

A Hybrid Approach for Detecting Prerequisite Relations in Multi-modal Food Recipes

Liangming Pan, Jingjing Chen, Shaoteng Liu, Chong Wah Ngo, *Member, IEEE*,
 Min-Yen Kan, *Member, IEEE*, and Tat-Seng Chua, *Member, IEEE*

Abstract—Modeling the structure of culinary recipes is the core of recipe representation learning. Current approaches mostly focus on extracting the workflow graph from recipes based on text descriptions. Process images, which constitute an important part of cooking recipes, has rarely been investigated in recipe structure modeling. We study this recipe structure problem from a multi-modal learning perspective, by proposing a *prerequisite tree* to represent recipes with cooking images at a step-level granularity. We propose a simple-yet-effective two-stage framework to automatically construct the prerequisite tree for a recipe by (1) utilizing a trained classifier to detect pairwise prerequisite relations that fuses multi-modal features as input; then (2) applying different strategies (greedy method, maximum weight, and beam search) to build the tree structure. Experiments on the MM-ReS dataset demonstrates the advantages of introducing process images for recipe structure modeling. Also, compared with neural methods which require large numbers of training data, we show that our two-stage pipeline can achieve promising results using only 400 labeled prerequisite trees as training data.

Index Terms—Food Recipes, Cooking Workflow, Prerequisite Trees, Multi-modal Fusion, Cause-and-Effect Reasoning, Deep Learning

I. INTRODUCTION

Posting recipes and food images to sharing cooking experience is prevalent on social media. Millions of cooking recipes are available online on cooking sharing platforms, such as “all recipes”, “Cookpad”, and “Yummly”, etc. A recipe is usually presented in multimedia setting, with textual description of cooking steps aligned with process images to illustrate the visual outcome of each step. See Figure 1(a) for multimedia presentation of the recipe for “strawberry shortcake”. These information potentially provide opportunity for multimodal analysis of recipes, including cuisine classification [1], food recognition [2], [3], recipe recommendation [4], [5] and cross-modal image-to-recipe search [6], [7], [8], [9]. A common fundamental problem among these tasks is the understanding of the cause-and-effect relations of cooking process. Nevertheless, this problem is not fully explored and mostly addressed with text-only analysis. For example, text-based dependency parsing is employed for prerequisite tree construction [10], [11], [12], and hierarchical LSTMs is studied to model the causality effect for feature embedding [13].

Liangming Pan, Min-Yen Kan and Tat-Seng Chua are with National University of Singapore, Singapore. Jingjing Chen is with Fudan University, Shanghai, China. Chong Wah Ngo is with City University of Hong Kong, Hong Kong, China. Shaoteng Liu is with Xi'an Jiaotong University, Shanxi, China.

Corresponding author is Jingjing Chen (chenjingjing@fudan.edu.cn).

This paper studies the construction of prerequisite tree representation for recipe based on multimodal fusion. The representation provides the cooking workflow by ordering of instructions as a tree, which vividly describes the temporal evolution of food preparation as shown in Figure 1(b). Building a multi-modal prerequisite structure for food recipes is beneficial for both real-world cooking and automatic food analysis. In real-life, the prerequisite tree representation provides a clear guidance for the cook on which steps should be done sequential and which steps can be done in parallel. Many applications of food analysis, such as cross-modal retrieval and image-to-recipe generation, can also benefit from this representation. For example, in image-to-recipe generation, it is easier for machines to produce a recipe with an understanding of the prerequisite relations between cooking images. The workflow structure also benefits cross-modal retrieval by providing additional semantic information about cause-and-effect relations. Moreover, tree-based matching can be employed for similarity ranking of recipes for food recommendation.

We formulate the prerequisite tree construction as a classification problem, by classifying two instructions as either sequential or parallel. Sequential relation results in parent-child nodes, while parallel relation creates a new branch. By chaining these relations in temporal order, a prerequisite tree as depicted in Figure 1(b) can be produced. Constructing such tree is a non-trivial problem because it requires an deep understanding of both the process image (*e.g.*, tracking how ingredients are being cut and cooked) and the text description (*e.g.*, understanding how a cooking action changes the state of an ingredient).

Prerequisite relation detection for cooking recipes are previously addressed by defining hand-crafted textual features [14], [10], [11]. Although these features are able to capture shallow semantics, they are mostly domain dependent and not transferable across applications. To address this, our recent work [15] constructed a large-scale dataset, namely *MM-ReS*, consisting of 9,850 recipes with labeled prerequisite trees. Based on this dataset, advanced multi-modal deep learning architectures, such as Multimodal Bitransformers (MMBT) [16] are employed to detect prerequisite relations. Despite achieving promising results, we argue that this model can hardly be deployed in practical applications for two reasons. First, food recipes in the MM-ReS are mostly western cuisine. To generalize the model to other recipes such as Chinese cuisine, we need to label a large number of new prerequisite trees, which is extremely time-consuming and expensive. Second, due to the huge size of the neural model, it is hard to deploy

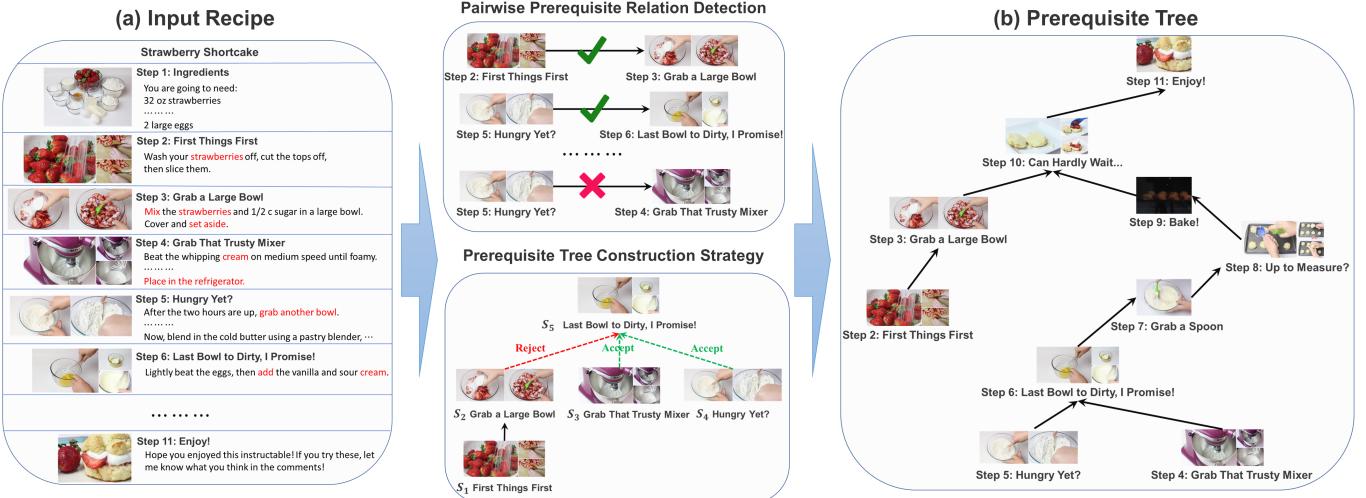


Fig. 1. Framework of multi-modal prerequisite tree construction. Given a multi-modal food recipe as the input (a), we propose a two-stage pipeline: 1) pairwise prerequisite relation detection, and 2) prerequisite tree construction strategy. The output is a multi-modal prerequisite tree (b).

the model to resource-intensive devices such as mobiles.

To address this problem, we propose a more practical pipeline, which first utilizes the pretrained visual and textual features for multimodal-based prerequisite relation detection, and then we employ different heuristic strategies to construct the prerequisite tree by temporally chaining the pairwise prerequisite relations. Figure 1 gives a sketch of tree construction, where a comprehensive tree is produced by temporally chaining the pairwise prerequisite relations. Compared with [15], our model is more light-weighted and only requires a few hundreds of training instances. Surprisingly, this only makes the model to suffer a slightly drop in performance. Experiments on the MM-ReS show that our method can achieve 80.39 in recipe-level accuracy, compared with the 82.44 accuracy achieved by [15].

II. RELATED WORK

The task of building multi-modal prerequisite tree for food recipes can be regarded as a twin problem of *recipe structure modeling* and *prerequisite relation detection*. Our research is also related to the broad topic of *Deep Learning for Food Analysis*.

