential reason might be the ingredients contain *refined flour, baking powder* and *baking soda*, which are common in breads. Note that recipe titles are not used for representation learning to enable model generalization, and to assess the results. In this case, the *pudding* in the title clearly points to *Dessert* as the correct category. Our Reciptor model ignores the order of ingredients, and uses self-attention to model the interaction within ingredients, which enables it to correctly classify *Rich Chocolate Pudding* as a kind of dessert. As a final example, the last row in Table 5, shows the case of *Tofu-Almond French Toast*. This is in fact a type of bread, but is tagged as *miscellaneous*. All our joint models regard it as *bread*. Thus, this error is caused by tag ambiguity, and suggests that tag information is not perfect, and that we need to explore more effective measures to refine the tags.

4.2.2 Quantitative analysis on region prediction. For the region prediction task, the results are reported for a 2-layer GCN which achieves the best performance in our experiments. As shown in Table 6, among the three baselines – word2vec, Cross-Modal, FoodKG – the region prediction results on recipe network constructed from FoodKG performs the best (Word2Vec 68.6% vs. Cross-Modal 72.1% vs. FoodKG 75.1%). The FoodKG baseline also outperforms the two ablated joint models (shallow joint model 72.5% vs. joint model 74.8%). This is reasonable, since the FoodKG baseline builds the recipe network through a similarity model which operates directly on the knowledge graph with the tag information, which explicitly contains the region tags. The performance of cross-modal embeddings and the shallow joint model is similar, which indicates that the visual features do not assist much for region tag prediction.

Finally, RECIPTOR achieves the best results with a significant improvement on *F1* score compared to FoodKG (3.5% points), which indicates that recipe representations learnt from our model can construct a recipe network that captures essential relationships between recipes.

5 CONCLUSION AND FUTURE WORK

In this paper, we addressed the problem of learning general purpose or pretrained recipe representation based purely on the textual content of recipes, including both the ingredients and cooking instructions. We proposed the RECIPTOR model – a novel set transformer-based joint model to learn recipe embeddings. The permutation-invariant property of the ingredient set is maintained and the interaction among ingredient is enhanced via self-attention. Furthermore, we propose a novel knowledge graph derived triplet sampling approach to optimize the learned embeddings so that related recipes are closer in the latent space. Finally, the ingredient and step embeddings are jointly optimized by combining the cosine similarity and triplet loss. We experimentally show that RECIPTOR's recipe embeddings outperform state-of-the-art baselines on two newly designed downstream classification tasks; we also provide these two new benchmark datasets for follow-on research.

In the future, we plan to incorporate more information to improve RECIPTOR. We observe that there is still plenty of useful information that can be explored from a food knowledge graph like the nutrient and provenance information. One promising direction is to learn the nutrient information from FoodKG and integrate it into recipe representations. With the enrichment of recipe representation, more interesting downstream tasks can be developed like food healthiness prediction and healthy food recommendation.

ACKNOWLEDGMENTS

This work is supported by IBM Research AI through the AI Horizons Network.