A. Recipe Structure Modeling

Recipe structure modeling [17], [12], [11], [18], [19], [20], [21] aims to learn the semantic representations of cooking recipes. Based on the granularity of representation, existing methods can be categorized into *ingredient-level* and *instruction-level* structure modeling.

Ingredient-level recipe structure modeling aims to represent a recipe as a work-flow graph [11] or an action graph [10]. On the work-flow graph, each vertex represents either a cooking action or a raw ingredient and the directed edges represent the “action flows” describing the temporal execution sequence or “ingredient flows” tracking the ingredient sources. Manually constructing such graph is highly time consuming. Therefore, recent works mostly focus on leveraging machine learning

methods to automatically build the graph. In [10], an unsupervised hard-EM approach was proposed to automatically map instructional recipes to action graphs. The action graph is obtained with a segmentation model and a graph model. The segmentation extracts actions from text recipe while the graph model defines a distribution over the connections between the extracted actions. Yamakata *et al.* [11] further proposed to enrich the action graph with cooking tools and duration with semi-supervised method. The action graph is built via a four-step pipeline: word segmentation, recipe term identification, edge weight estimation, and manually action graph refinement. Nevertheless, the aforementioned methods cannot attain high quality recipe structures for real-world applications mainly due to two reasons: (1) the results are highly dependent on several NLP tasks, such as named entity recognition, co-reference resolution and dependency parsing, which are noise-prone due to free-form writing style in recipe, and (2) training with large-scale labeled dataset is infeasible because fine-grained recipe structure requires huge labeling efforts.

Compared with ingredient-level recipe modeling, instruction-level recipe modeling is more practical in terms of scalability. In [14], an ingredient-instruction dependency tree representation named Simplified Ingredient Merging Map in Recipes (SIMMER) was proposed to represent the recipe structure. SIMMER represents a recipe as a dependency tree with ingredients as leaf nodes and recipe instructions as internal nodes. In SIMMER, several hand-crafted text features were designed to train the Linear SVM-rank model for predicting ingredient-to-instruction links as well as instruction-to-instruction links. Similar to [14], we also focus on instruction-level recipe modeling; however, we study from the perspective of multi-modal learning by considering both text procedures and process images in the recipe. Moreover, instead of defining hand-crafted features, we improve the feature extraction using neural models with two-stage training to obtain deep semantic features.

B. Prerequisite Relation Detection

Our work is also related to prerequisite relation detection. Despite being a relatively new research area, data-driven methods for learning concept prerequisite relations have been explored in multiple domains. In educational data mining, prerequisite relations have been studied among courses or course concepts for curriculum planning [22], [23], [24], [25]. For example, researchers have tried to find general prerequisite structures by analyzing student assessment data [26], [27], [28]. Pan *et al.* [22], [29] proposed hand-crafted features such as video references and sentence references for learning prerequisite relations among concepts in MOOCs. Besides education domain, prerequisite relation has also been mined between Wikipedia articles [30], [31], concepts in textbooks [32], as well as concepts in scientific corpus [33]. There are also a few works attempting to build prerequisite structure based on existing prerequisite relations. For example, Yang *et al.* [24] proposed to mine the prerequisite relations among courses for curriculum planning based on pre-defined concept prerequisite relations; while Liang *et al.* [34] proposed to recover concept prerequisite relations from university course dependencies.

Our work is different from existing works in three aspects. First, by making full use of the multimedia nature of the recipes, we focus on mining the prerequisite relation to describe causality effect in cooking process. Second, we propose a neural-based pipeline to extract features from both image and text, which is more general than the domain-dependent hand-crafted features such as in [22]. Lastly, we propose a beam-search-based method for tree construction, which combines the greedy search and maximum score in building prerequisite structure.

C. Deep Learning for Food Analysis

This paper applies deep learning for food analysis. With the enormous success of deep learning, it has been widely adopted in various food-related tasks, including ingredient/food recognition [35], cross-modal retrieval [36], [37], [38] and recipe generation [39]. Food recognition typically involves food categorization [40], [41], ingredients recognition [42], [43], [44], and food attributes recognition [2], [8]. Deep features extracted from DCNN [45], which is trained on ImageNet [46] and fine-tuned on food images, often exhibit impressive recognition performance [47], [48]. In this work, we follow this paradigm to extract deep features in cooking images.

Cross-modal learning in food domain has started to attract research interest in recent years, and several large food and recipe datasets have been developed recently, for example, Cookpad [49] and Recipe1M+ [50] datasets. Existing neural-based methods [8], [51], [6] typically learns a joint embedding space between food images and recipe texts. For example, in [8], deep belief network is used to learn the joint space between food images and ingredients extracted from recipes. However, previous works consider a recipe as a whole, while ignoring its inherent structure. Different from these works, our work investigate the cause-and-effect relations inherent in

cooking recipes, based on which we can learn better recipe representations to benefit downstream tasks.

III. METHODOLOGY

In this section, we first give the definition of prerequisite tree (Section III-A), and then formally introduce our proposed two-stage framework consisting of: learning to detect pairwise prerequisite relation (Section III-B), and constructing prerequisite tree based on learned pairwise relations (Section III-C).

A. Problem Formulation

A **recipe** is composed of n cooking steps, denoted as $R = \{S_1, \dots, S_n\}$, where S_i is the i -th step. Each **cooking step** S is further represented as its text description and cooking images, *i.e.*, $S = \{W, G\}$, where the text description W is a word sequence $(w_1, \dots, w_{|W|})$, and G is the set of cooking images, *i.e.*, $G = \{g_1, \dots, g_{|G|}\}$.

Next, we give the definition of prerequisite tree. Before this, we first define the **dependency tree** of a recipe R , denoted as \mathcal{D}_R , as a directed tree that satisfies the following conditions.

- \mathcal{D}_R has n nodes S_1, \dots, S_n , where S_i represents the i -th step in the recipe. The root node of \mathcal{D}_R is the last step S_n .
- For any none-root step S_i in \mathcal{D}_R , its parent S_{P_i} must be a subsequent step of S_i , *i.e.*, satisfying $i < P_i$.
- Each directed edge in \mathcal{D}_R points from S_i to its parent S_{P_i} .

The **prerequisite tree** of R , denoted as \mathcal{T}_R , is then defined as a dependency tree that satisfies the following condition.

- For any two nodes S_i and S_j ($i < j$), there is a path from S_i to S_j in \mathcal{T}_R *i.f.f.* S_i is a *prerequisite cooking step* of S_j , *i.e.*, we cannot perform step S_j without finishing step S_i .

Figure 1(b) shows the prerequisite tree of the recipe “Strawberry Shortcake”. Step 5 is a prerequisite step of 9 since the dough has to be made before baking it. However, step 2 and step 9 are independent since we can bake the dough without preparing the strawberry.

Given a recipe R as input, the objective is to build the prerequisite tree \mathcal{T}_R . This poses two major challenges: (1) how to judge whether a given step is a prerequisite step of another, and (2) how to construct the prerequisite tree based on noisy and even contradictory pairwise predictions. To address the first challenge, we extract semantic features from both images and texts with a two-stage training process and then use early/late-fusion to combine them for prediction. For the second challenge, we propose a beam-search-based strategy to build the prerequisite tree, which makes a balance between greedy search and global optimization.

B. Pairwise Prerequisite Relation Detection

We first learn to detect prerequisite relations between cooking steps. Specifically, given any two steps S_i and S_j ($i < j$) from a recipe R , the objective is to learn a function $P(\langle S_i, S_j \rangle)$ that maps a step pair $\langle S_i, S_j \rangle$ to a scalar value between 0 and 1, representing the probability that S_i is a prerequisite step of S_j .

1) *Training Data Acquisition*: To train the pairwise prerequisite relation classifier P and evaluate the performance, we harvest step pairs from the labeled prerequisite trees in the training data. Specifically, for a given recipe $R = \{S_1, \dots, S_n\}$ and its associated prerequisite tree \mathcal{T}_R , we obtain labeled step pairs following the definition of prerequisite tree using the two rules below.

- For any step S_i and its parent S_{P_i} , we harvest a positive sample ($x = \langle S_i, S_{P_i} \rangle, y = +1$).
- If there is no path from S_i to S_j in \mathcal{T}_R , we harvest a negative sample ($x = \langle S_i, S_j \rangle, y = -1$).

This gives us a dataset of labeled pairwise prerequisite relations $\mathcal{D} = \{v(x^{(i)}), y^{(i)}\}_{i=1}^M$, based on which we train the feature extractor and binary classifier to learn the mapping $\mathcal{P} : v(x) \rightarrow y$.

2) *Image Feature Extraction*: Process images serve as an important clue for detecting prerequisite relations. In most cases, visually similar steps are operating on the same ingredient, thus they are more likely to have a prerequisite relation (e.g., Step 2 and Step 3 in Figure 1(b)). We use ResNet-50 [52] to extract features for cooking images. ResNet introduces the residual block to alleviate the gradient vanishing problem of deep convolution network, which allows the model to go deeper without much increase in training difficulty. The pre-trained ResNET-50 has been widely used to extract image features in computer vision [53], [7]. It inputs the raw pixels of an image and outputs its image feature vector. To make the model more adaptable to the food domain, we fine-tune the pre-trained ResNet-50 with Recipe1M [6] dataset, which contains 251,980 training images of 1,047 food categories (e.g., chocolate cake, cookie). The image features learned by the last convolutional layer of the ResNET-50 is projected to a softmax output layer to predict the food category during training. After fine-tuning, we drop the softmax layer and use the outputs from last layer as image features.

3) *Text Feature Extraction*: To extract deep semantic features from instruction texts, we adopt the idea of language model (LM) pre-training [54], [55], [56], [57], [58], which first pretrain neural networks on large-scale unlabeled text corpora, and then finetune the models or representations on downstream tasks. Among them, the Bidirectional Encoder Representations from Transformers (BERT) [56] is a multi-layer Transformer network [59] consisting of stacked self-attention layers, which is a widely-used pretraining approach in NLP. Given an input sequence $s = \{[\text{CLS}], w_1, \dots, w_{|s|}\}$ ([CLS] is special token marking the whole sequence), BERT first packs them together into $\mathbf{H}^0 = [\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_{|s|}]$, where \mathbf{x}_i is the input vector for token w_i . Then, an L -layer Transformer is used to encode the input to obtain the final layer representations $\mathbf{H}^L = [\mathbf{h}_0^L, \mathbf{h}_1^L, \dots, \mathbf{h}_{|s|}^L]$. Typically, \mathbf{h}_0^L is used as the semantic representation of the whole sequence s .

BERT is pre-trained on two large corpus (English Wikipedia and BookCorpus [60]) for the masked token prediction task. Given the input token sequence s , a certain portion of tokens are replaced by a special symbol [MASK], and the model is trained to recover the original tokens from the corrupted

version \hat{s} . As BERT is not trained on recipe data, we fine-tune BERT with the same masked token prediction task on the training dataset. The fine-tuned model is then used to extract text features for each step. We treat each step as a sequence of tokens and obtain the semantic representations from \mathbf{h}_0^L .

4) *Multi-modal Fusion*: To fuse the features from different modalities, we explore two widely-adopted paradigms: (1) *early-fusion* which combines the image and text features before predicting the prerequisite relations, and (2) *late-fusion* which first predicts the prerequisite relations from each modality separately and then combines the prediction results.

Early Fusion. As shown in Figure 2(a), given a step pair $\langle S_i, S_j \rangle$, we first extract the feature vectors of process images and text instructions respectively for steps S_i and S_j . The resulting feature vectors are denoted as m_i and m_j for images, and e_i and e_j for text features. We then add a feature transformation layer to map the image feature and text feature into the same semantic space with same dimension. Feature transformation layer is implemented as a feed forward network with one hidden layer. The transformed vectors for m_i , e_i , m_j , and e_j are concatenated as a single vector v , and then we employ a feed forward network with 2 hidden layers to learn the prerequisite relations based on the feature vector v . In training, the parameters for the feed forward layer and the feature transformation layer are jointly trained, while the parameters for ResNet-50 and BERT are fixed.

Late Fusion. As shown in Figure 2(b), in late fusion, we train two prerequisite relation classifiers separately from image features and text features. In testing time, the output probability from the image side (P_m) and from the text side (P_e) are linearly combined to obtain the final prediction score P , i.e., $P = \alpha \cdot P_m + (1 - \alpha) \cdot P_e$, where α is a parameter trading off the prediction. To train a text-based relation classifier, we use BERT in double-sentence mode, in which the texts from step S_i and S_j are concatenated as a single sequence separated by a special token [SEP]. The extracted feature vector e is then linked to a feed forward network with 2 hidden layers to predict prerequisite relations. For image-based relation classifier, we adopt the Siamese network [61], a famous network structure in learning image relations. The siamese neural network consists of twin networks (in purple) which accept distinct inputs but share weight matrices at each layer. We follow the structure of [61] to use a network with three convolutional layers, with a contractive loss function at the top.

C. Prerequisite Tree Construction

We denote the learned pairwise prerequisite relation classifier as a function $P(S_i, S_j)$ that takes any step pair $x = \langle S_i, S_j \rangle$ from a recipe R as input, and outputs a confidence score ranging from 0 to 1. The score represents the probability that S_i is a prerequisite step of S_j , denoted as $S_i \rightarrow S_j$. The major challenge of constructing the prerequisite tree from $P(S_i, S_j)$ is the presence of conflicting relation between step pairs, which results in transitivity property cannot be

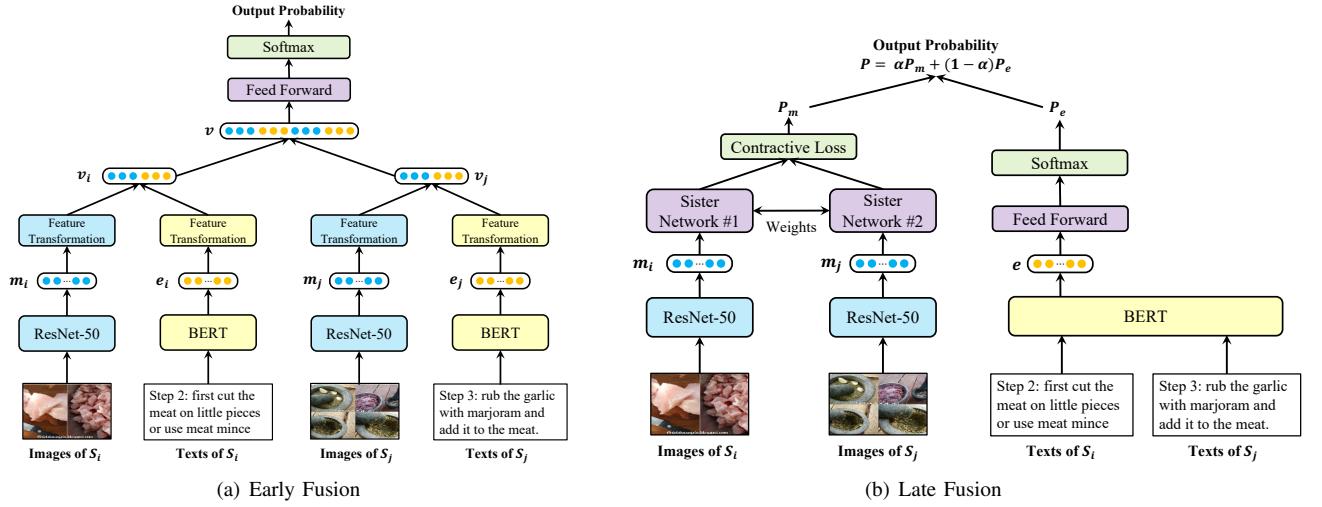


Fig. 2. Multi-modal Fusing for Prerequisite Relation Detection. Early fusion (a) concatenates visual and textual features before detecting prerequisite relation. Late fusion (b) first detect prerequisite relation from texts and images separately and then combines the predictions.

guaranteed for pairwise prediction. An example is: $S_1 \rightarrow S_2$, $S_2 \rightarrow S_3$, but $S_1 \not\rightarrow S_3$. In this case, the tree can be either $S_1 \rightarrow S_2 \rightarrow S_3$ or $S_1 \rightarrow S_2, S_1 \rightarrow S_3$. How to address these potential contradictions in tree construction depends on the prior assumption and the construction strategy. This section first proposes two baseline algorithms based on greedy search and maximum score, which give priority to step pairs in close proximity and high-scoring step pairs respectively. A generalized algorithm based on beam search is then proposed to combine their respective advantages.