REFERENCES

[1] Yong-Yeol Ahn, Sebastian E Ahnert, James P Bagrow, and Albert-László Barabási. 2011. Flavor network and the principles of food pairing. *Scientific Reports* 1 (2011), 196.
[2] Amir Bakarov. 2018. A survey of word embeddings evaluation methods. *arXiv preprint arXiv:1801.09536* (2018).
[3] Fernando Batista, Joana Paulo Pardal, Paula Vaz Nuno Mamede, and Ricardo Ribeiro. 2006. Ontology construction: cooking domain. *Artificial Intelligence: Methodology, Systems, and Applications* 41, 1 (2006), 30.
[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 *ACM SIGIR Conference*.
[5] Gal Chechik, Varun Sharma, Uri Shalit, and Samy Bengio. 2010. Large scale online learning of image similarity through ranking. *Journal of Machine Learning Research* 11, Mar (2010), 1109–1135.
[6] 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 International conference on Multimedia*.
[7] George Dahl, Tara Sainath, and Geoffrey Hinton. 2013. Improving deep neural networks for LVCSR using rectified linear units and dropout. In *IEEE international conference on acoustics, speech and signal processing*.
[8] Damion M Dooley, Emma J Griffiths, Gurinder S Gosal, Pier L Buttigieg, Robert Hoehndorf, Matthew C Lange, Lynn M Schriml, Fiona SL Brinkman, and William WL Hsiao. 2018. FoodOn: a harmonized food ontology to increase global food traceability, quality control and data integration. *npj Science of Food* 2, 1 (2018), 1–10.
[9] Matthias Fontanellaz, Stergios Christodoulidis, and Stavroula Mougiakakou. 2019. Self-Attention and Ingredient-Attention Based Model for Recipe Retrieval from Image Queries. In *5th Int'l Workshop on Multimedia Assisted Dietary Management.*
[10] 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 *International Semantic Web Conference*. Springer, 146–162.
[11] 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).
[12] Yoshiyuki Kawano and Keiji Yanai. 2014. Foodcam-256: a large-scale real-time mobile food recognition system employing high-dimensional features and compression of classifier weights. In *ACM international conference on Multimedia*.
[13] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
[14] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).
[15] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam R Kosiorek, Seungjin Choi, and Yee Whye Teh. 2018. Set transformer: A framework for attention-based permutation-invariant neural networks. *arXiv preprint arXiv:1810.00825* (2018).
[16] Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101* (2017).
[17] 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 Trans. Pattern Anal. Mach. Intell.* (2019).
[18] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.
[19] Weiqing Min, Bing-Kun Bao, Shuhuan Mei, Yaohui Zhu, Yong Rui, and Shuqiang Jiang. 2017. You are what you eat: Exploring rich recipe information for cross-region food analysis. *IEEE Transactions on Multimedia* 20, 4 (2017), 950–964.
[20] Weiqing Min, Shuqiang Jiang, Linhu Liu, Yong Rui, and Ramesh Jain. 2019. A survey on food computing. *ACM Computing Surveys (CSUR)* 52, 5 (2019), 1–36.
[21] Weiqing Min, Shuqiang Jiang, Jitao Sang, Huayang Wang, Xinda Liu, and Luis Herranz. 2016. Being a supercook: Joint food attributes and multimodal content modeling for recipe retrieval and exploration. *IEEE Transactions on Multimedia* 19, 5 (2016), 1100–1113.
[22] 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 international conference on Multimedia*.
[23] 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* 4 (2017), 15.
[24] Paritosh Pandey, Akella Deepthi, Bappaditya Mandal, and Niladri B Puhan. 2017. FoodNet: Recognizing foods using ensemble of deep networks. *IEEE Signal Processing Letters* 24, 12 (2017), 1758–1762.
[25] Markus Rokicki, Christoph Trattner, and Eelco Herder. 2018. The impact of recipe features, social cues and demographics on estimating the healthiness of online recipes. In *AAAI Conference on Web and Social Media*.
[26] 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 *International conference on world wide web companion*.
[27] 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 *IEEE CVPR Conference*.
[28] Tobias Schnabel, Igor Labutov, David Mimno, and Thorsten Joachims. 2015. Evaluation methods for unsupervised word embeddings. In *EMNLP Conference*.
[29] Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. Facenet: A unified embedding for face recognition and clustering. In *IEEE CVPR Conference*.
[30] Chun-Yuen Teng, Yu-Ru Lin, and Lada A Adamic. 2012. Recipe recommendation using ingredient networks. In *Annual ACM Web Science Conference*.
[31] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.
[32] Hao Wang, Doyen Sahoo, Chenghao Liu, Ee-peng Lim, and Steven CH Hoi. 2019. Learning cross-modal embeddings with adversarial networks for cooking recipes and food images. In *IEEE CVPR Conference*.
[33] Jiang Wang, Yang Song, Thomas Leung, Chuck Rosenberg, Jingbin Wang, James Philbin, Bo Chen, and Ying Wu. 2014. Learning fine-grained image similarity with deep ranking. In *IEEE CVPR Conference*.
[34] Semih Yagcioglu, Aykut Erdem, Erkut Erdem, and Nazli Ikizler-Cinbis. 2018. RecipeQA: A challenge dataset for multimodal comprehension of cooking recipes. *arXiv preprint arXiv:1809.00812* (2018).
[35] Longqi Yang, Cheng-Kang Hsieh, Hongjian Yang, John P Pollak, Nicola Dell, Serge Belongie, Curtis Cole, and Deborah Estrin. 2017. Yum-me: a personalized nutrient-based meal recommender system. *ACM Transactions on Information Systems (TOIS)* 36, 1 (2017), 1–31.
[36] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. 2017. Deep sets. In *NIPS*.