1) *Greedy-based Method*: Given a recipe $R = (S_1, \dots, S_n)$ with n steps as input, *Greedy-based Prerequisite Tree Construction* (Greedy-PTC) iteratively builds the tree by manipulating the *dependency forest* \mathcal{F} (initialized as an empty set) each time the algorithm reads in a new step from R . Each tree in \mathcal{F} is a subtree of the final prerequisite tree, and the node set of \mathcal{F} contains all steps that have been read before. In the i -th iteration, the algorithm reads step S_i and tests its prerequisite relation with the root node of each tree $T \in \mathcal{F}$ via the learned prediction function $P(S_i, r_T)$ (r_T is the root of T). Once $P(S_i, r_T)$ is above a pre-defined threshold θ , a new edge from r_T to S_i is created. Note that two trees T_1 and T_2 in \mathcal{F} will merge into a single tree with S_i as the root, if both $P(S_i, r_{T_1})$ and $P(S_i, r_{T_2})$ are above the threshold. Finally, if S_i is not connected to any tree in \mathcal{F} , S_i is treated as a single-node tree and add it to the forest \mathcal{F} . The greedy-PTC essentially links S_i to S_j whenever it finds an S_j that satisfies $P(S_i, S_j) > \theta$. This makes the greedy-PTC to some extent “short-sighted”, preferring to link S_i to a satisfactory subsequent step close in distance, but failing to consider better steps after S_j .

2) *Maximum-weight Method*: The *Maximum-weight Prerequisite Tree Construction* (Max-PTC) addresses the problem of greedy-PTC by searching for the prerequisite tree \mathcal{T}_R globally, *i.e.*, finding the best-scoring tree from all possible dependency trees of R , denoted as Ω_R . The score of a dependency tree $T \in \Omega_R$ is simply defined as the sum of

prediction scores of all $n - 1$ edges in T , as follows.

$$\mathcal{T}_R = \arg \max_{T \in \Omega_R} \left\{ \sum_{(S_i, S_j) \in \mathcal{E}_T} P(S_i, S_j) \right\} \quad (1)$$

where \mathcal{E}_T is the edge set of T . Note that in a dependency tree, the parent step of S_i ($1 \leq i \leq n - 1$) comes from one of its subsequent steps $\Lambda_i = \{S_{i+1}, \dots, S_n\}$. With this constraint, to find a maximum-scoring tree in Ω_R , for each step S_i , we simply need to choose the step $S \in \Lambda_i$ with the maximum prediction score $P(S_i, S)$ as the parent of S_i , *i.e.*, $S_{P_i} = \arg \max_{S \in \Lambda_i} P(S_i, S)$. In this way, we can efficiently find \mathcal{T}_R in a time complexity of $\mathcal{O}(n^2)$.

Although max-PTC addresses the shortsightedness problem in greedy-based method, a maximum-scoring tree is not guaranteed to be a prerequisite tree. Consider a tree where each step S_1, \dots, S_{n-1} directly links to the last step S_n . As each step S_i ($1 \leq i \leq n-1$) is a prerequisite step of S_n , the overall score for the tree is high, although it is not the prerequisite tree we want. In practice, max-PTC tends to directly link early steps to latter steps without considering the intermediate process. This problem is not likely to happen in greedy-PTC as closer steps are processed in priority.

3) *Beam Search-based Algorithm*: To inherit the advantages of both greedy-PTC and max-PTC, we propose a novel *Beam Search-based Prerequisite Tree Construction* (beam-PTC) algorithm to jointly consider the distance and the score in the building process. Given an input recipe $R = (S_1, \dots, S_n)$, we sequentially read one step at a time. In the i -th iteration, the algorithm maintains \mathcal{L} number of *intermediate states* $\Psi^{(i)} = \{F_1^{(i)}, \dots, F_{\mathcal{L}}^{(i)}\}$. Each intermediate state $F_j^{(i)}$ is a dependency forest with i nodes in total. In each iteration, given the previous intermediate states $\Psi^{(i-1)} = \{F_1^{(i-1)}, \dots, F_{\mathcal{L}}^{(i-1)}\}$, and the current step S_i , we derive the intermediate states for the current iteration $\Psi^{(i)}$ via over-generating and ranking. First, for each state $F_k^{(i-1)}$, we generate its derived states by enumerating all valid dependency forests after adding S_i as an additional node. As an example, in Figure 3, we show the first

three iterations of beam-PTC. We generate 4 derived states $(F_1^{(3)}, \dots, F_4^{(3)})$ for the state $F_1^{(2)}$, by considering all valid cases after adding the node S_3 . Then, we rank all derived states $\Psi_d^{(i)}$ and select the top- \mathcal{L} scoring states as the intermediate states $\Psi^{(i)}$ for the current iteration. The full algorithm is presented in Algorithm 1.

Algorithm 1: Beam Search Algorithm for Prerequisite Tree Construction

Input: Recipe $R = \{S_1, \dots, S_n\}$, Prediction function $P(\cdot, \cdot)$, State size \mathcal{L}
Output: Prerequisite tree \mathcal{T}_R

$$\Psi_d^{(0)} \leftarrow \{\emptyset\}$$

```

for  $i$  from 1 to  $n$  do
     $\Psi_d^{(i)} = \emptyset$ 
    foreach state  $F$  in  $\Psi_d^{(i-1)}$  do
         $\Psi_d^{(i)} \leftarrow \Psi_d^{(i)} \cup \text{derive\_states}(F, S_i)$ 
    end
     $\Psi^{(i)} \leftarrow \text{select\_topN}(\Psi_d^{(i)}, \mathcal{T})$ 
end
 $\mathcal{T}_R \leftarrow \text{select\_topN}(\Psi^{(n)}, 1)$ 
return  $\mathcal{T}_R$ 

```

To rank the derived states, we propose a *scoring function* h to evaluate each state. A state (random forest) F can be formulated as a directed graph with a node set $\mathcal{N} = \{S_1, \dots, S_m\}$ and an edge set \mathcal{E} . The score for state F is defined as follows.

$$h(F) = \sum_{1 \leq i < j \leq m} h(S_i, S_j) \quad (2)$$

$$h(S_i, S_j) = \begin{cases} P(S_i, S_j), & \text{if } \langle S_i, S_j \rangle \in \mathcal{E} \\ 1 - P(S_i, S_j), & \text{otherwise} \end{cases} \quad (3)$$

where $h(S_i, S_j)$ is the *edge score* of $\langle S_i, S_j \rangle$. If this edge exists in F , its edge score is the prediction score $P(S_i, S_j)$. Otherwise, we set the edge score as $1 - P(S_i, S_j)$. To calculate the score of the state, we enumerate all $\langle S_i, S_j \rangle$ pairs with $1 \leq i < j \leq m$, and sum up their edge scores. For example, in Figure 3, suppose $P(S_1, S_2) = 0.55$, $P(S_1, S_3) = 0.6$, $P(S_2, S_3) = 0.3$, the score for state $F_2^{(3)}$ would be $(1 - 0.55) + 0.6 + 0.3 = 1.35$.

When $\mathcal{L} = 1$, beam-PTC is equivalent to greedy-PTC with the threshold $\theta = 0.5$, as we only derive the most promising state in each iteration. When $\mathcal{L} = \infty$, the states in the last iteration will contain all valid dependency trees of R , *i.e.*, $\Psi^{(n)} = \Omega_R$. In this case, beam-PTC is equivalent to max-PTC.

4) *Time Complexity*: We analyze the time complexity for each algorithm in this section. Note that we denote n as the number of steps and \mathcal{L} is the beam size. For greedy-based method, we iteratively read in the new step for n times, and each time we perform one testing operation which requires constant time $\mathcal{O}(1)$. Therefore, the overall time complexity for greedy-based method is $\mathcal{O}(n)$. For maximum-weight method, as discussed in Section III-C2, we need to find the node $S_{P_i} = \arg \max_{S \in \Lambda_i} P(S_i, S)$ for each step S_i , which requires

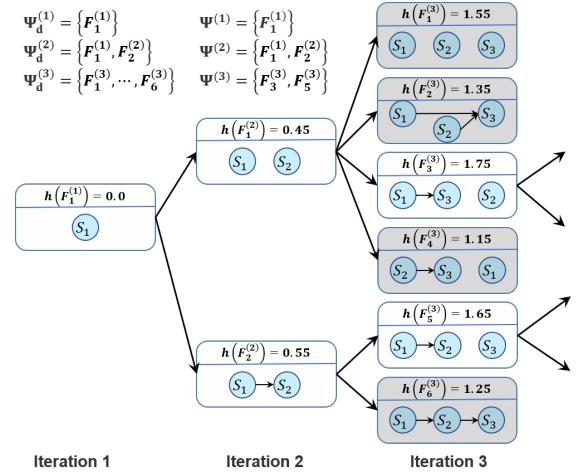


Fig. 3. An example illustrating the beam-PTC algorithm.

n^2 times of operations in total. Therefore, the overall time complexity for maximum-weight method is $\mathcal{O}(n^2)$.

For our proposed beam search-based algorithm, there are n iterations in total. For each iteration, \mathcal{L} number of states are expanded. In the extreme case, there are at most 2^n number of possible derived states for a state in the n -th iteration, which is computationally expensive when the number of step n is large. To avoid this, we employ an approximation strategy which randomly select M derived states when the number of possible derived states is larger than M . The overall time complexity is therefore $\mathcal{O}(n\mathcal{L}M)$. In experiments, we set $M = 512$.

IV. EXPERIMENTS

We separately evaluate the pairwise prerequisite relation detection and the prerequisite tree construction. We aim to explore the following questions: 1) What is the advantage of considering multi-modal information in detecting pairwise prerequisite relations, and 2) What are the effects when considering different tree construction strategies.

A. Dataset

The evaluation is based on the *Multi-modal Recipe Structure (MM-ReS)* [15] dataset, consisting of 9,850 real food recipes with labeled prerequisite trees. Each recipe contains an average of 11.26 cooking steps, where each step comprises of both textual instructions and multiple cooking images. As mentioned in the introduction, we want to explore whether our method works well with only limited amount of training data. If so, our method can quickly adapt to other food domains with only small efforts of gathering new training data. To this end, we randomly sampled 400 recipes from MM-ReS to obtain the Mini-MM-ReS dataset to train and evaluate our method. The key data statistics of MM-ReS and Mini-MM-ReS are summarized in Table I. The dataset has been released at <https://github.com/teacherpeterpan/The-MRePT-Dataset/>.

B. Pairwise Prerequisite Relation Detection

We first evaluate our method for pairwise prerequisite relation detection. The 400 recipes in the Mini-MM-ReS dataset

TABLE I
DATA STATISTICS OF MM-RES AND MINI-MM-RES.

Features	MM-Res	Mini-MM-Res
# Recipes	9,850	400
# Cooking steps	110,878	3,625
# Cooking images	227,082	8,314
# Sentences	143,580	15,418
Avg. steps / recipe	11.26	9.77
Avg. images / recipe	23.05	22.41
Avg. images / step	2.05	2.29

are randomly split into training (80%), validation (10%), and testing set (10%). For each set, we obtain the labeled step pairs with the method described in Section III-B1. In total, we obtain 6,447 step pairs for training, 718 pairs for validation, and 1,000 pairs for testing. Among the total 8,165 step pairs, 3,201 are positive samples and 4,964 are negative samples. We use precision (P), recall (R), and F_1 -score (F_1) as evaluation metrics.

1) *Baselines*: We perform a performance comparison among the following 7 methods, which can be categorized into 3 groups based on the use of information.

Text-Only. We first choose two baselines that only utilize instruction texts. Most related works for prerequisite detection can hardly be applied in our task, such as [22] and [34], as their features are highly domain-dependent. The only comparable work is **Jerm surawong et al., 2015** [14], in which several text-based features are proposed to detect prerequisite relations between recipe instructions, and an SVM classifier is trained for relation classification. To evaluate the effectiveness of deep features, we further apply BERT in the way as described in the late fusion method in Section III-B4. We name this baseline as **BERT Classification**.

Image-Only. To the best of our knowledge, there is no existing work that uses process images to predict prerequisite relation. Therefore, we propose the following two baselines that only utilize process images. The first weak baseline simply judge the prerequisite relation based on **image similarity**. After extracting image features using ResNet-50 described in Section III-B2, the image similarity between any two process images is then defined as the normalized cosine distance between their feature vectors. As one step is often associated with multiple cooking images, we propose the following three features. The *average image similarity* is the normalized cosine distance between the average image vector from step S_i and step S_j . We also incorporate the *maximum / minimum image similarity*, which is defined as the maximum/minimum similarity between the process images from S_i and S_j . These three features are then linearly combined to train an SVM classifier for relation detection. The second baseline applies the **Siamese network** as the prerequisite relation classifier, as described in Section III-B4.

Multi-Modal. We compare the following three methods that utilize both image and text information. We employ a baseline that enriches the feature set of [14] by adding the three image

similarity features as proposed above. We name this method as [14] + **Image Similarity**. The other two methods are the **Early Fusion** and the **Late Fusion** method that we proposed in Section III-B4.

2) *Performance Comparison*: We first report the performance of all the methods, and then investigate how multi-modality, neural features, and early/late fusion affects the results. The performance comparison results are presented in Table II. We discuss the results from the following three aspects.

Effects of extracting neural features. Within all three groups of methods (text-only, image-only, and multi-modal), the neural network-based model performs better than the method using hand-crafted features in prerequisite detection. For text-only models, when applying pre-trained BERT, it outperforms the hand-crafted features in [14] by a large margin. This is because BERT has been found to be able to capture high-level linguistic features such as sentence composition and semantic dependency, which are very important for this task. For image-only models, Siamese Network outperforms image similarity by 8.6 in F_1 score. The improvement is less significant as compared with text features. There are two potential reasons: (1) image similarity already serves as a strong clue in determining prerequisite relations, and (2) instructional texts are more informative than process images although they play a complementary role.

Effects of considering multi-modality. When adding the three image similarity features to [14], it achieves an F_1 score of 70.0, outperforming the original model by 5.4 in F_1 . This demonstrates the effectiveness of utilizing process images beyond text descriptions. Similar results are also observed when using deep neural features. BERT Classification and Siamese Network achieve F_1 score of 80.1 and 78.6, respectively. However, the performance improves to 82.2 in F_1 when using late fusion to combine the predictions from both sides. This further indicates that instructional texts and process images complement each other in detecting prerequisite relations. This point will be demonstrated more intuitively in IV-B3.

Early fusion and late fusion. For multi-modal feature fusion, early fusion and late fusion achieve comparable results of 81.4 and 82.2 in F_1 score. Late fusion slightly outperforms early fusion, which can be explained by the parameter size. For early fusion, as we combine image feature and text feature before classification, the number of parameters of early fusion is much larger than that of late fusion. This makes the model easy to overfit in training.

3) *Result Visualization and Error Analysis*: In Table III, we show some typical examples of predictions made by the best model (Late Fusion) in test set. The first two examples show that our model can make correct predictions based on both text and image clues. However, there are some typical mistakes that the model tends to make. We identify two major error types by analyzing the mis-classified samples.

From visual aspect, the error comes from *i.e.*, the lack of

TABLE II

THE PERFORMANCE COMPARISON ON THE TASK OF PAIRWISE PREREQUISITE RELATION DETECTION IN THE MINI-MM-RES DATASET. THE BEST PERFORMANCE FOR EACH VARIANT OF METHODS IS IN BOLD.

Models		<i>P</i>	<i>R</i>	<i>F₁</i>
Text-Only	Jermsurawong et al., 2015 [14]	68.5	68.4	68.4
	BERT Classification	79.8	80.4	80.1
Image-Only	Image Similarity	69.6	70.4	70.0
	Siamese Network	79.2	78.1	78.6
Multi-Modal	[14] + Image Similarity	73.8	74.6	74.0
	Early Fusion	82.4	80.8	81.4
	Late Fusion	82.1	82.3	82.2

fine-grained ingredient recognition. Sometimes, prerequisite relation is inferred from overlapping ingredients or tools in the image; however, this is hard to be captured by the overall image similarity. For example, in sample 3, despite that the sausage appears in both steps, their cooking images look visually different. This problem may be overcome by endowing the feature extractor with the ability to recognize ingredients. One potential way is to perform multi-task training of both image classification and ingredient recognition to fine-tune the image feature extractor.

For textual aspect, the error mainly comes from the *lack of context understanding*. For example, in the first step of sample 4, the burner actually contains rice, making this step relevant to the second step; however, we can tell the rice neither from the picture nor from the text. Therefore, context information is required in this case, as we can know from previous steps that the rice has already been put into the burner. To address this problem, prerequisite relation detection beyond pairwise comparison is required, which will be left to future exploration.

C. Prerequisite Tree Construction

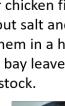
We now investigate how different strategies proposed in Section III-C impacts the prerequisite tree construction.

1) *Experimental Settings:* Given a recipe $R = (S_1, \dots, S_n)$, the task of constructing prerequisite tree is to find the prerequisite tree \mathcal{T}_R from all possible dependency trees of R , denoted as Ω_R . Formally, a valid dependency tree $T \in \Omega_R$ can be represented by the $n-1$ child-to-parent edges of the tree, *i.e.*, $\{\langle S_1, S_{P_1} \rangle, \langle S_{n-1}, S_{P_{n-1}} \rangle\}$, where $i < P_i$ for all i from 1 to $n-1$. Therefore, building the prerequisite tree is essentially determining the parents $S_{P_1}, \dots, S_{P_{n-1}}$ for each step from S_1 to S_{n-1} , where $P_i \in \{i+1, \dots, n-1\}$. Both the pairwise relation classifier and the tree construction strategy contribute to the output prerequisite tree. To investigate how they individually affect the result, we introduce two baselines as follows.

- **Random-PTC:** This is a weak baseline that employs a completely random strategy to build the tree without the help of classifier, *i.e.*, we randomly select each P_i from its value range $\{i+1, \dots, n-1\}$.
- **Cls-Random-PTC:** Instead of employing a certain strategy proposed in Section III-C, we apply a random strategy with the guidance of the pairwise relation classifier. In determining P_i , we first extract all $j \in \{i+1, \dots, n-1\}$

TABLE III

TYPICAL EXAMPLES OF MODEL PREDICTIONS. (G.T.) REFERS TO THE GROUND-TRUTH LABEL, AND (PRED.) REFERS TO THE MODEL PREDICTION. + AND - DENOTE THE SEQUENTIAL RELATION AND THE PARALLEL RELATION, RESPECTIVELY.

	Step Pair	Labels
1	   <p>Cut your chicken filets into little pieces, put salt and pepper, and brown them in a hot frying pan. Take the bay leaves out of the chicken stock.</p>  <p>Put the peanut butter in a big pan and add the chicken stock until it gets watery. Add the salty meat, the browned chicken filets, the tomatoes and the onion. Add stock cubes or Aromat according to taste.</p>	(G.T.) + (Pred.) +
2	  <p>Now its time to make the filling! Mix 1/3 cups of cream cheese, 1 teaspoon of vanilla, and a 1/2 a cup of powdered sugar on medium for 1 minute in your mixer. Then set it aside for later.</p> <p>Cut up 6-7 strawberries into thin slices.</p>	(G.T.) - (Pred.) -
3	  <p>I cut the sausage into 2 nice slices and throw them in the skillet.</p> <p>Dump those crazy mixed up eggs into the skillet!</p>	(G.T.) + (Pred.) -
4	  <p>Turn the burner down to the lowest temperature once the water is boiling, to simmer and apply lid for 20-25 minutes.</p> <p>Add the cooked rice to the wok along with the onions and garlic.</p>	(G.T.) + (Pred.) -

that satisfies $P(S_i, S_j) > \theta$ to form a new candidate set C_i . Then, P_i is randomly selected from C_i . When $C_i = \emptyset$, we select P_i completely randomly from $\{i+1, \dots, n-1\}$. By comparing the performance of our tree construction strategies with Cls-Random-PTC, we can qualitatively evaluate the additional benefit brought by the strategy without the interference of the pairwise classifier.

To evaluate the output prerequisite tree, we adopt the same metric as in [14], using the accuracy of predicting edges in the prerequisite tree. The overall accuracy of the task is computed at the *edge level* (counting all edges in the data set), and at the *recipe level* (average accuracy over all recipes).

2) *Performance of Different Strategies:* We evaluate the performance of our proposed strategies on the Mini-MM-ReS dataset. The 400 annotated recipes are randomly split into training, development, and test sets with the ratios of 70%, 10%, and 20%, respectively. The training set is used to train the pairwise relation classifier based on the late fusion model. The development set is used for tuning parameters. We report here the edge-level and recipe-level accuracy for each strategy on the test set as summarized in Table IV.

We have four observations. First, for random strategies, Cls-Random-PTC outperforms Random-PTC by 14.35% in edge-level accuracy. This demonstrates the effectiveness of

TABLE IV
PERFORMANCE OF PREREQUISITE TREE CONSTRUCTION (%).

	Strategy	Edge-level Accuracy	Recipe-level Accuracy
Baseline	Random-PTC	33.09	35.87
	Cls-Random-PTC	47.44	49.80
Proposed	Greedy-PTC ($\theta = 0.5$)	75.77	77.01
	Max-PTC	64.06	66.22
	Beam-PTC ($L = 2$)	79.84	80.21
	Beam-PTC ($L = 3$)	80.07	80.39

utilizing pairwise prerequisite relation classifier. Second, all of the proposed strategies outperform the Cls-Random-PTC by a large margin, ranging from 16.62% to 32.63% in edge-level accuracy. This reveals that employing appropriate strategies is vital to building the prerequisite tree from a noisy pairwise classifier with erroneous and contradictory predictions. Third, among the three strategies, greedy-PTC outperforms max-PTC by 11.71% in edge-level accuracy, showing that when finding the parent of S_i , it is better to give priority to closer subsequent steps rather than the highly-scored subsequent steps of S_i . Lastly, beam-PTC ($L = 3$) further improves the performance over greedy-PTC by 4.3% in edge-level accuracy. This is as expected because beam-PTC overcomes the “short-sighted” problem of greedy-PTC, by considering closer subsequent steps in priority while keeping track of the overall score at the same time.



Fig. 4. The accuracy with the change of hyper-parameters θ and L .

3) *Parameter Analysis:* We further investigate how hyper-parameters affect the performance including: (1) the threshold θ in the greedy-PTC, and (2) the number of intermediate states L in the beam-PTC. Figure 4(a) shows the performance of the greedy-PTC in the validation set for each θ ranging from 0.05 to 1.0. The performance is consistently good when θ ranges from 0.05 to 0.5 until a significant drop after $\theta > 0.8$. This reveals that a prerequisite relation is more common than an independent relation between two steps, as $1 - \theta$ essentially functions as the prior probability of a prerequisite relation existing between steps. Figure 4(b) shows the accuracy of the beam-PTC when L ranges from 1 to 10. The performance consistently increases when $L \leq 3$ but starts to drop afterwards. This shows that beam-PTC makes a balance between distance-centric strategy and score-centric strategy.

V. CONCLUSIONS

We have presented a two-step framework for automatically building prerequisite tree for cooking recipes, leveraging both text procedures and process images for pairwise prerequisite relation detection, and applying three tree construction strategies for prerequisite tree construction. Experimental results on a challenging dataset (MM-ReS) show that considering multi-modal features enables better performance for prerequisite tree construction, and the process images are highly complementary to procedure text.

While encouraging, the current work requires further investigation in at least two directions. First, user-posted cooking steps are often noisy and coarse-grained, e.g., many parallel processes are written within a single step. Therefore, constructing fine-grained prerequisite tree via automatic cooking step segmentation is an important direction. Second, applying multi-modal prerequisite tree to downstream applications such as cross-modal retrieval is also useful. For example, representation learning on prerequisite tree should benefit applications such as recipe classification and recipe retrieval.

ACKNOWLEDGMENT

This research is supported by the National Research Foundation, Singapore under its International Research Centres in Singapore Funding Initiative. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore.

REFERENCES

- [1] W. Min, B. Bao, S. Mei, Y. Zhu, Y. Rui, and S. Jiang, “You are what you eat: Exploring rich recipe information for cross-region food analysis,” *IEEE Transactions on Multimedia (TMM)*, vol. 20, no. 4, pp. 950–964, 2018.
- [2] R. Xu, L. Herranz, S. Jiang, S. Wang, X. Song, and R. Jain, “Geolocalized modeling for dish recognition,” *IEEE Transactions on Multimedia (TMM)*, vol. 17, no. 8, pp. 1187–1199, 2015.
- [3] L. Herranz, S. Jiang, and R. Xu, “Modeling restaurant context for food recognition,” *IEEE Transactions on Multimedia (TMM)*, vol. 19, no. 2, pp. 430–440, 2017.
- [4] T. Maruyama, Y. Kawano, and K. Yanai, “Real-time mobile recipe recommendation system using food ingredient recognition,” in *Proceedings of IMMPD*, 2012, pp. 27–34.

- [5] S. Horiguchi, S. Amano, M. Ogawa, and K. Aizawa, "Personalized classifier for food image recognition," *IEEE Transactions on Multimedia (TMM)*, vol. 20, no. 10, pp. 2836–2848, 2018.
- [6] A. Salvador, N. Hynes, Y. Aytar, J. Marin, F. Oflí, I. Weber, and A. Torralba, "Learning cross-modal embeddings for cooking recipes and food images," in *Proceedings of CVPR*, 2017, pp. 3020–3028.
- [7] J.-j. Chen, C.-W. Ngo, and T.-S. Chua, "Cross-modal recipe retrieval with rich food attributes," in *Proceedings of ACM-MM*, 2017, pp. 1771–1779.
- [8] W. Min, S. Jiang, J. Sang, H. Wang, X. Liu, and L. Herranz, "Being a supercook: Joint food attributes and multimodal content modeling for recipe retrieval and exploration," *IEEE Transactions on Multimedia (TMM)*, vol. 19, no. 5, pp. 1100–1113, 2017.
- [9] M. Carvalho, R. Cadène, D. Picard, L. Soulíer, N. Thome, and M. Cord, "Cross-modal retrieval in the cooking context: Learning semantic text-image embeddings," in *Proceedings of SIGIR*, 2018, pp. 35–44.
- [10] C. Kiddon, G. T. Ponnuraj, L. Zettlemoyer, and Y. Choi, "Mise en place: Unsupervised interpretation of instructional recipes," in *Proceedings of EMNLP*, 2015, pp. 982–992.
- [11] Y. Yamakata, S. Imahori, H. Maeta, and S. Mori, "A method for extracting major workflow composed of ingredients, tools, and actions from cooking procedural text," in *Proceedings of ICMEW*, 2016, pp. 1–6.
- [12] Y. Yamakata, H. Maeta, T. Kadokami, T. Sasada, S. Imahori, and S. Mori, "Cooking recipe search by pairs of ingredient and action—word sequence vs flow-graph representation—," *Transactions of the Japanese Society for Artificial Intelligence*, vol. 32, no. 1, pp. WII-F_1, 2017.
- [13] J. Chung, S. Ahn, and Y. Bengio, "Hierarchical multiscale recurrent neural networks," in *Proceedings of ICLR*, 2017.
- [14] J. Jermusurawong and N. Habash, "Predicting the structure of cooking recipes," in *Proceedings of EMNLP*, 2015, pp. 781–786.
- [15] L. Pan, J. Chen, J. Wu, S. Liu, C. Ngo, M. Kan, Y. Jiang, and T. Chua, "Multi-modal cooking workflow construction for food recipes," in *Proceedings of ACMMM*, 2020, pp. 1132–1141.
- [16] D. Kiela, S. Bhooshan, H. Firooz, and D. Testuggine, "Supervised multimodal bitransformers for classifying images and text," in *Proceedings of NeurIPS 2019 Workshop*, 2019.
- [17] L. Wang, Q. Li, N. Li, G. Dong, and Y. Yang, "Substructure similarity measurement in chinese recipes," in *Proceedings of WWW*, 2008, pp. 979–988.
- [18] Y. Yamakata, S. Imahori, Y. Sugiyama, S. Mori, and K. Tanaka, "Feature extraction and summarization of recipes using flow graph," in *Proceedings of SocInfo*, 2013, pp. 241–254.
- [19] R. Hamada, J. Okabe, I. Ide, S. Satoh, S. Sakai, and H. Tanaka, "Cooking navi: assistant for daily cooking in kitchen," in *Proceedings of ACM-MM*, 2005, pp. 371–374.
- [20] S. Karikome and A. Fujii, "Improving structural analysis of cooking recipe text," *IEICE technical report. Data engineering*, vol. 112, no. 75, pp. 43–48, 2012.
- [21] K. Walter, M. Minor, and R. Bergmann, "Workflow extraction from cooking recipes," in *Proceedings of the ICCBR 2011 Workshops*, 2011, pp. 207–216.
- [22] L. Pan, C. Li, J. Li, and J. Tang, "Prerequisite relation learning for concepts in moocs," in *Proceedings of ACL*, vol. 1, 2017, pp. 1447–1456.
- [23] C. Liang, J. Ye, Z. Wu, B. Pursel, and C. L. Giles, "Recovering concept prerequisite relations from university course dependencies," in *Proceedings of AAAI*, 2017, pp. 4786–4791.
- [24] Y. Yang, H. Liu, J. Carbonell, and W. Ma, "Concept graph learning from educational data," in *Proceedings of WSDM*, 2015, pp. 159–168.
- [25] J. Liu, L. Jiang, Z. Wu, Q. Zheng, and Y. Qian, "Mining learning dependency between knowledge units from text," *The VLDB Journal*, vol. 20, no. 3, pp. 335–345, 2011.
- [26] A. Vuong, T. Nixon, and B. Towle, "A method for finding prerequisites within a curriculum," in *Proceedings of EDM*, 2011, pp. 211–216.
- [27] R. Scheines, E. Silver, and I. M. Goldin, "Discovering prerequisite relationships among knowledge components," in *Proceedings of EDM*, 2014, pp. 355–356.
- [28] X. Huang, K. Yang, and V. B. Lawrence, "An efficient data mining approach to concept map generation for adaptive learning," in *Proceedings of ICDM*, 2015, pp. 247–260.
- [29] L. Pan, X. Wang, C. Li, J. Li, and J. Tang, "Course concept extraction in moocs via embedding-based graph propagation," in *Proceedings of IJCNLP*, 2017, pp. 875–884.
- [30] C. Liang, Z. Wu, W. Huang, and C. L. Giles, "Measuring prerequisite relations among concepts," in *Proceedings of EMNLP*, 2015, pp. 1668–1674.
- [31] P. P. Talukdar and W. W. Cohen, "Crowdsourced comprehension: predicting prerequisite structure in wikipedia," in *Workshop on Building Educational Applications Using NLP*, 2012, pp. 307–315.
- [32] S. Wang, A. Ororbia, Z. Wu, K. Williams, C. Liang, B. Pursel, and C. L. Giles, "Using prerequisites to extract concept maps from textbooks," in *Proceedings of CIKM*, 2016, pp. 317–326.
- [33] J. Gordon, L. Zhu, A. Galstyan, P. Natarajan, and G. Burns, "Modeling concept dependencies in a scientific corpus," in *Proceedings of ACL*, vol. 1, 2016, pp. 866–875.
- [34] C. Liang, J. Ye, Z. Wu, B. Pursel, and C. L. Giles, "Recovering concept prerequisite relations from university course dependencies," in *Proceedings of AAAI*, 2017, pp. 4786–4791.
- [35] J. Chen, L. Pan, Z. Wei, X. Wang, C. Ngo, and T. Chua, "Zero-shot ingredient recognition by multi-relational graph convolutional network," in *Proceedings of AAAI*, 2020, pp. 10 542–10 550.
- [36] M. Carvalho, R. Cadène, D. Picard, L. Soulíer, N. Thome, and M. Cord, "Cross-modal retrieval in the cooking context: Learning semantic text-image embeddings," in *Proceedings of SIGIR*, 2018, pp. 35–44.
- [37] B. Zhu, C. Ngo, J. Chen, and Y. Hao, "R2GAN: cross-modal recipe retrieval with generative adversarial network," in *Proceedings of CVPR*, 2019, pp. 11 477–11 486.
- [38] H. Wang, D. Sahoo, C. Liu, E. Lim, and S. C. H. Hoi, "Learning cross-modal embeddings with adversarial networks for cooking recipes and food images," in *Proceedings of CVPR*, 2019, pp. 11 572–11 581.
- [39] A. Salvador, M. Drozdal, X. Giró-i-Nieto, and A. Romero, "Inverse cooking: Recipe generation from food images," in *Proceedings of CVPR*, 2019, pp. 10 453–10 462.
- [40] Z. Ming, J. Chen, Y. Cao, C. Forde, C. Ngo, and T. Chua, "Food photo recognition for dietary tracking: System and experiment," in *Proceedings of MMM*, vol. 10705, 2018, pp. 129–141.
- [41] L. Deng, J. Chen, Q. Sun, X. He, S. Tang, Z. Ming, Y. Zhang, and T. Chua, "Mixed-dish recognition with contextual relation networks," in *Proceedings of ACMMM*, 2019, pp. 112–120.
- [42] J. Chen and C. Ngo, "Deep-based ingredient recognition for cooking recipe retrieval," in *Proceedings of ACMMM*, 2016, pp. 32–41.
- [43] L. Pan, S. Pouyanfar, H. Chen, J. Qin, and S. Chen, "Deepfood: Automatic multi-class classification of food ingredients using deep learning," in *Proceedings of CIC*, 2017, pp. 181–189.
- [44] J.-J. Chen, L. Pan, Z. Wei, X. Wang, C.-W. Ngo, and T.-S. Chua, "Zero-shot ingredient recognition by multi-relational graph convolutional network," in *Proceedings of AAAI*, 2020, pp. 10 542–10 550.
- [45] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of NeurIPS*, 2012, pp. 1106–1114.
- [46] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and F. Li, "Imagenet: A large-scale hierarchical image database," in *Proceedings of CVPR*, 2009, pp. 248–255.
- [47] A. Myers, N. Johnston, V. Rathod, A. Korattikara, A. N. Gorban, N. Silberman, S. Guadarrama, G. Papandreou, J. Huang, and K. Murphy, "Im2calories: Towards an automated mobile vision food diary," in *Proceedings of ICCV*, 2015, pp. 1233–1241.
- [48] X. Wang, D. Kumar, N. Thome, M. Cord, and F. Precioso, "Recipe recognition with large multimodal food dataset," in *Proceedings of ICME Workshops*, 2015, pp. 1–6.
- [49] J. Harashima, Y. Someya, and Y. Kikuta, "Cookpad image dataset: An image collection as infrastructure for food research," in *Proceedings of SIGIR*, 2017, pp. 1229–1232.
- [50] J. Marin, A. Biswas, F. Oflí, N. Hynes, A. Salvador, Y. Aytar, I. Weber, and A. Torralba, "Recipe1m+: A dataset for learning cross-modal embeddings for cooking recipes and food images," *IEEE transactions on pattern analysis and machine intelligence*, 2019.
- [51] A. Salvador, N. Hynes, Y. Aytar, J. Marin, F. Oflí, I. Weber, and A. Torralba, "Learning cross-modal embeddings for cooking recipes and food images," in *Proceedings of CVPR*, 2017, pp. 3068–3076.
- [52] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of CVPR*, 2016, pp. 770–778.
- [53] J. Chen and C.-W. Ngo, "Deep-based ingredient recognition for cooking recipe retrieval," in *Proceedings of ACM-MM*, 2016, pp. 32–41.
- [54] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *Proceedings of NAACL-HLT*, 2018, pp. 2227–2237.
- [55] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," in *Proceedings of ACL*, 2018, pp. 328–339.
- [56] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," in *Proceedings of NAACL-HLT*, 2019, pp. 4171–4186.

- [57] Z. Yang, Z. Dai, Y. Yang, J. G. Carbonell, R. Salakhutdinov, and Q. V. Le, "Xlnet: Generalized autoregressive pretraining for language understanding," in *Proceedings of NeurIPS*, 2019, pp. 5754–5764.
- [58] L. Dong, N. Yang, W. Wang, F. Wei, X. Liu, Y. Wang, J. Gao, M. Zhou, and H. Hon, "Unified language model pre-training for natural language understanding and generation," in *Proceedings of NeurIPS*, 2019, pp. 13 042–13 054.
- [59] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proceedings of NIPS*, 2017, pp. 6000–6010.
- [60] Y. Zhu, R. Kiros, R. S. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler, "Aligning books and movies: Towards story-like visual explanations by watching movies and reading books," in *Proceedings of ICCV*, 2015, pp. 19–27.
- [61] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," in *ICML deep learning workshop*, vol. 2, 2015.



Chong-Wah Ngo received the B.Sc. and M.Sc. degrees in computer engineering from Nanyang Technological University, Singapore, and the Ph.D. in computer science from the Hong Kong University of Science and Technology (HKUST), Hong Kong. He is currently a Professor with the Department of Computer Science, City University of Hong Kong, Hong Kong. Before joining the City University of Hong Kong, he was a Postdoctoral Scholar with the Beckman Institute, the University of Illinois at Urbana-Champaign (UIUC), Urbana, IL, USA. He was also a Visiting Researcher with Microsoft Research Asia, Beijing, China. His research interests include large-scale multimedia information retrieval, video computing, multimedia mining, and visualization. Prof. Ngo was the Associate Editor of the IEEE TRANSACTIONS ON MULTIMEDIA (2011–2014). He was the Conference Co-Chair of the ACM International Conference on Multimedia Retrieval 2015 and the Pacific Rim Conference on Multimedia 2014. He also served as Program Co-Chair of ACM Multimedia Modeling 2012 and ICMR 2012. He was the Chairman of ACM (Hong Kong Chapter) from 2008 to 2009.



Liangming Pan is a third year Computer Science Ph.D. student at National University of Singapore, jointly advised by Prof. Min-Yen Kan and Prof. Tat-Seng Chua. He received a Master degree from School of Computer Science at Tsinghua University in June, 2017 and obtained the Bachelor degree in Beihang University (2010 - 2014). His board research interests include knowledge base, natural language processing, and data mining. His Ph.D. research topics include neural question generation, text style transfer, and multi-modal recipe structure prediction. He has published several research papers in top-ranked conferences such as ACL, AAAI, ACM MM, and CVPR. He served as the Senior Program Committee (SPC) member in IJCAI 2021. He is also a 2019/2020 Research Achievement Award winner of NUS School of Computing.



Min-Yen Kan is an associate professor at the National University of Singapore. He is a senior member of the ACM and a member of the IEEE. Currently, he is an associate editor for the journal "Information Retrieval" and is the Editor for the ACL Anthology, the computational linguistics community's largest archive of published research. His research interests include digital libraries and applied natural language processing. Specific projects include work in the areas of scientific discourse analysis, full-text literature mining, machine translation and applied text summarization.



Jingjing Chen is now a pre-tenured associate professor at the School of Computer Science, Fudan University. Before joining Fudan University, she was a postdoc research fellow at the School of Computing in the National University of Singapore. She received her Ph.D. degree in Computer Science from the City University of Hong Kong in 2018. Her research interest lies in diet tracking and nutrition estimation based on multi-modal processing of food images, including food recognition, cross-modal recipe retrieval.



Tat-Seng Chua is the KITHCT Chair Professor at the School of Computing, National University of Singapore. He was the Acting and Founding Dean of the School from 1998–2000. Prof. Chua's main research interest is in multimedia information retrieval and social media analytics. In particular, his research focuses on the extraction, retrieval and question-answering (QA) of text and rich media arising from the Web and multiple social networks. He is the co-Director of NExT, a joint Center between NUS and Tsinghua University to develop technologies for live social media search. Prof. Chua is the 2015 winner of the prestigious ACM SIGMM Award for Outstanding Technical Contributions to Multimedia Computing, Communications and Applications. He is the Chair of steering committee of the ACM International Conference on Multimedia Retrieval (ICMR) and Multimedia Modeling (MMM) conference series. Prof. Chua is also the General Co-Chair of ACM Multimedia 2005, ACM CIVR (now ACM ICMR) 2005, ACM SIGIR 2008, and ACMWeb Science 2015. He serves on the editorial boards of four international journals. Dr. Chua is the co-Founder of two technology startup companies in Singapore. He holds a Ph.D. from the University of Leeds, UK.



Shaoteng Liu is a 4th-year undergraduate student major in Automation, at Xi'an Jiaotong University. During 2019 Spring and Summer, He worked as a research intern in NExT lab, National University of Singapore.